

Polyglot Software

Quels sont les outils et méthodes supports au cycle de vie des logiciels polyglottes ?



Yann Paillard, Ronan Tremoureux



Plan

01

**Polyglot
software ?**

02

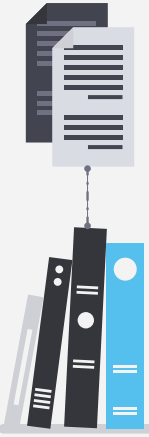
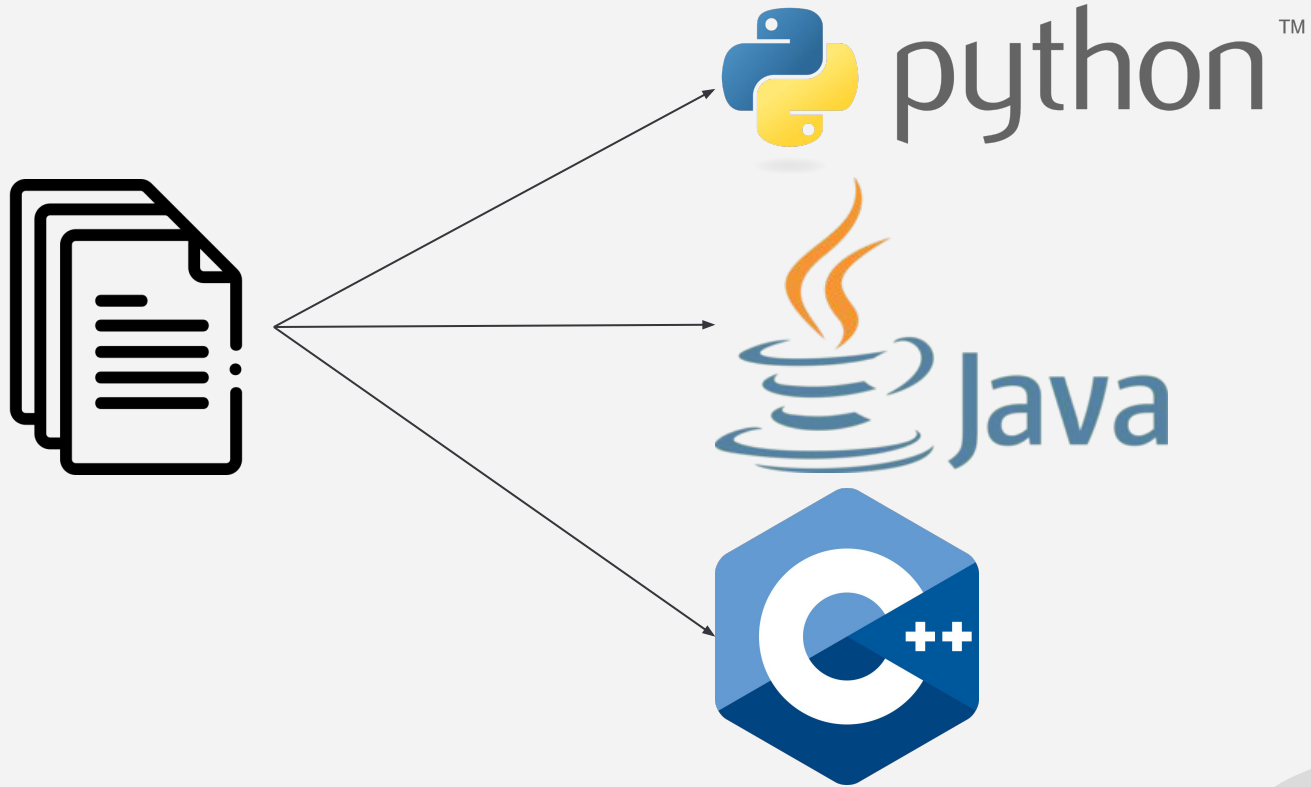
**Quelques
pratiques**

