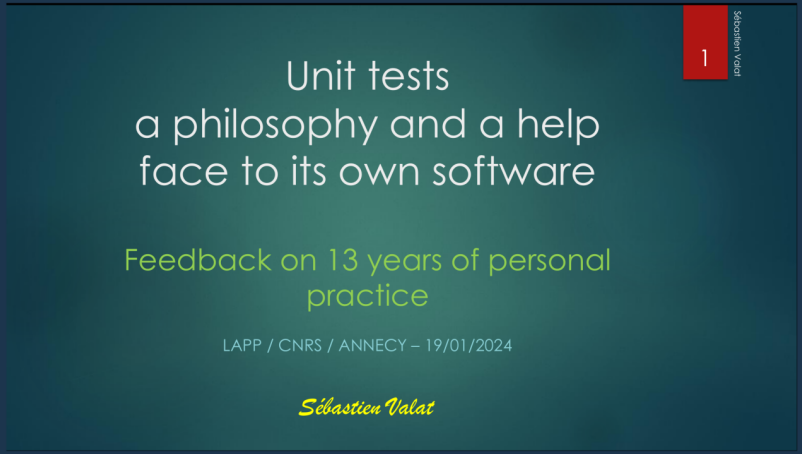
LAPP
Laboratoire d'Annecy de Physique des Particules

Se mocker (enfin)
Du Hardware

Vincent Pollet, Pierre Aubert

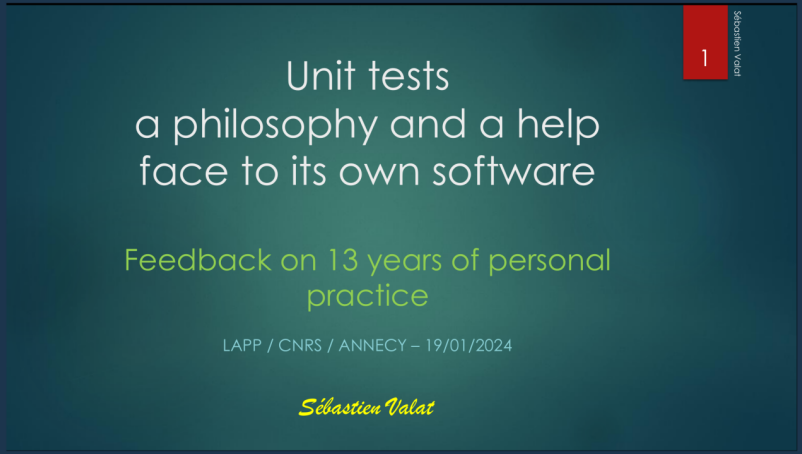
ESCAPE
cta cherenkov telescope array
cnrs
UNIVERSITÉ SAVOIE MONT BLANC

Introduction



Seminar on Unit Tests

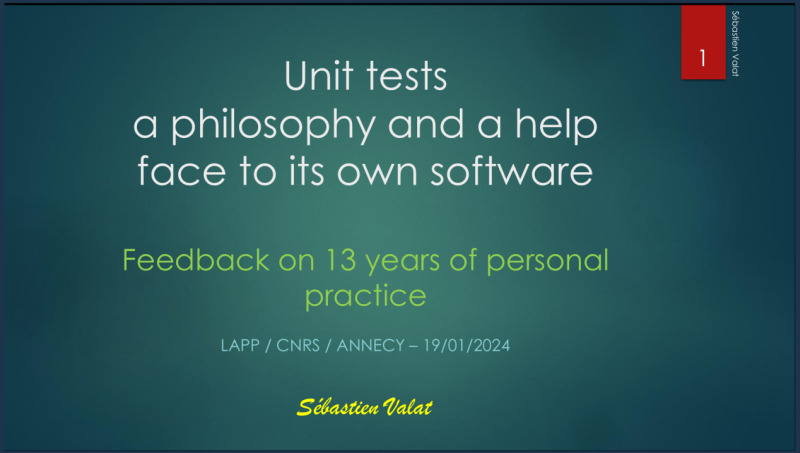
Introduction



Seminar on Unit Tests

Possibility to **Mock network**

Introduction

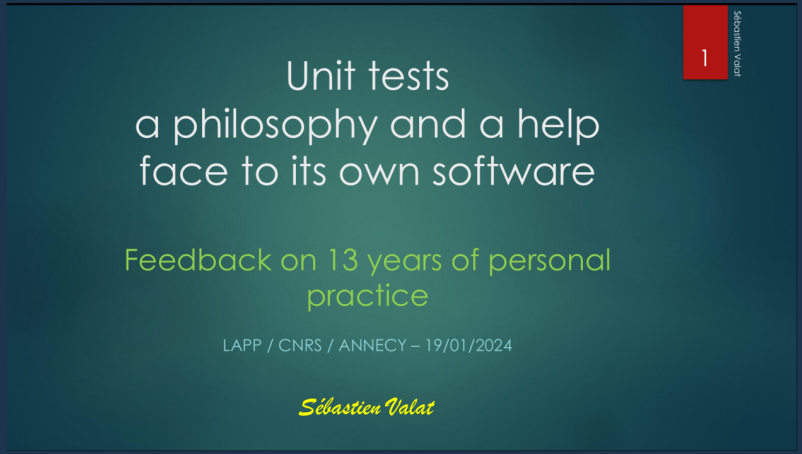


Seminar on Unit Tests

Possibility to **Mock network**

Mock **LHCb DAq** on a **laptop**

Introduction



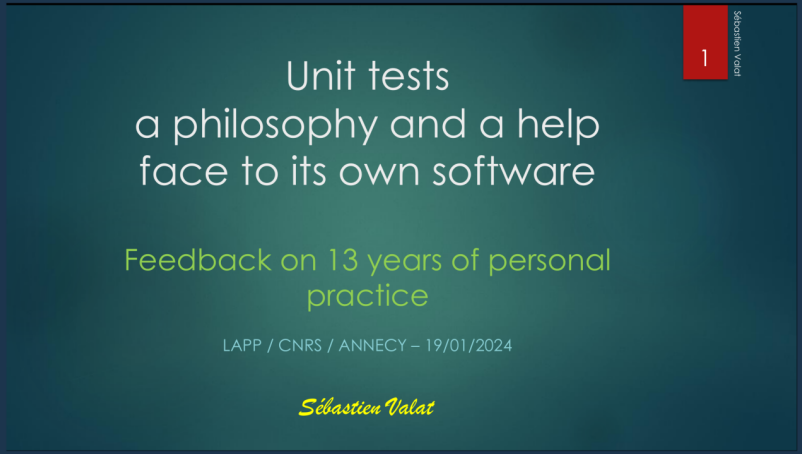
Seminar on Unit Tests

Possibility to **Mock network**

Mock **LHCb DAq** on a **laptop**

We could **mock only server interactions**

Introduction



Seminar on Unit Tests

Possibility to **Mock network**

Mock **LHCb DAq** on a **laptop**

We could **mock only server interactions**

No implementation of a mock server **needed**

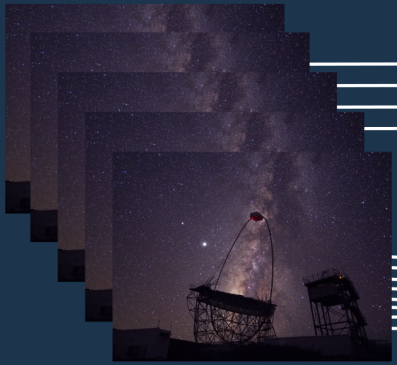
Motivation

Motivation

Motivation

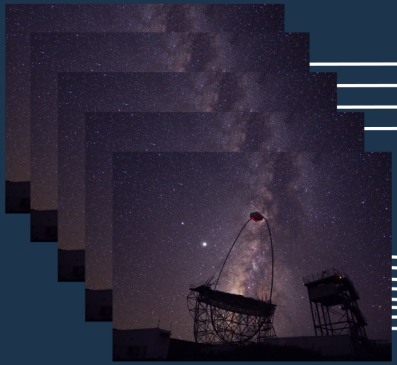
hipeRTA

Motivation



x19 telescopes North Site
x100 Telescopes South Site

Motivation

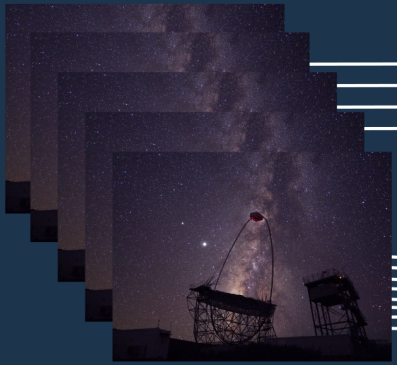


x19 telescopes North Site
x100 Telescopes South Site

hipeRTA

How To **Unit Test** It ?

Motivation



x19 telescopes North Site
x100 Telescopes South Site

hipeRTA

How To **Unit Test** It ?

- On our **laptops**
- On **Gitlab CI**

Mock of Sockets ?

Mock of Sockets ?

Camera

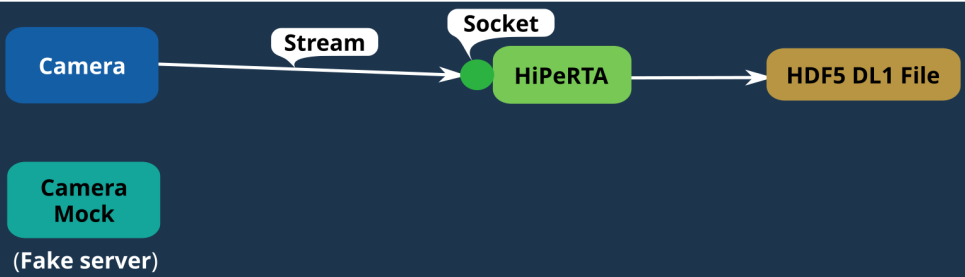
Mock of Sockets ?

Mock of Sockets ?

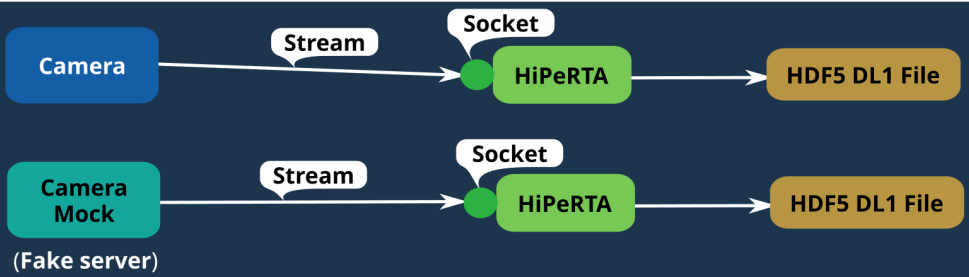
Mock of Sockets ?

Mock of Sockets ?

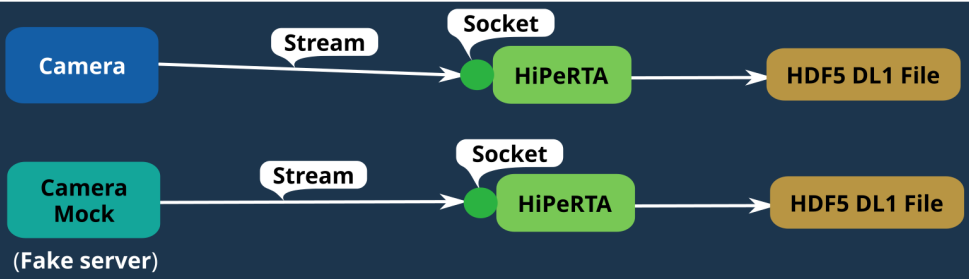
Mock of Sockets ?



