Plateforme de compilation et d'exécution à distance, sécurisée et basée sur Docker

Réalisé par :

- Salmane Bah
- Timothée Sollaud
- Tristan Braquelaire
- Wassim Romdan

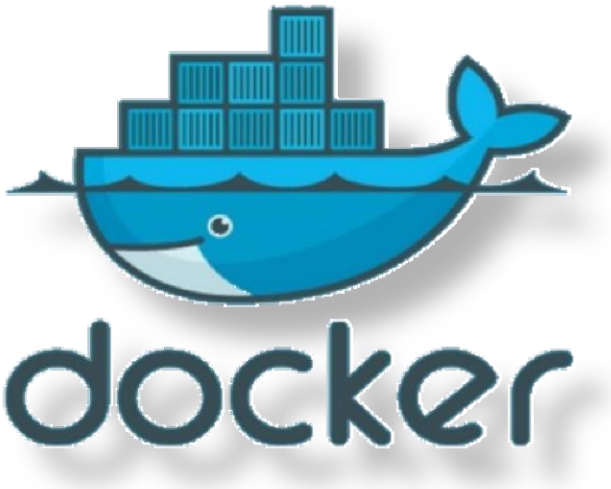
université de BORDEAUX

Client :

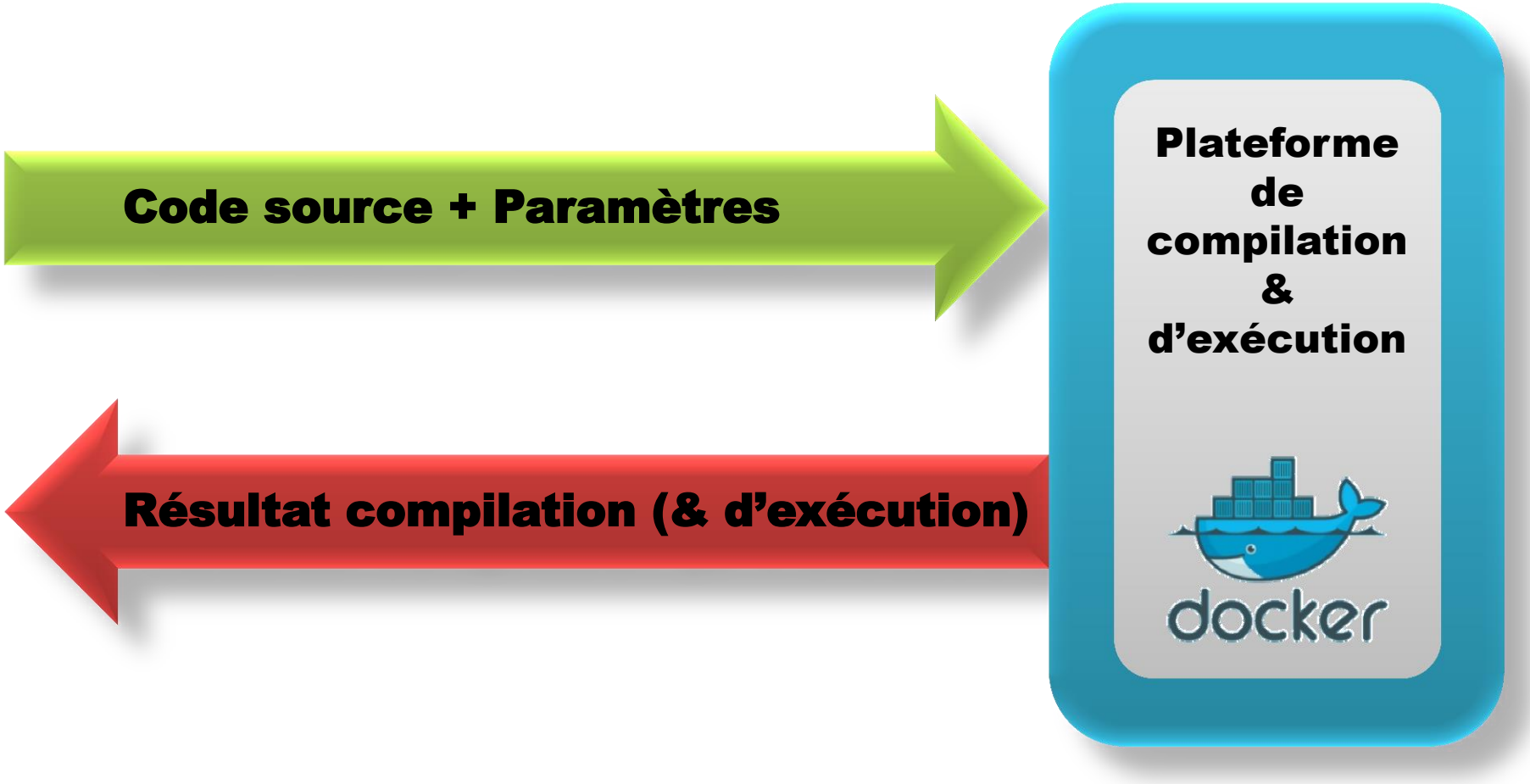
- M. Blin Guillaume

Chargé de TD :