03

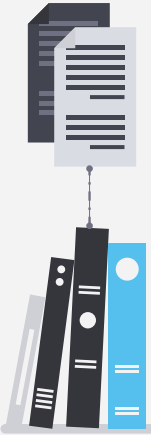
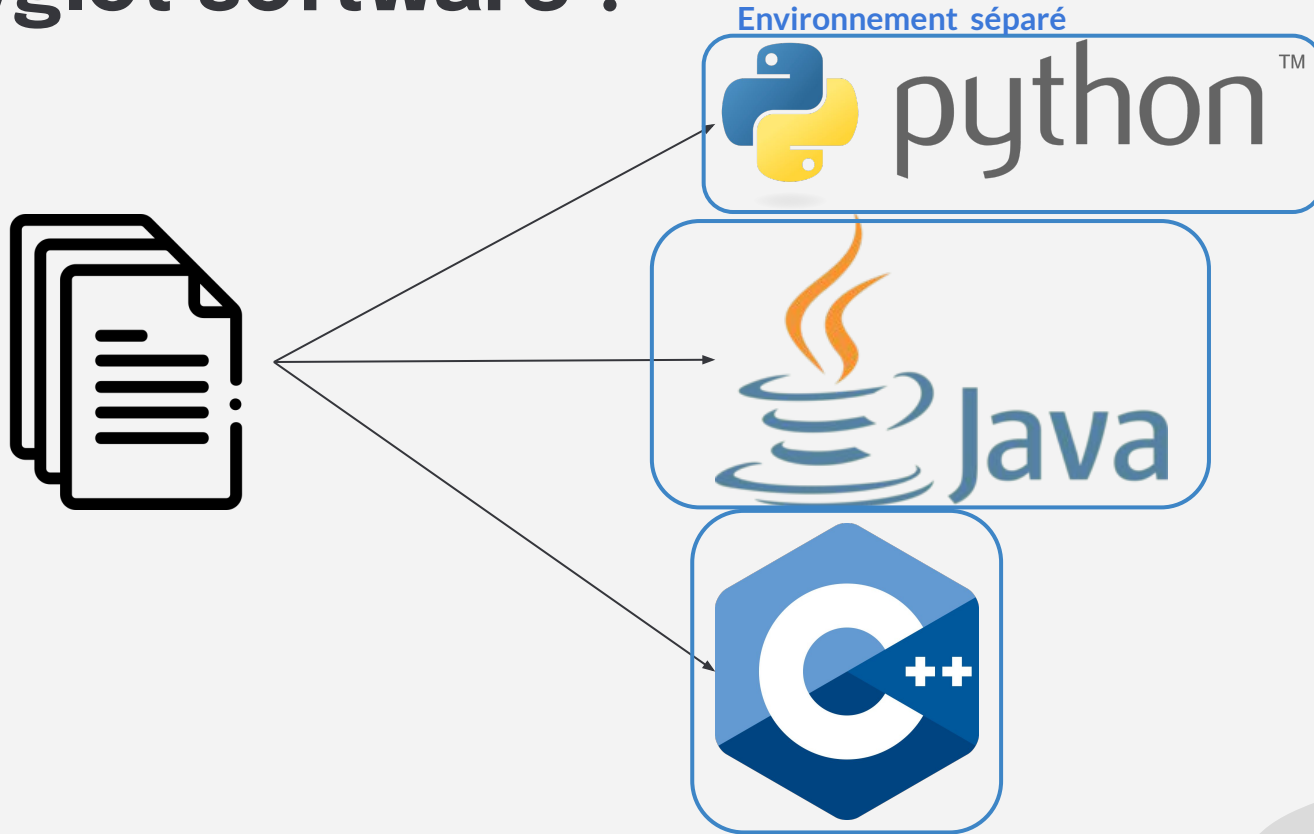
**Le positif et le
négatif de la
pratique**



Polyglot software ?

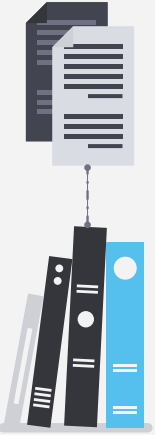
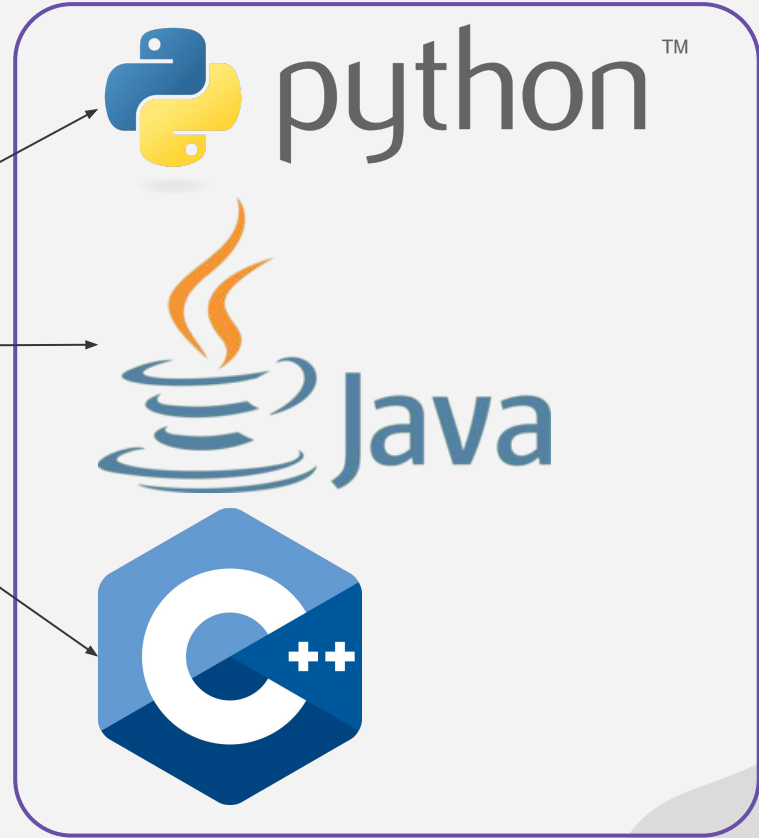
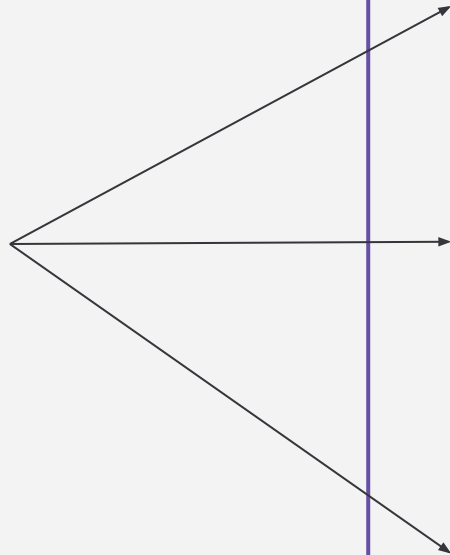