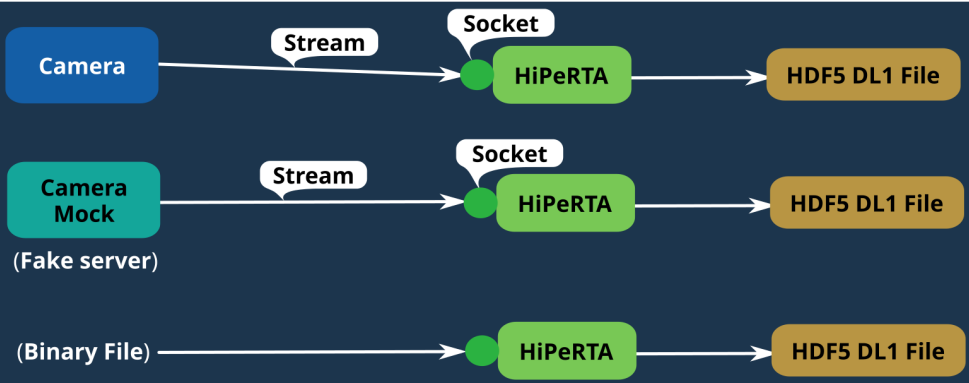
Mock of Sockets ?



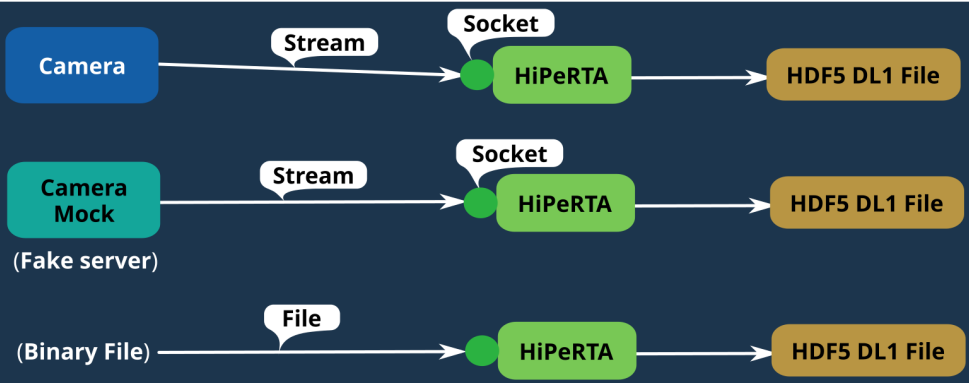
Mock of Sockets ?



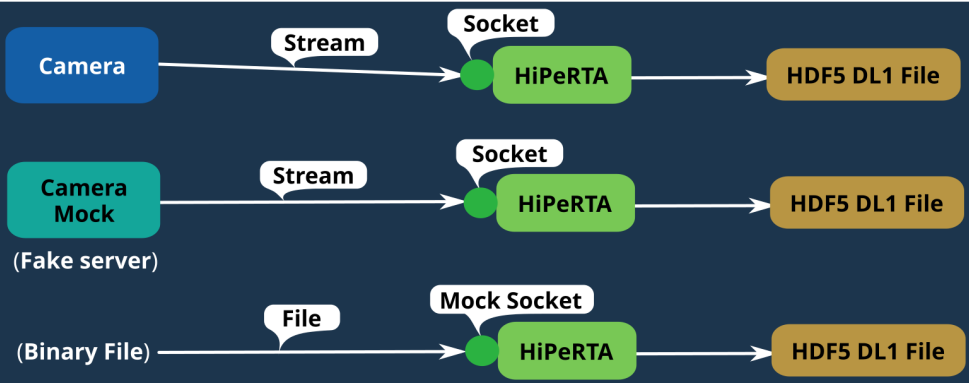
Mock of Sockets ?



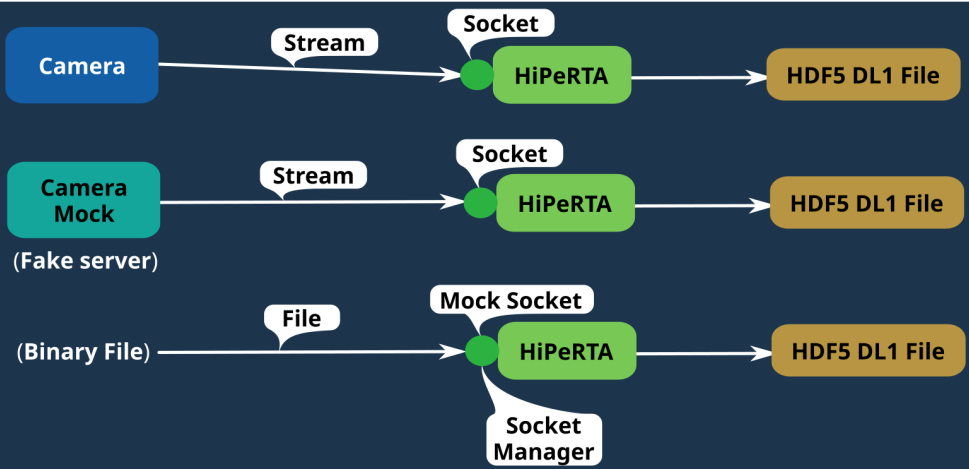
Mock of Sockets ?



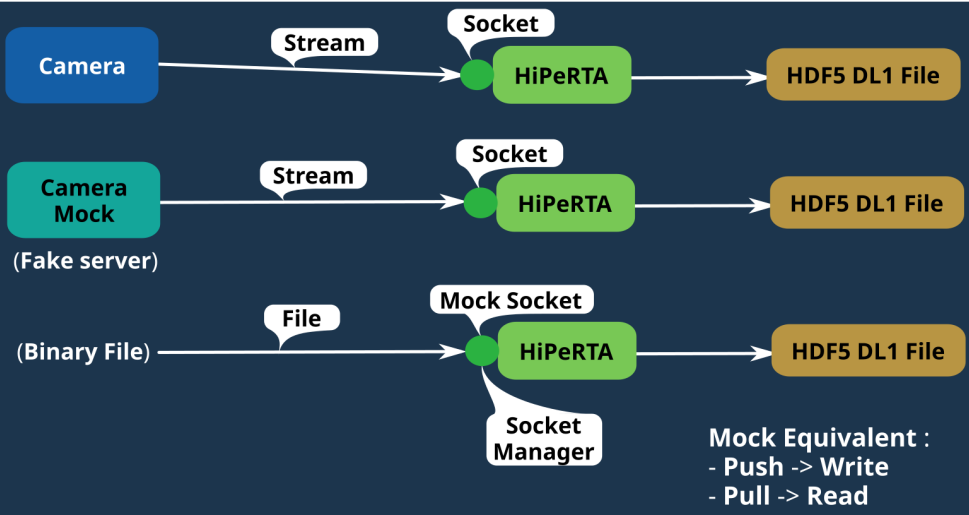
Mock of Sockets ?



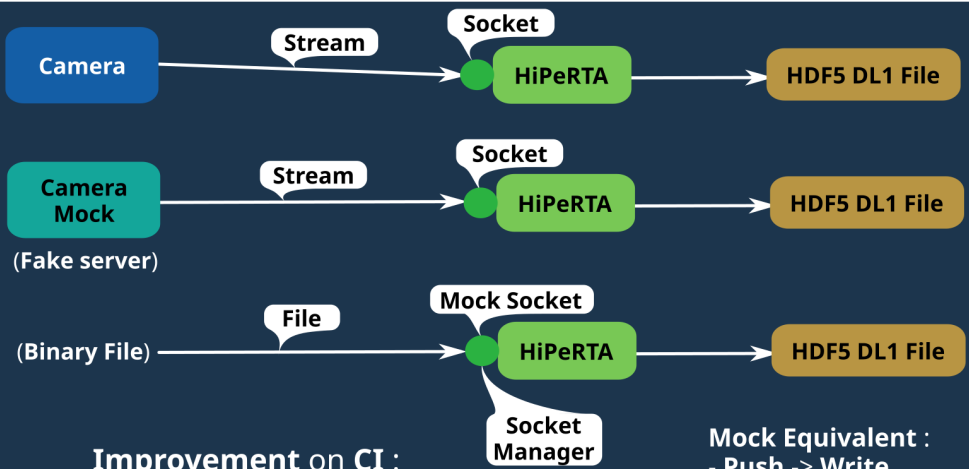
Mock of Sockets ?



Mock of Sockets ?



Mock of Sockets ?



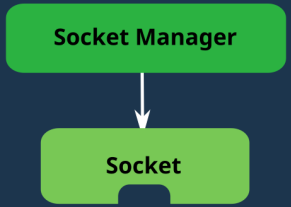
Improvement on **CI** :
- Unit tests use to take **40s**
- Now only **5s**