- M. Blanc Xavier

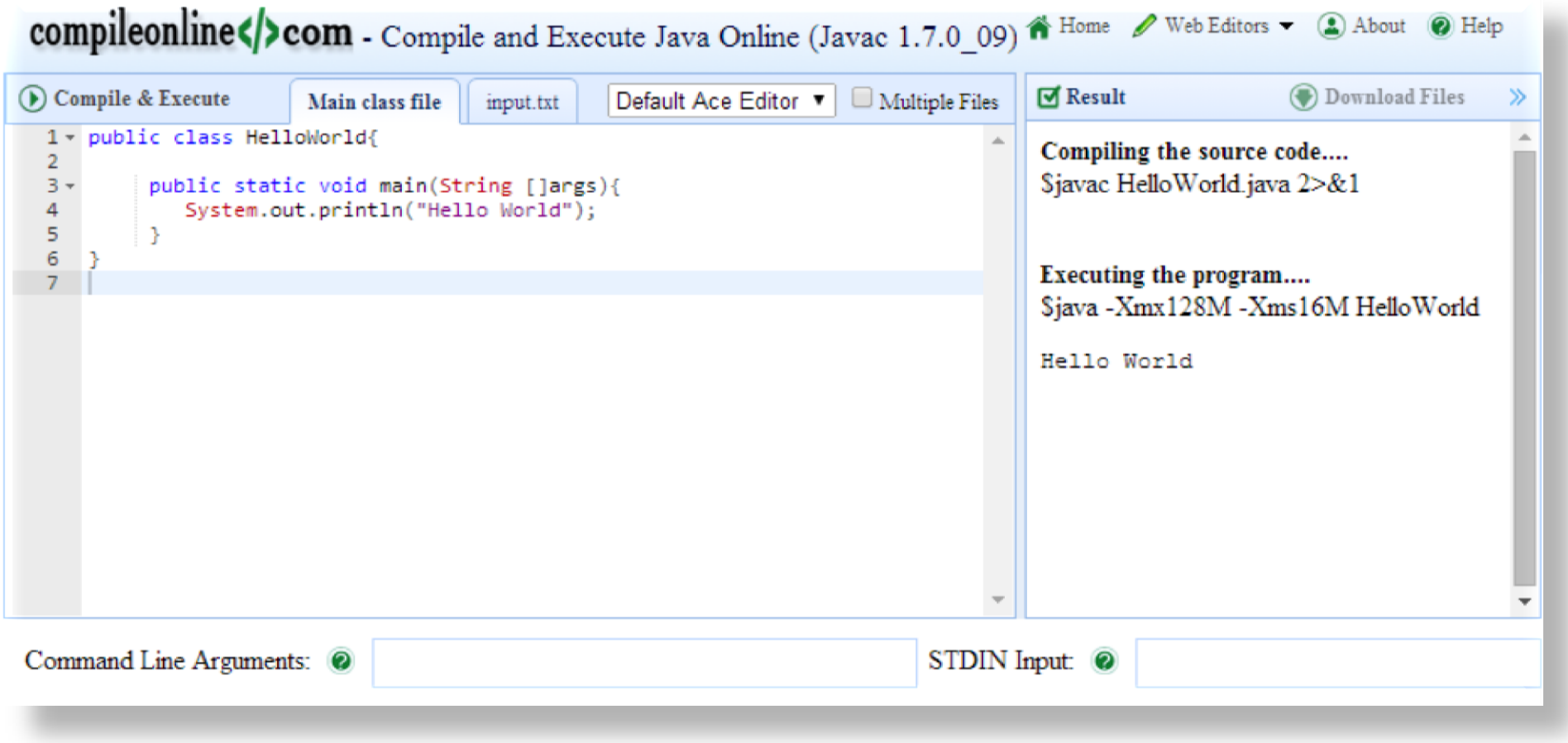


Description du sujet