Polyglot software ?



Polyglot software ?

Environnement partagé



Polyglot software ?

Pourquoi ?

01

Plus de logique
métier



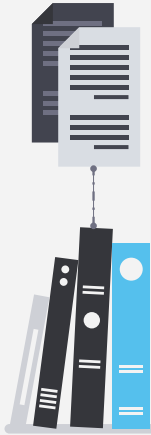
02

Réutilisation
de code

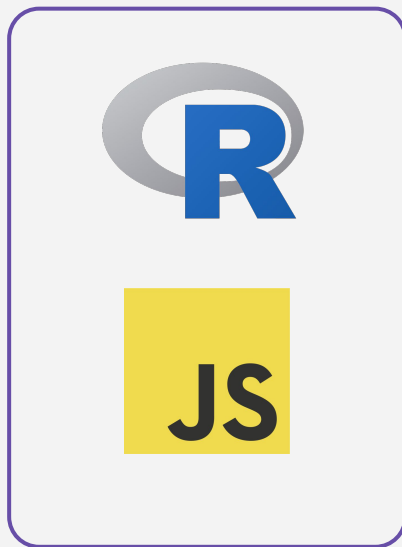
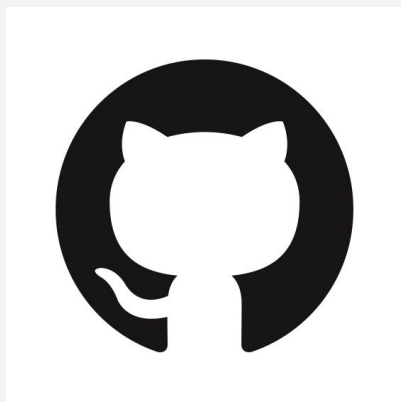


03

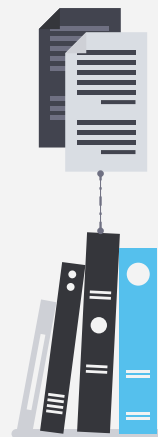
Expertise différente
des développeurs



Exemples d'utilisation



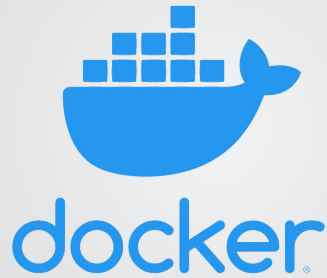
Environnement partagé



<https://github.com/graalvm/graalvm-demos/tree/master/polyglot-javascript-java-r1>

Quelques outils

GraalVM™



.NET



GraalVM™

- **Machine Virtuelle**
- **beaucoup de langages supportés**
- **Création de DSL avec truffle**

Example Applications

Here you will find example applications written in Java, JavaScript, R, Ruby, and other JVM languages to illustrate the diverse capabilities of GraalVM.

The GraalVM compiler, enabled by default in GraalVM, assures performance advantages for highly abstracted programs due to its versatile optimization techniques. Code using more abstraction and modern Java features like Streams or Lambdas will see even greater speedups. [Java Performance Examples](#) and [Java stream API Benchmark](#) demonstrate this.