Mock Equivalent :
- **Push** -> **Write**
- **Pull** -> **Read**

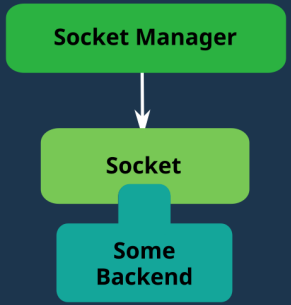
Mock Sockets Design

Mock Sockets Design

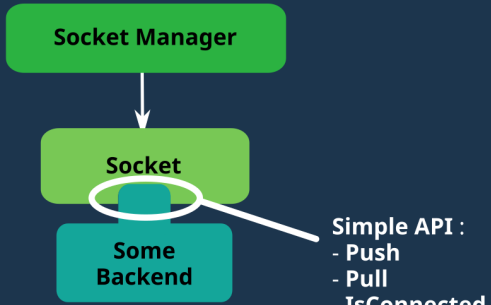
Mock Sockets Design



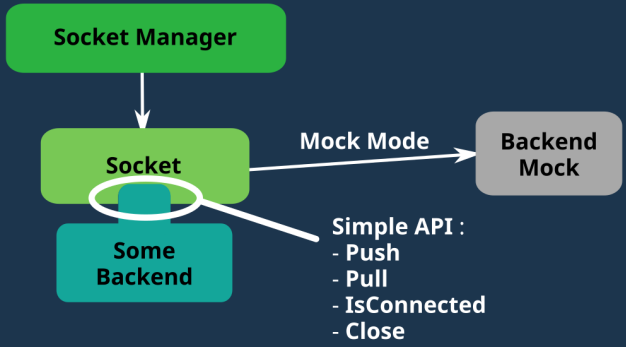
Mock Sockets Design



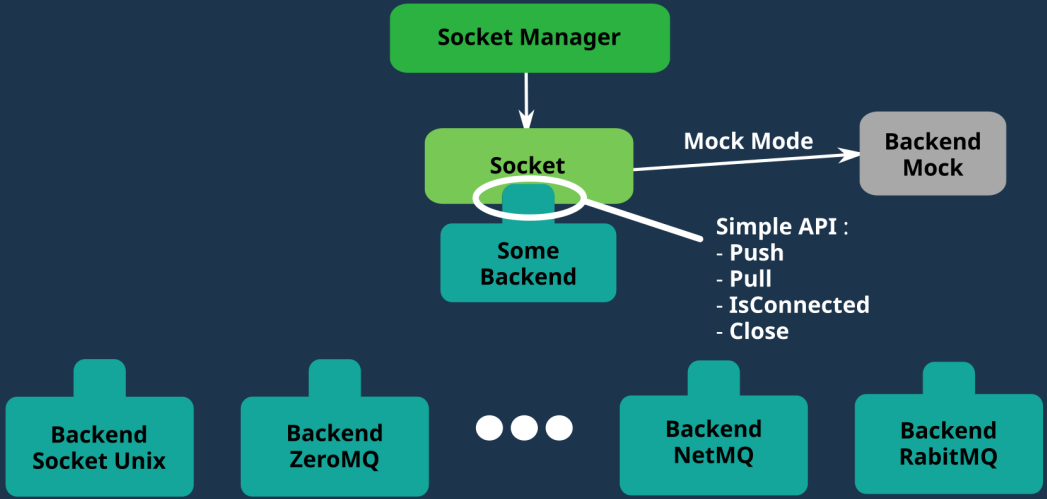
Mock Sockets Design



Mock Sockets Design

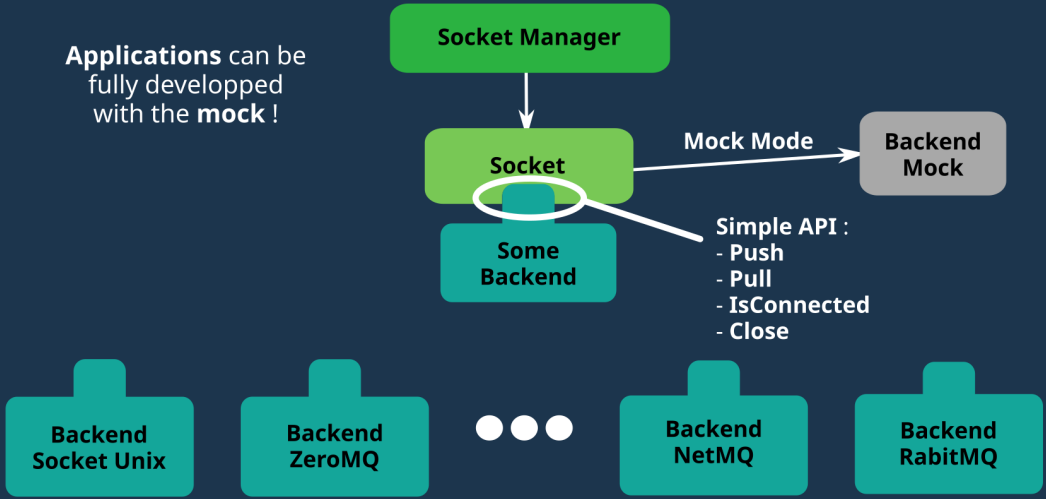


Mock Sockets Design



Mock Sockets Design

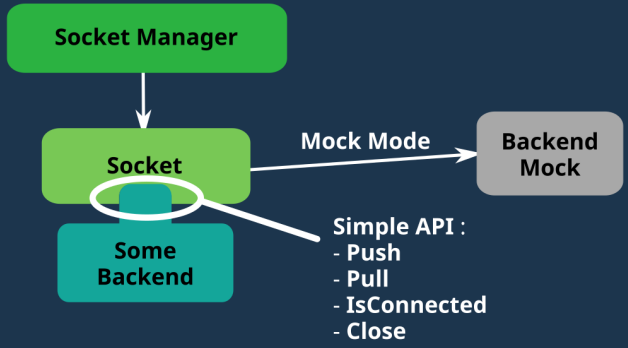
Applications can be fully developped with the **mock** !



Mock Sockets Design

Applications can be fully developped with the **mock** !

The **production backend** can be **choosen later**



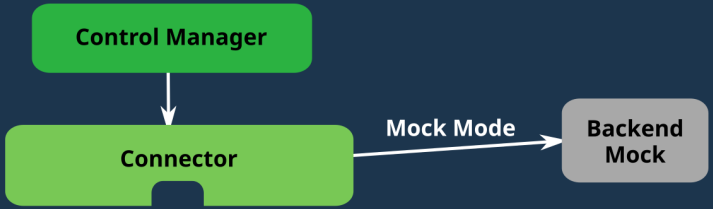
Extending Socket Backend

Extending Socket Backend

We could make **Sockets more generic**

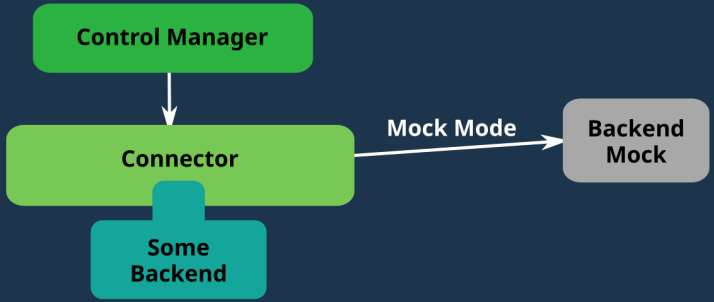
Extending Socket Backend

We could make **Sockets more generic**



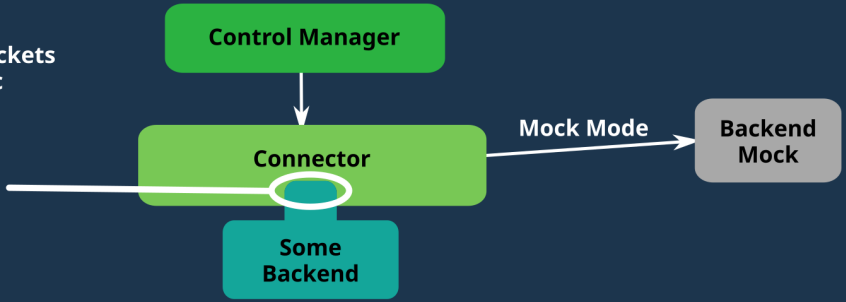
Extending Socket Backend

We could make **Sockets more generic**



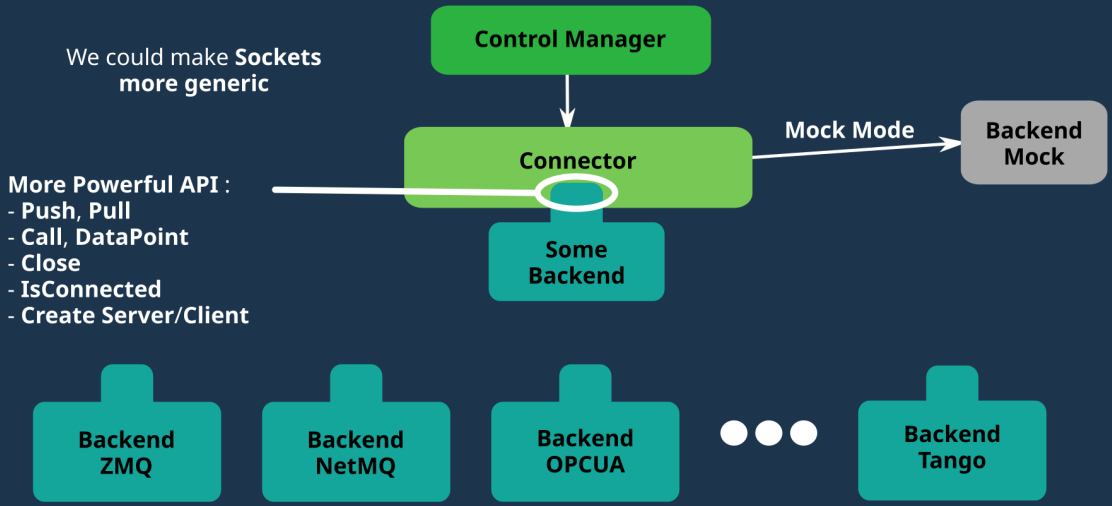
Extending Socket Backend

We could make **Sockets more generic**

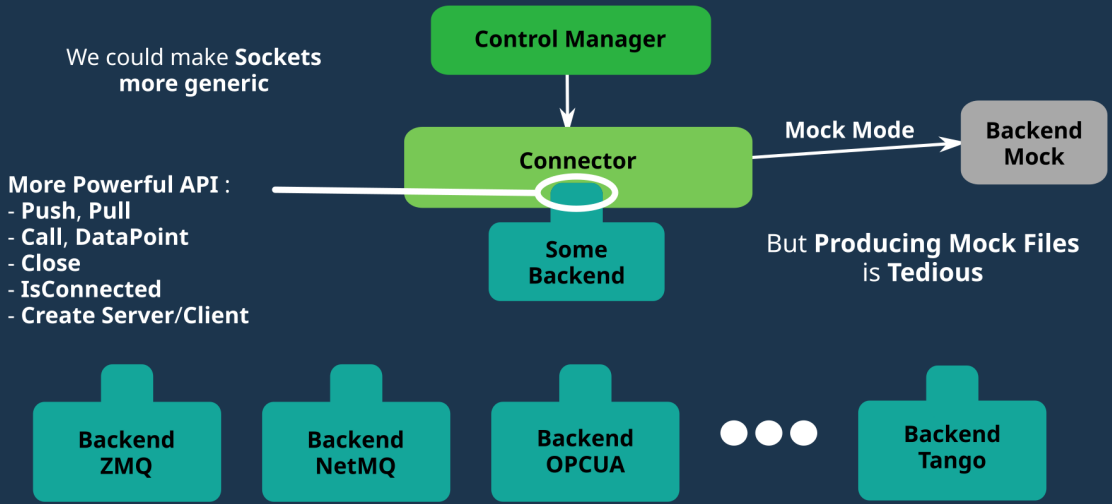


More Powerful API :
- **Push**, **Pull**
- **Call**, **DataPoint**
- **Close**
- **IsConnected**
- **Create Server/Client**

Extending Socket Backend



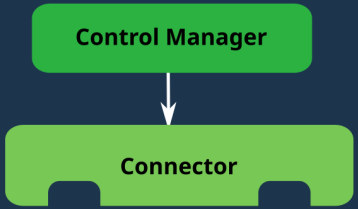
Extending Socket Backend



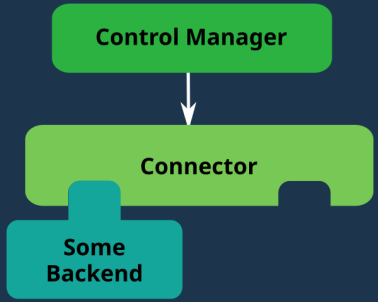
Mock Control Design

Mock Control Design

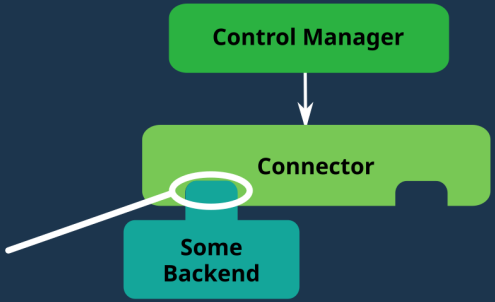
Mock Control Design



Mock Control Design

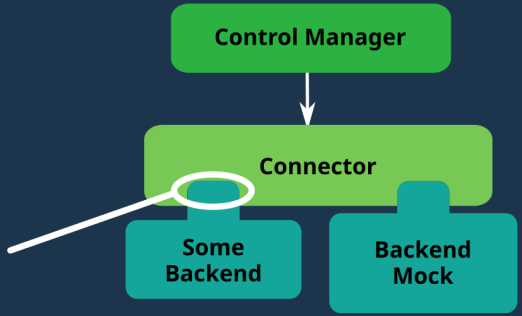


Mock Control Design



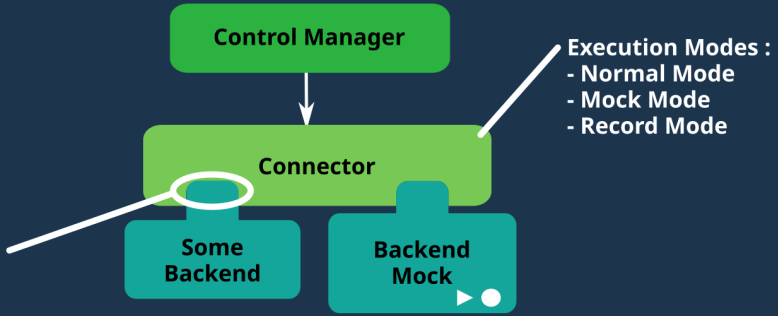
More Powerful API :
- **Push, Pull**
- **Call, DataPoint**
- **Close**
- **IsConnected**
- **Create Server/Client**

Mock Control Design

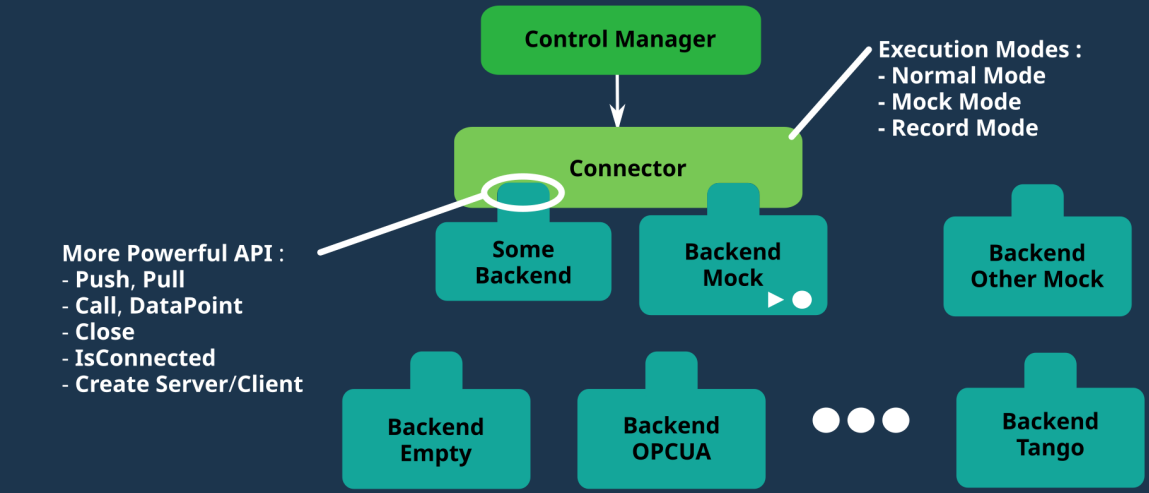


More Powerful API :
- **Push, Pull**
- **Call, DataPoint**
- **Close**
- **IsConnected**
- **Create Server/Client**

Mock Control Design



Mock Control Design

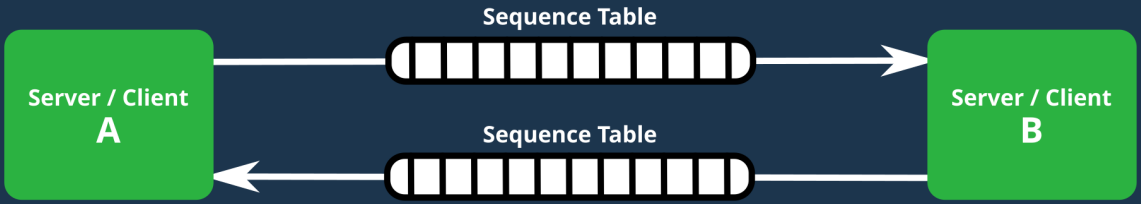


Normal Use and Mock

Normal Use

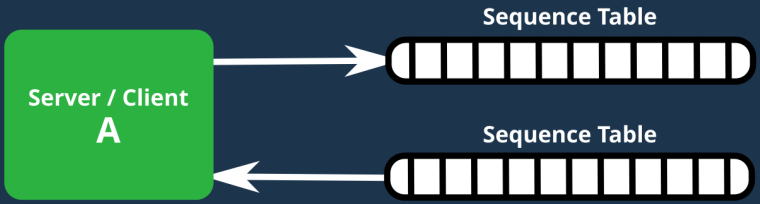
Normal Use and Mock

Normal Use

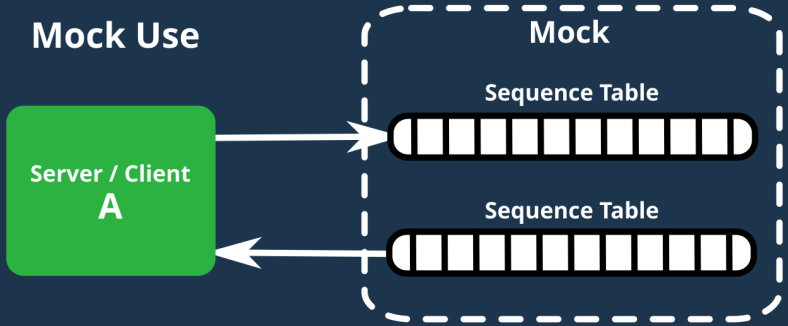


Normal Use and Mock

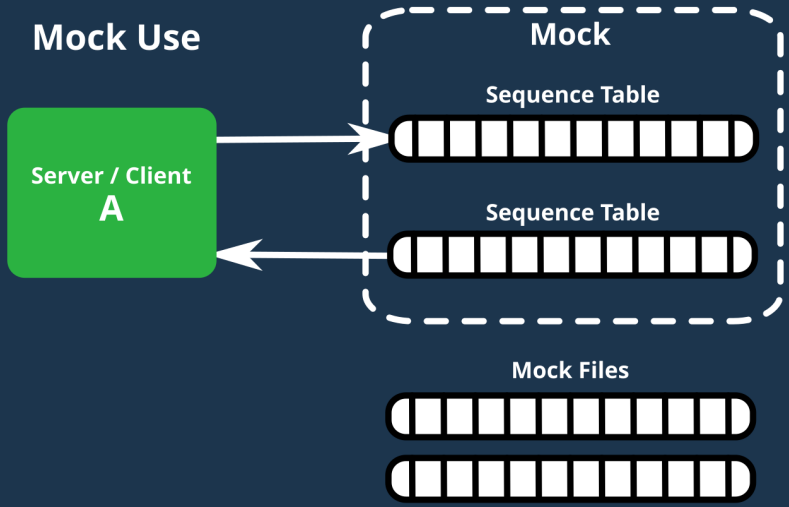
Mock Use



Normal Use and Mock

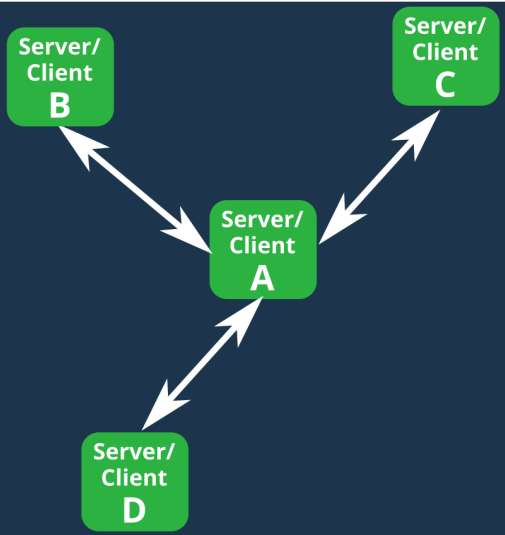


Normal Use and Mock

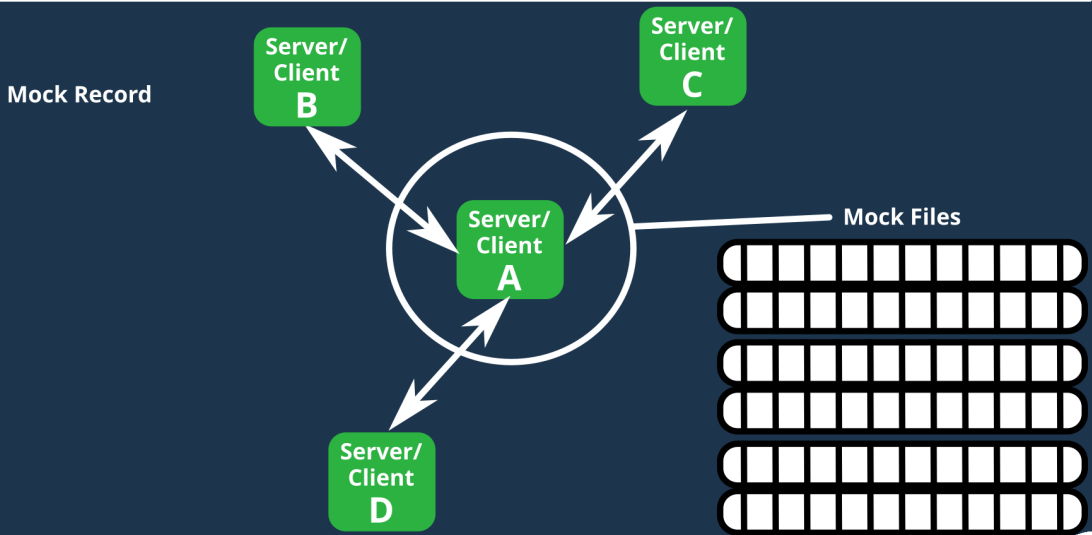


Normal Use and Mock

Normal Use

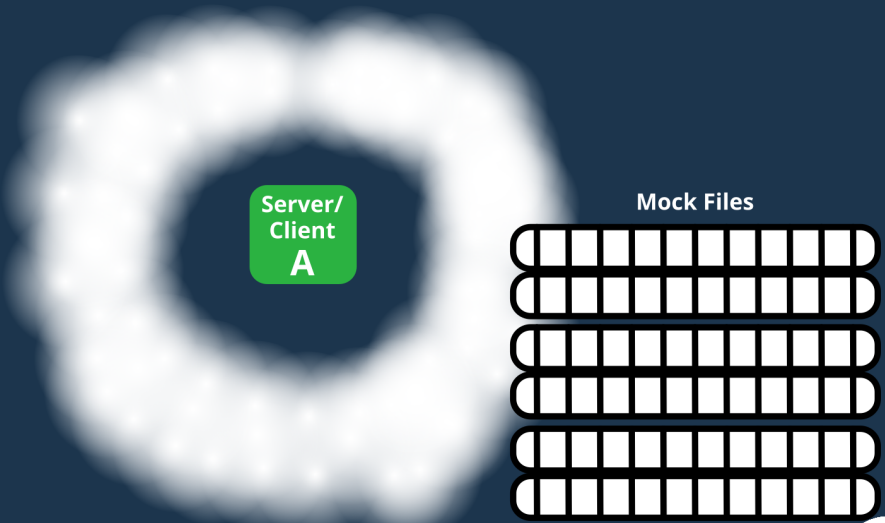


Normal Use and Mock



Normal Use and Mock

Mock Use



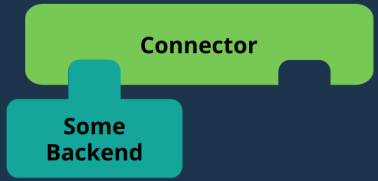
Sequence Table of A

Event Id and/or Time	State of A	Client/Sever : B			Client/Sever : C	
		Data Point B1	Calls From B of A1(X)		Calls From C of A2(X)	
			Params	Expected Result	Params	Expected Result
0	Waiting	-	-	-	-	-
1	-	B1 value	-	-	-	-
2	-	-	42	21	-	-
3	Ready	-	-	-	[63, 7]	[48, 12, 4]
28	-	-	-	-	-	-
56	Stop	-	-	-	-	-

Double Backend Design

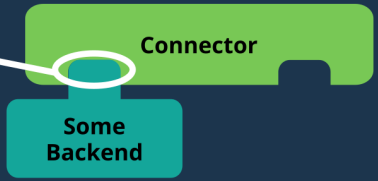
Double Backend Design

Double Backend Design

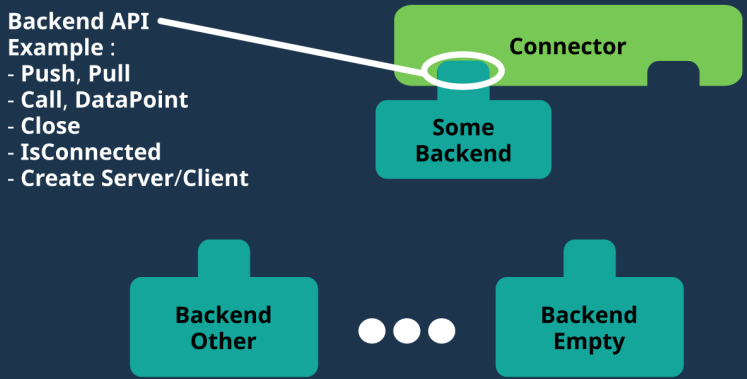


Double Backend Design

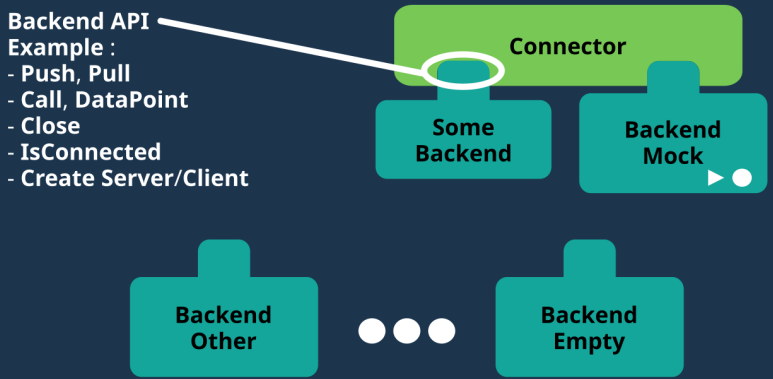
Backend API
Example :
- Push, Pull
- Call, DataPoint
- Close
- IsConnected
- Create Server/Client