The [Polyglot JavaScript, Java, R Example Application](#) displays GraalVM's abilities as a polyglot runtime, processing programs written in two or more languages.





docker

- Environnement séparé
- Déploiement facile
- Très utilisé : web, microservices...



```
1  version: '3.8'
2  services:
3    backend:
4      build:
5        context: ./path-to-your-backend
6        dockerfile: Dockerfile
7      ports:
8        - "8080:8080"
9
10   frontend:
11     build:
12       context: ./path-to-your-frontend
13       dockerfile: Dockerfile
14     ports:
15       - "3000:80"
16     depends_on:
17       - backend
```



Polynote

- Multi Lingual Notebook

- Basé sur  python™

- Projet supporté par Netflix

```
In(1): Scala {} ☰ Ran on 3/24/2021, 12:46:04 PM PDT took 658ms  
1 case class Point(x: Double, y: Double, color: String)  
2 val colors = Seq("red", "green", "blue")  
3 val n = 100  
4 val idiomaticData = for {  
5     ...  
6     n ← (0 to n)  
7 } yield Point(Math.random(), Math.random(), color)
```

Now, we'll do some data preparation to get the Scala data in a format we can plot with `matplotlib`.

```
In(3): Python {} ☰ Ran on 3/24/2021, 12:53:04 PM PDT took 235ms  
1 data = []  
2 for idx in range(idiomaticData.length()):  
3     point = idiomaticData.apply(idx)  
4     data.append([point.x(), point.y(), point.color()])  
5  
6 data  
  
Out:  
  
[[0.8768817563655382, 0.6024507503626798, 'red'], [0.9565236520716693, 0.32427813654323145, 'red'], [0.7466390903321652, 0
```



.NET

Interactive

- Multi Lingual Notebook
- microsoft
- Extension visual studio



Language	Variable sharing
C#	✓
F#	✓
PowerShell	✓
JavaScript	✓
SQL	✓
KQL (Kusto Query Language)	✓
Python	✓
R	✓
HTML	✗
Mermaid	✗

Les avantages et les inconvénients

Avantages

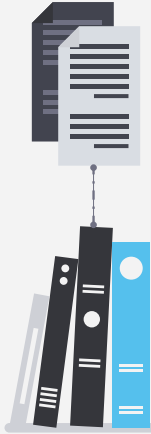
Plein de DSL

Inconvénients

Code smells

Sécurité

1. Passing Excessive Objects
2. Unnecessary Parameters
3. Not Caching Objects' Elements
4. Memory Management Mismatch



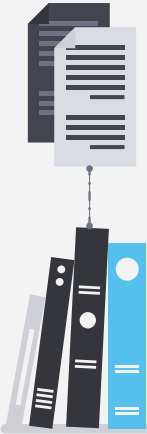
Conclusion

Réutilisation de code

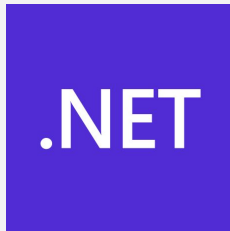
Communication = Problème de sécurité

Utilisé à grande échelle

Séparation en métier



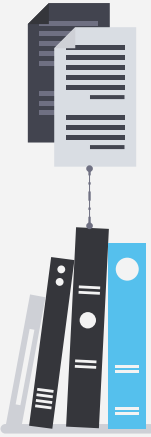
Démonstration | Environnement partagés



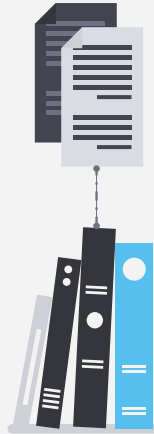
Interactive



Polynote



Bibliographie



1. [Gunter Mussbacher](#); [Benoit Combemale](#); [Jörg Kienzle](#); [Lola Burqueño](#); [Antonio Garcia-Dominguez](#); [Jean-Marc Jézéquel](#); [Gwendal Jouneaux](#); [Djamel-Eddine Khelladi](#); [Sébastien Mosser](#); [Corinne Pulgar](#); [Houari Sahraoui](#); [Maximilian Schiedermeier](#); [Tijs van der Storm](#) , IEEE, “Polyglot Software Development: Wait, What?”, 2024, [10.1109/MS.2023.3347875](#)
2. [Mouna Abidi](#); [Manel Grichi](#); [Foutse Khomh](#); [Yann-Gaël Guéhéneuc](#), ACM, “Code Smells for Multi-language Systems”, 2019, [10.1145/3361149.3361161](#)
3. [Michael Van De Vanter](#); [Chris Seaton](#); [Michael Haupt](#); [Christian Humer](#); [Thomas Würthinger](#), ARVI, “Fast, Flexible, Polyglot Instrumentation Support for Debuggers and other Tools”, 2018, [10.22152/programming-journal.org/2018/2/14](#)
4. [Vamsi Krishna Yepuri](#); [Venkata Kalyan Polamarasetty](#); [Shivani Donthi](#); [Ajay Kumar Reddy Gondj](#); “Containerization of a polyglot microservice application using Docker and Kubernetes”, 2023, [arXiv:2305.00600](#)