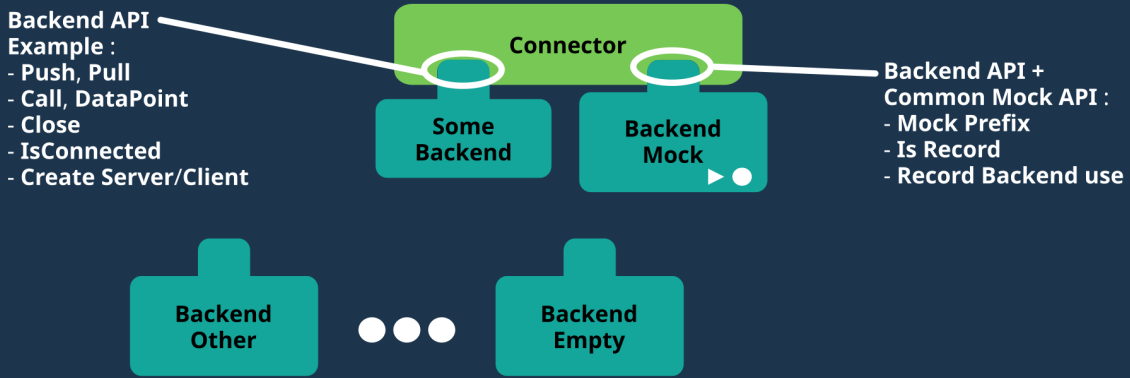
Double Backend Design



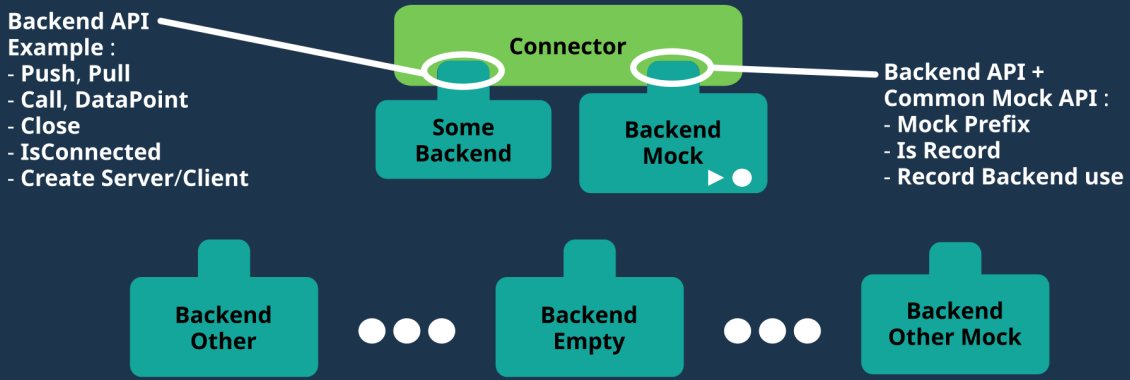
Double Backend Design



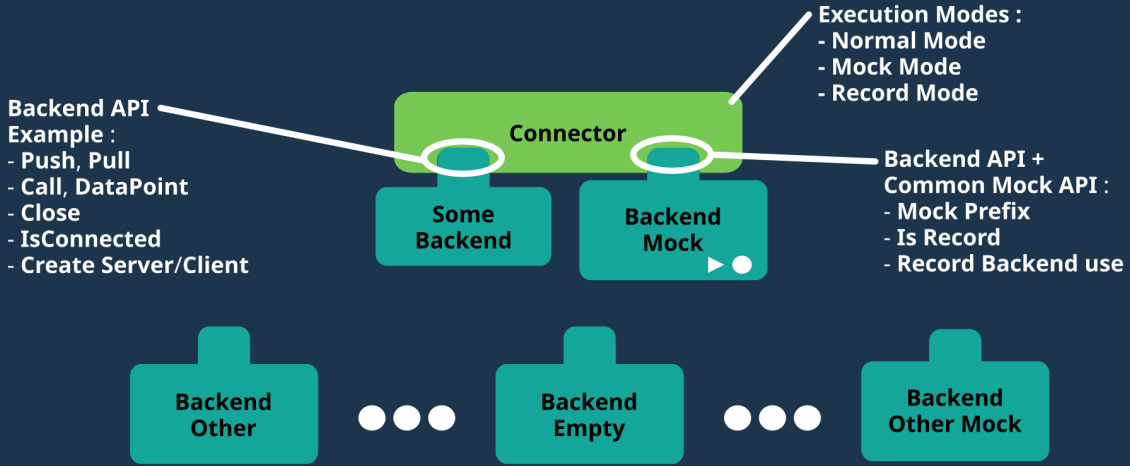
Double Backend Design



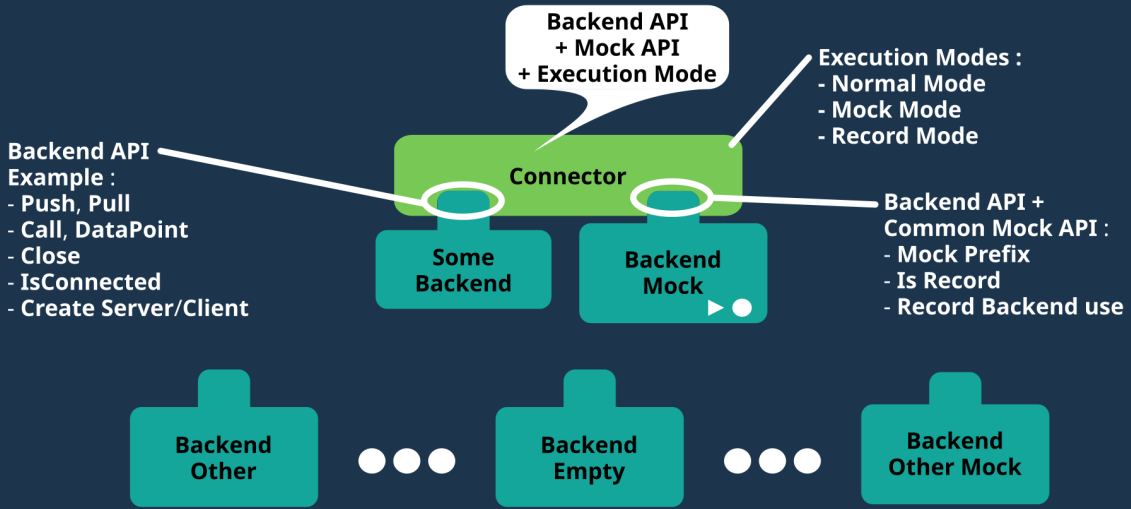
Double Backend Design



Double Backend Design



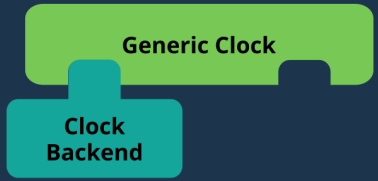
Double Backend Design



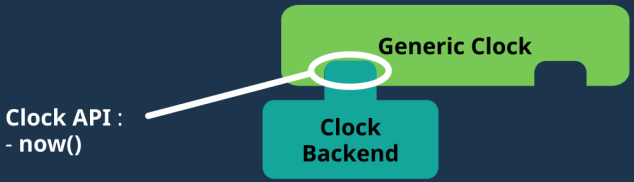
Double Backend Clock

Double Backend Clock

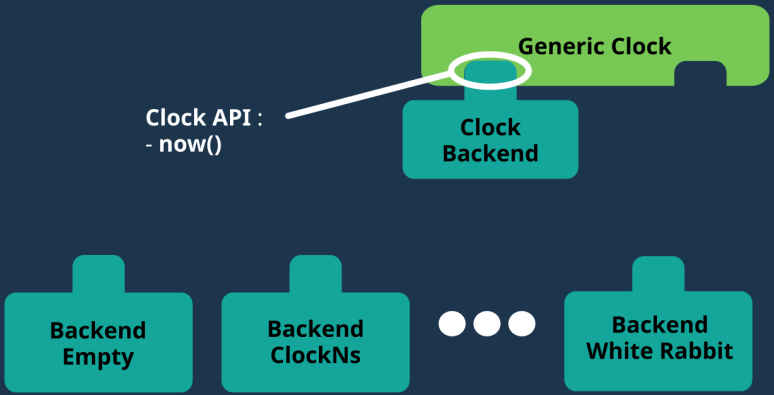
Double Backend Clock



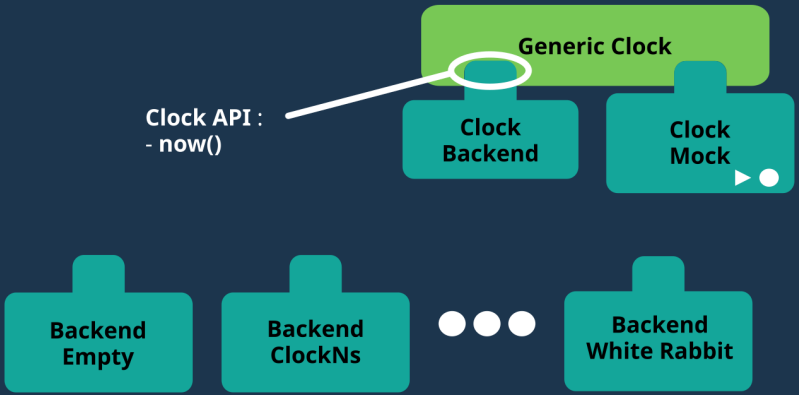
Double Backend Clock



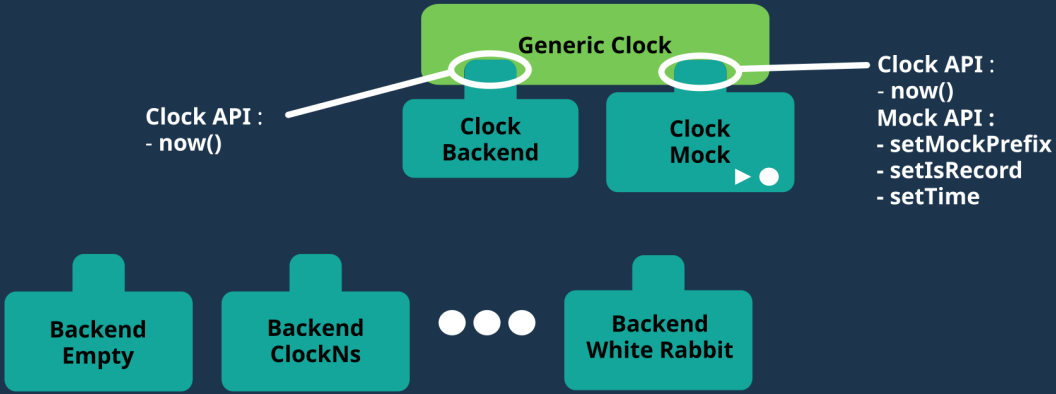
Double Backend Clock



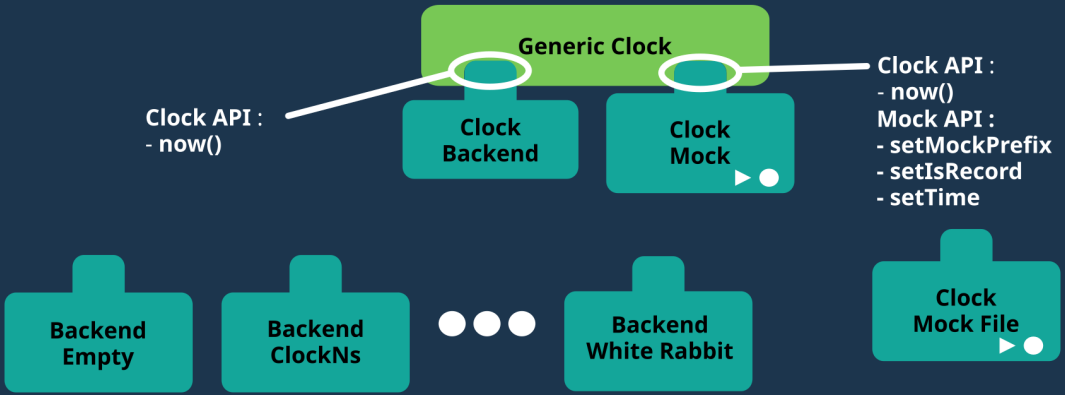
Double Backend Clock



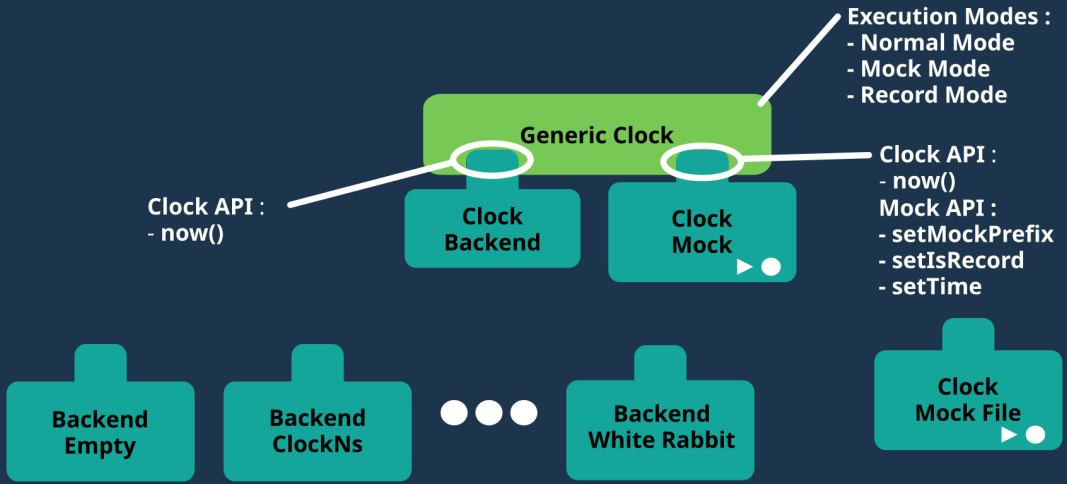
Double Backend Clock



Double Backend Clock



Double Backend Clock

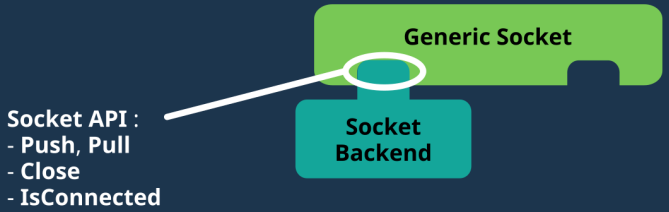


Double Backend Socket

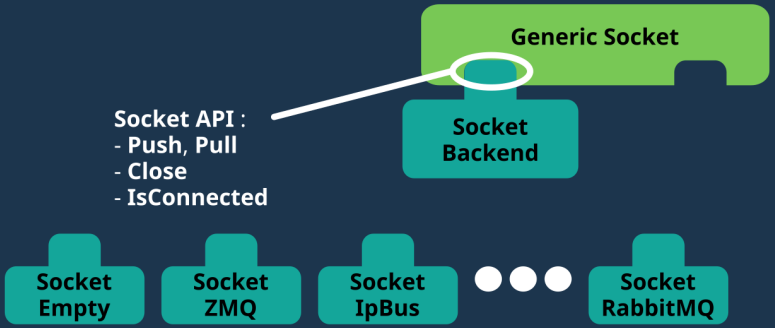
Double Backend Socket

Double Backend Socket

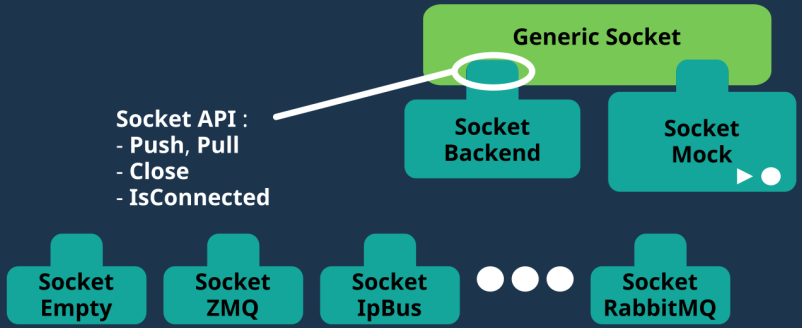
Double Backend Socket



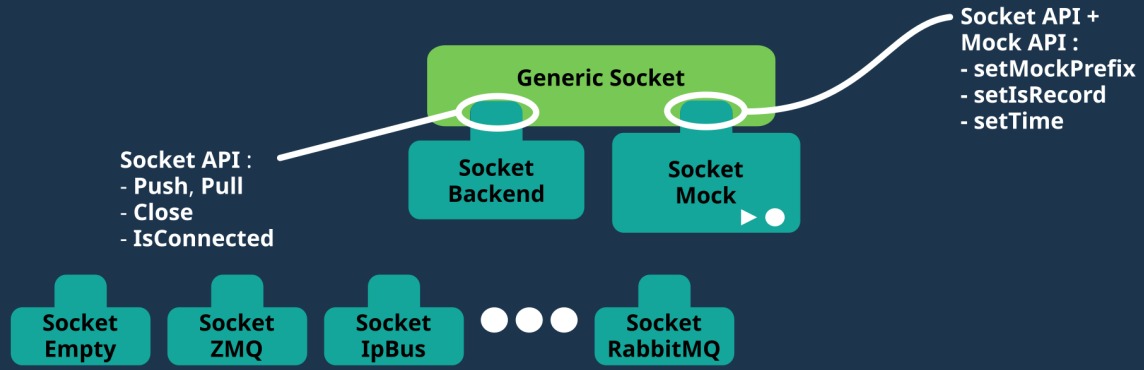
Double Backend Socket



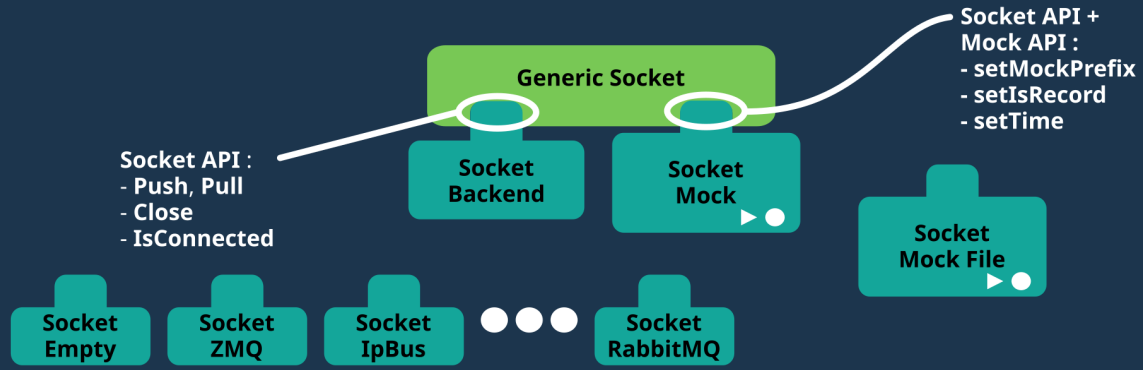
Double Backend Socket



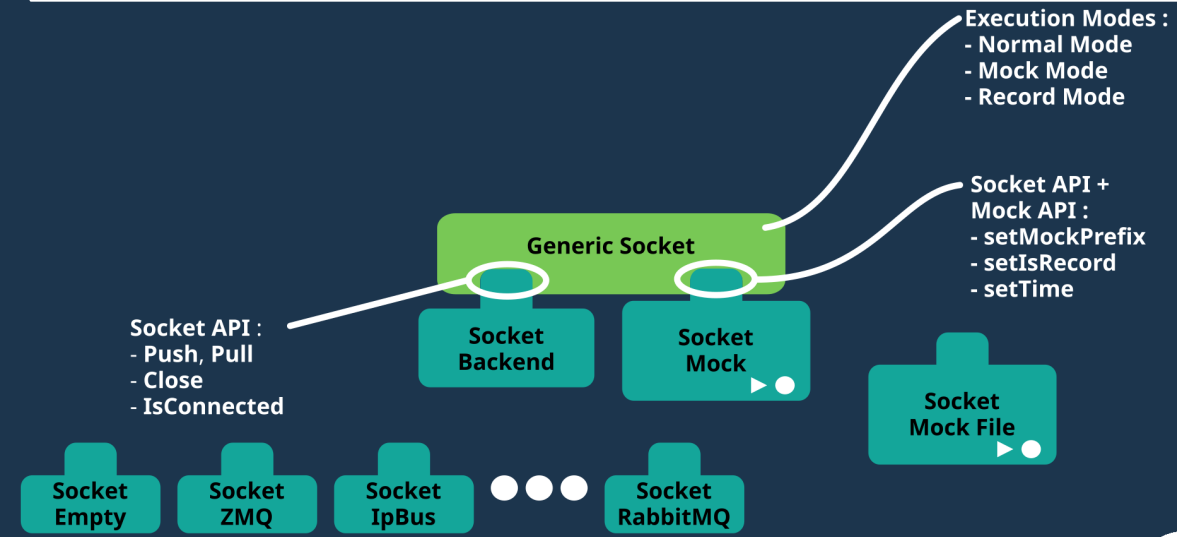
Double Backend Socket



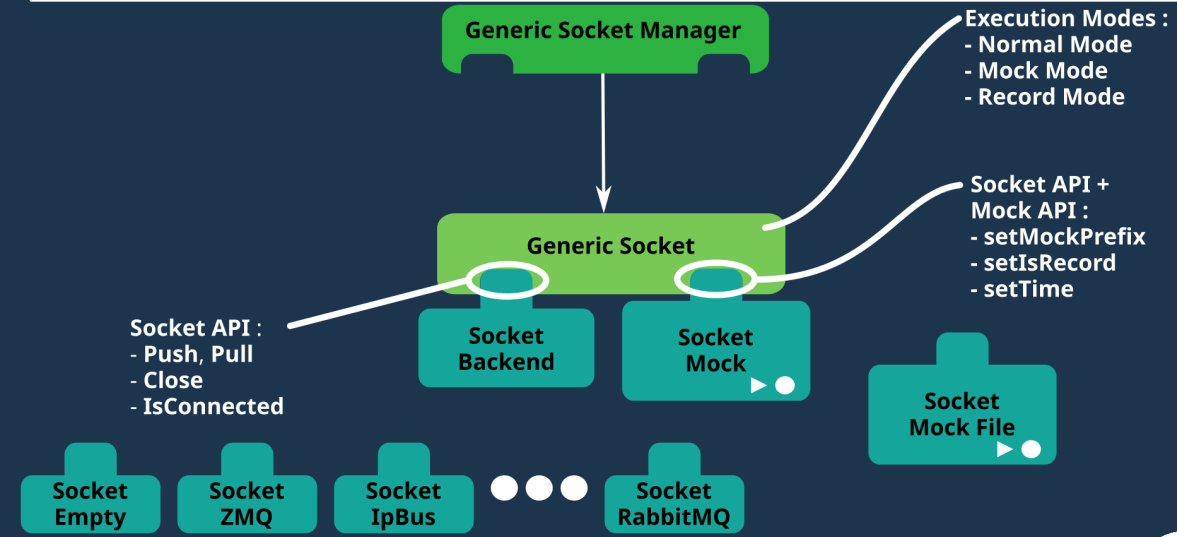
Double Backend Socket



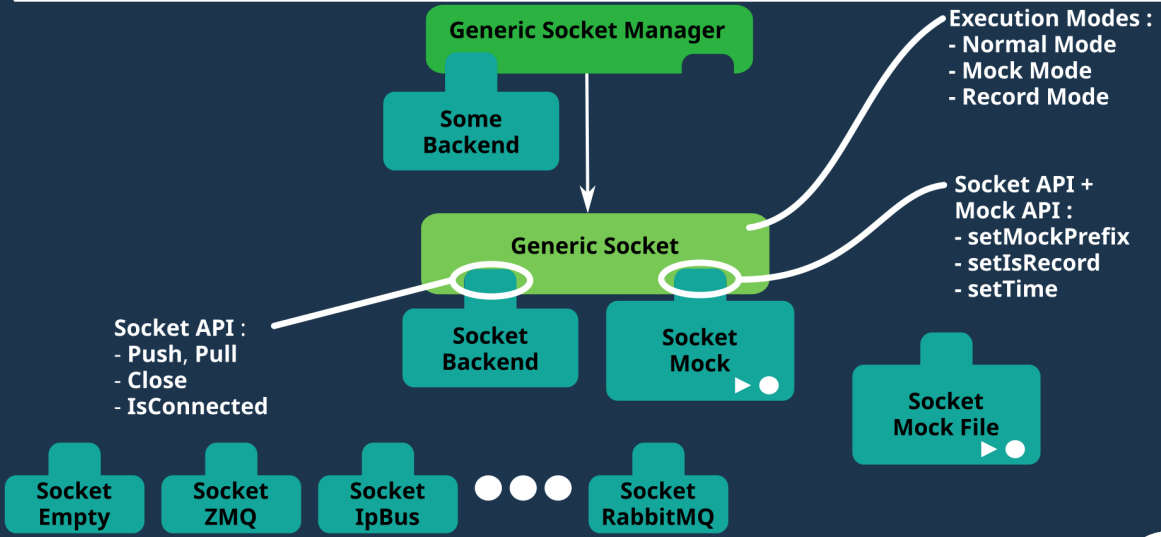
Double Backend Socket



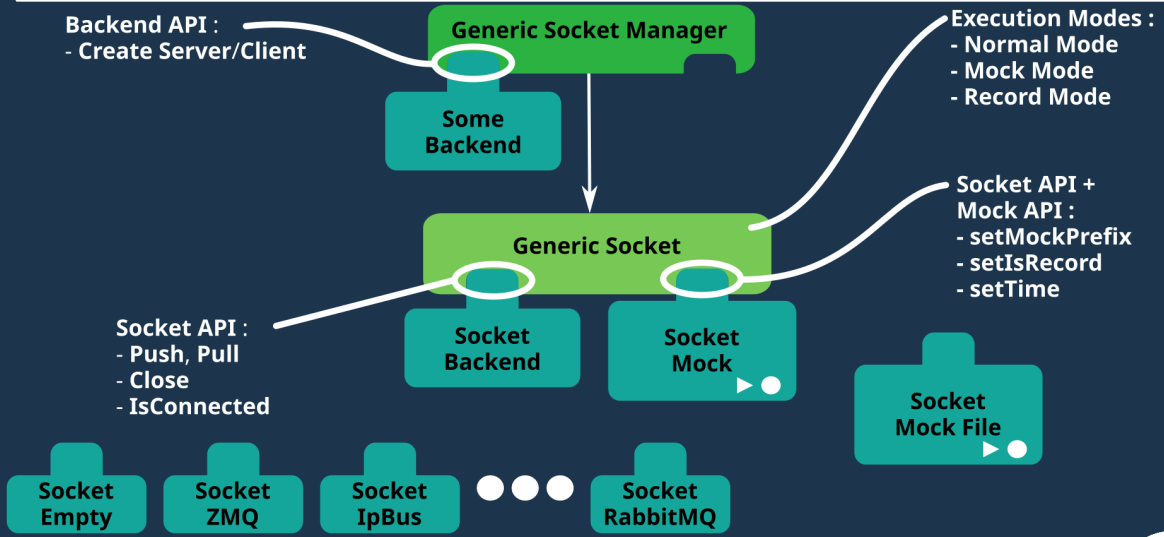
Double Backend Socket



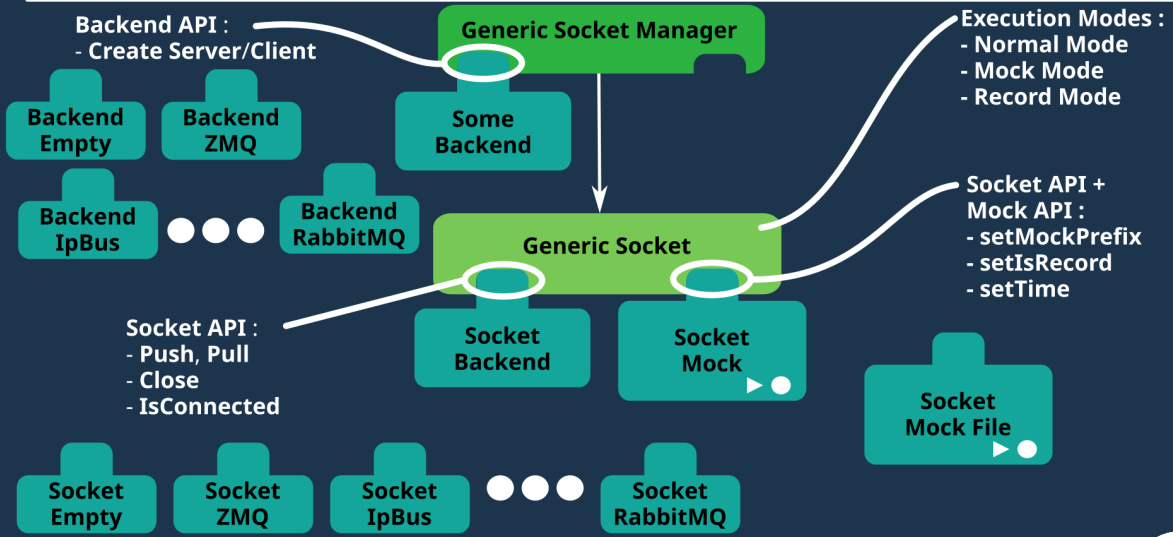
Double Backend Socket



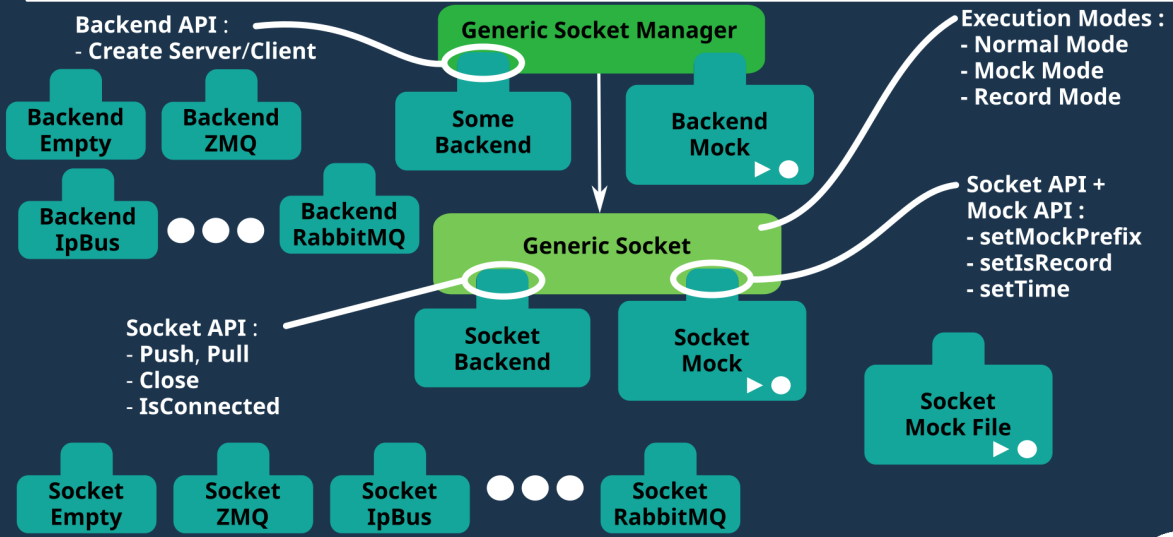
Double Backend Socket



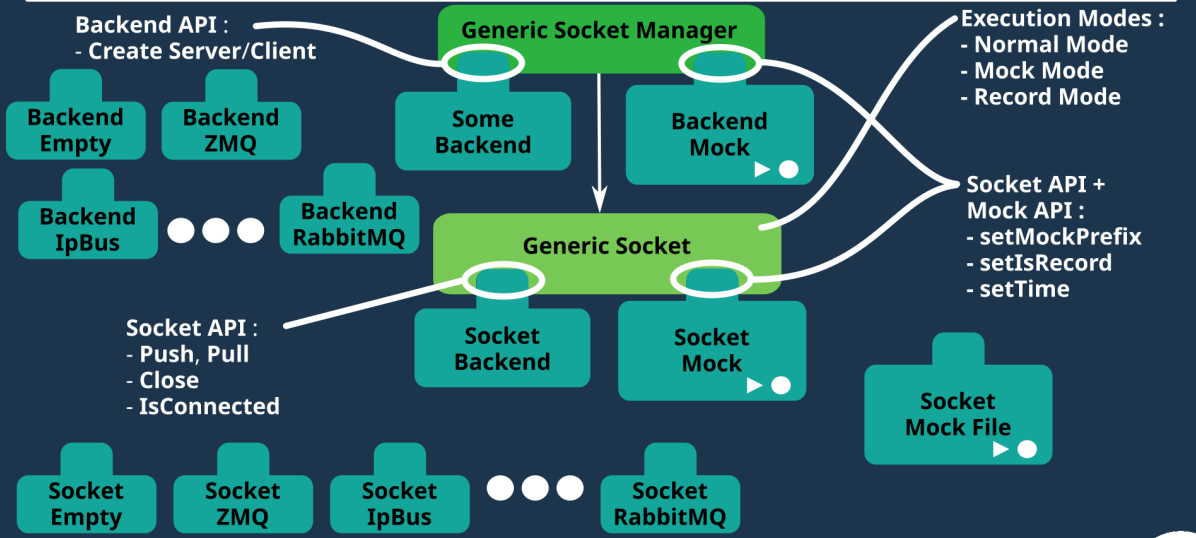
Double Backend Socket



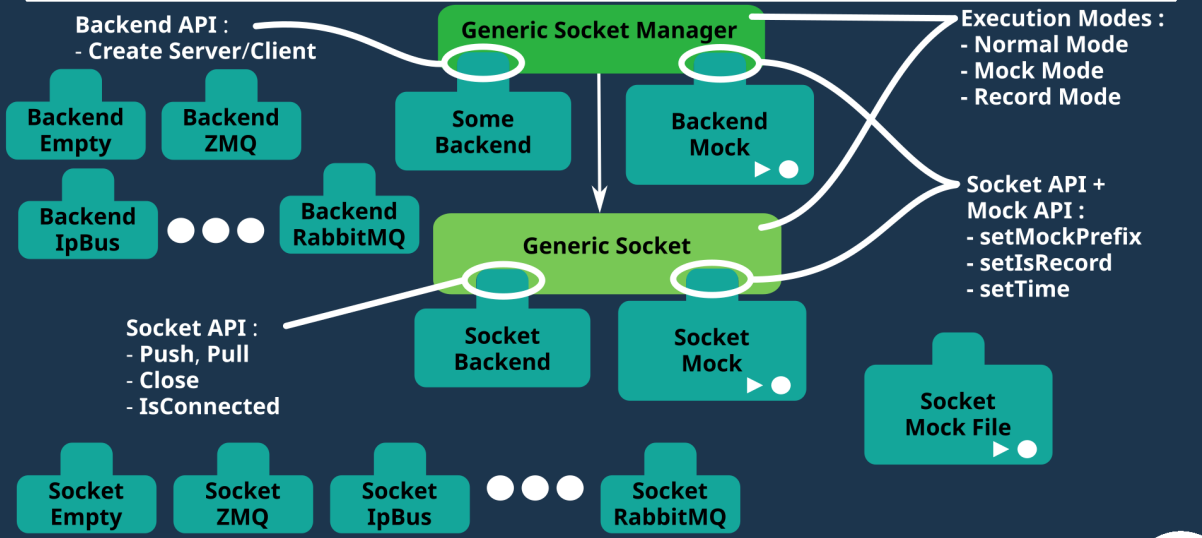
Double Backend Socket



Double Backend Socket

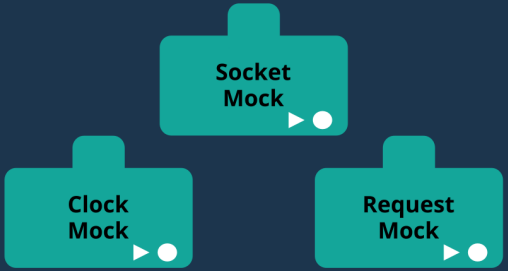


Double Backend Socket

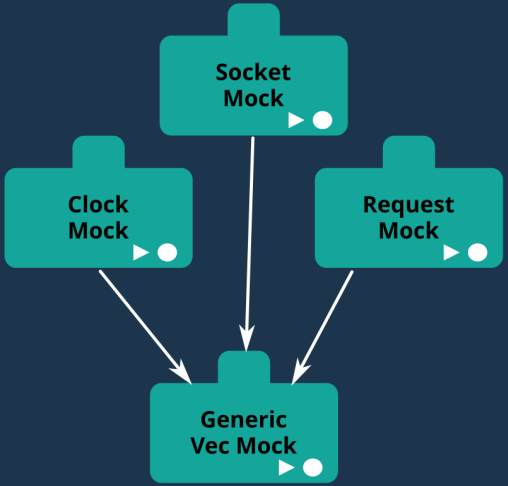


Generic Mock

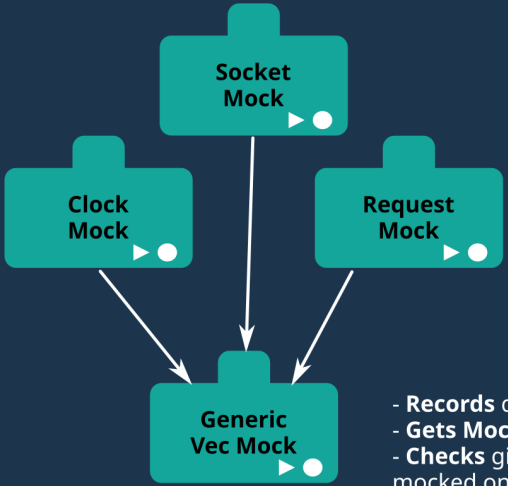
Generic Mock



Generic Mock

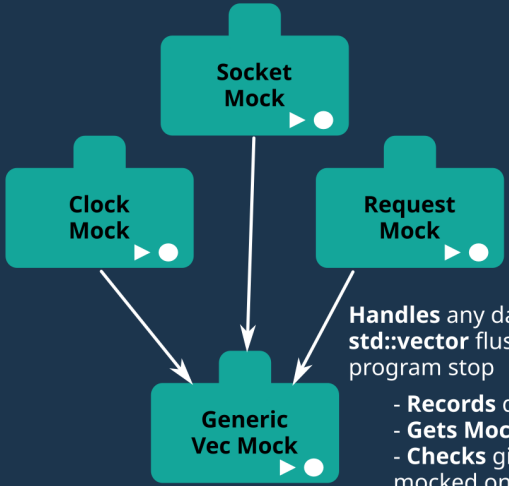


Generic Mock



- **Records** data
- **Gets Mocked** data
- **Checks** given **data** compare to mocked ones
- **Human readable feedback** if data are different

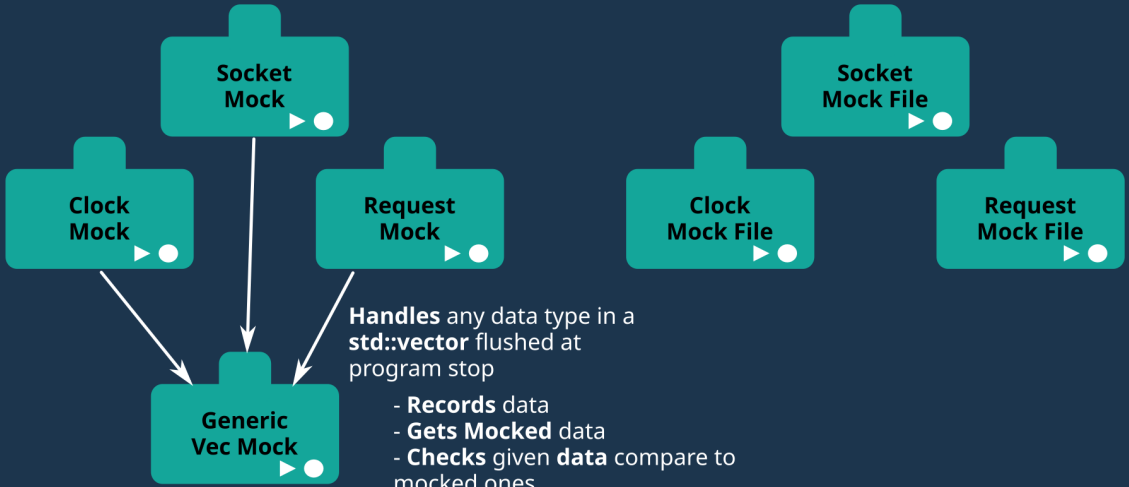
Generic Mock



Handles any data type in a **std::vector** flushed at program stop

- **Records** data
- **Gets Mocked** data
- **Checks** given **data** compare to mocked ones
- **Human readable feedback** if data are different

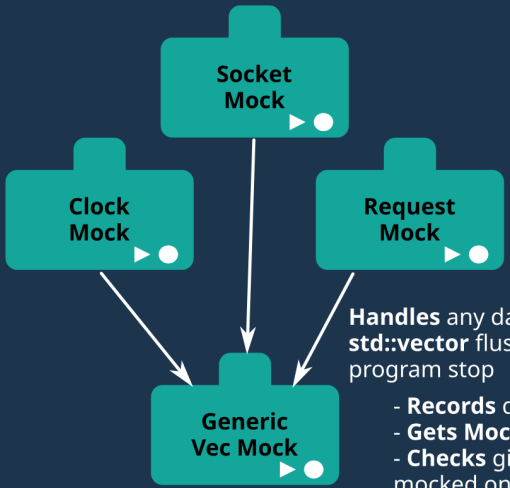
Generic Mock



Handles any data type in a **std::vector** flushed at program stop

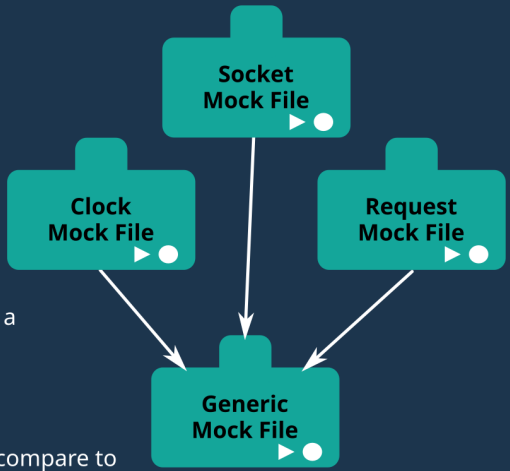
- **Records** data
- **Gets Mocked** data
- **Checks** given **data** compare to mocked ones
- **Human readable feedback** if data are different

Generic Mock

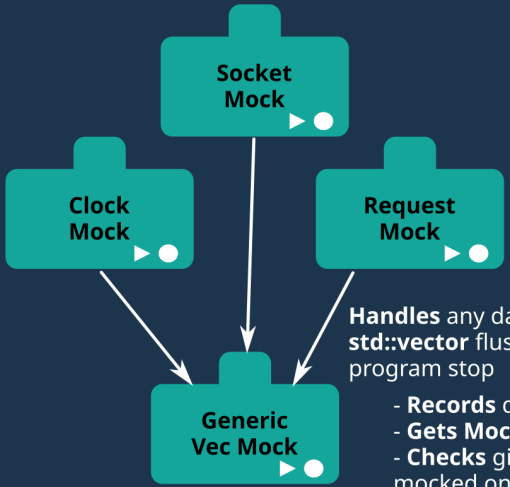


Handles any data type in a **std::vector** flushed at program stop

- **Records** data
- **Gets Mocked** data
- **Checks** given **data** compare to mocked ones
- **Human readable feedback** if data are different

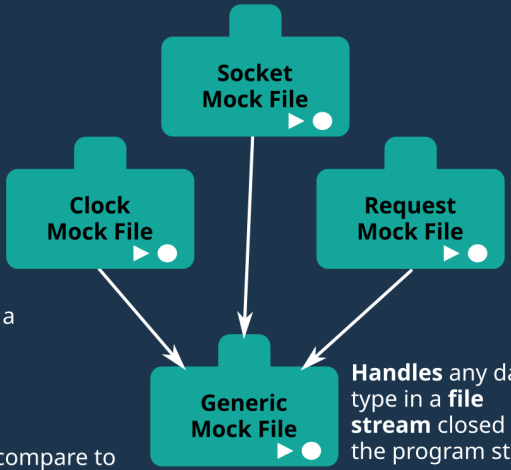


Generic Mock



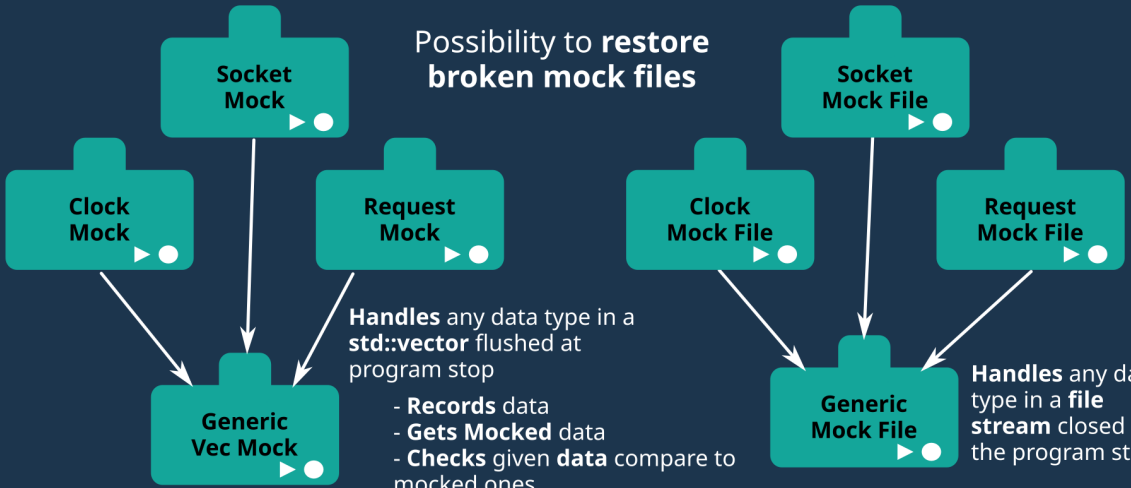
Handles any data type in a **std::vector** flushed at program stop

- **Records** data
- **Gets Mocked** data
- **Checks** given **data** compare to mocked ones
- **Human readable feedback** if data are different



Handles any data type in a **file stream** closed at the program stop

Generic Mock



Handles any data type in a **std::vector** flushed at program stop

- **Records** data
- **Gets Mocked** data
- **Checks** given **data** compare to mocked ones
- **Human readable feedback** if data are different

Handles any data type in a **file stream** closed at the program stop

Conclusion

Conclusion

Why use **mocks** ?

- **Run tests** on **laptop**, small **CI worker**
- **No latency**, **no sleeps**, **no timeouts**

Conclusion

Why use **mocks** ?

- **Run tests** on **laptop**, small **CI worker**
- **No latency**, **no sleeps**, **no timeouts**

Mock your **connection API**, not your connected servers :

- Implement **1 back-end**, mock **any** number of **server using it**
- Still running in **1 process** on your laptop

Conclusion

Why use **mocks** ?

- **Run tests** on **laptop**, small **CI worker**
- **No latency**, **no sleeps**, **no timeouts**

Mock your **connection API**, not your connected servers :

- Implement **1 back-end**, mock **any** number of **server using it**
- Still running in **1 process** on your laptop

Use a **mock back-end** and a **real back-end at the same time** :

- **Create mock** files **automatically** with **record mode**
- **Develop** your application with the **mock back-end**, **choose** your real **back-end later**!

Conclusion

Why use **mocks** ?

- **Run tests** on **laptop**, small **CI worker**
- **No latency**, **no sleeps**, **no timeouts**

Mock your **connection API**, not your connected servers :

- Implement **1 back-end**, mock **any** number of **server using it**
- Still running in **1 process** on your laptop

Use a **mock back-end** and a **real back-end at the same time** :

- **Create mock** files **automatically** with **record mode**
- **Develop** your application with the **mock back-end**, **choose** your real **back-end later**!

What is **available now** (C++, Rust, and Python soon) :

- C++ Project : `https://gitlab.in2p3.fr/CTA-LAPP/PHOENIX_LIBS2/serialize-io`
- C++ Project : `https://gitlab.in2p3.fr/CTA-LAPP/PHOENIX_LIBS2/network`
- C++ Project : `https://gitlab.in2p3.fr/CTA-LAPP/PHOENIX_LIBS2/clock`