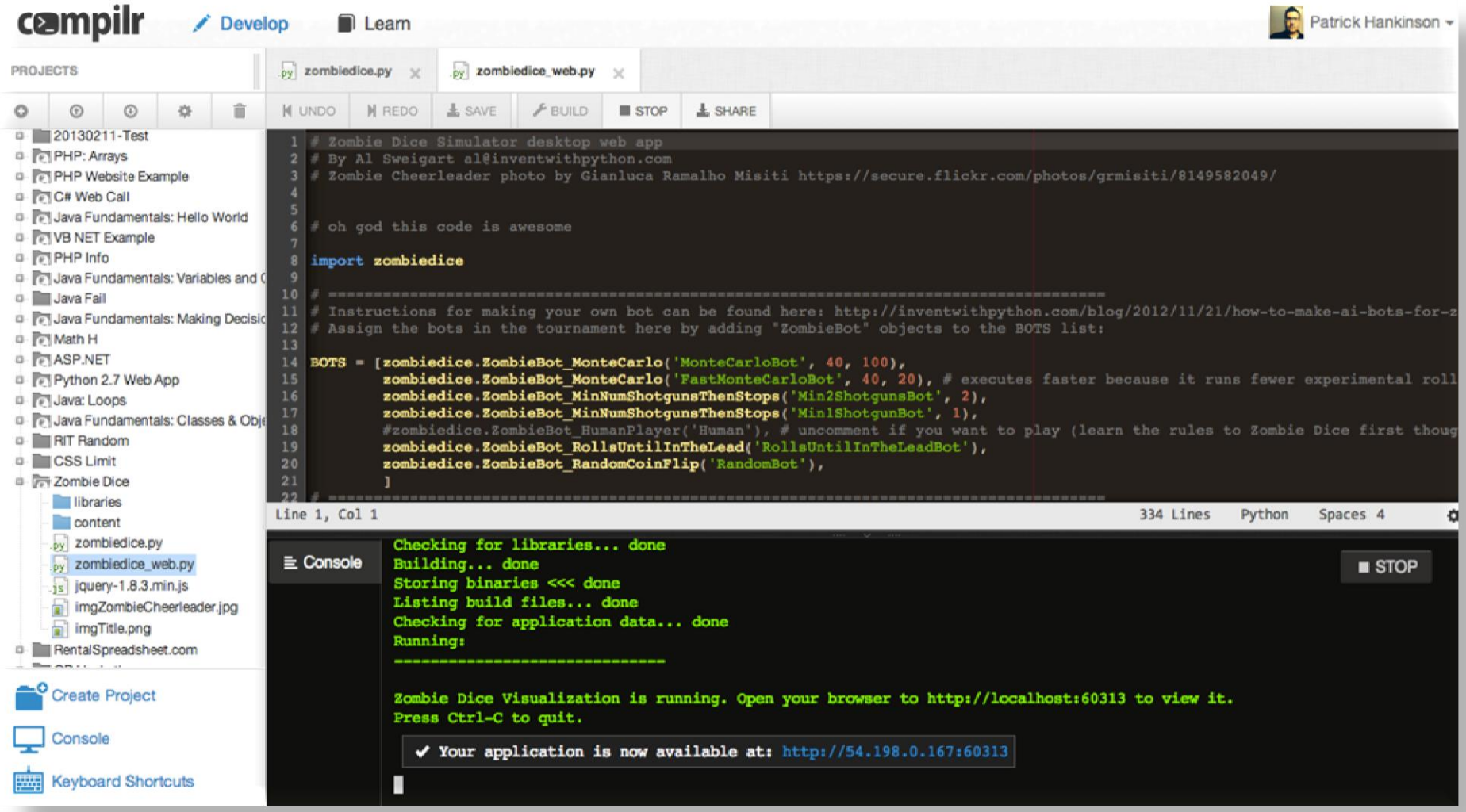


Etude de l'existant : *Ideone.com*

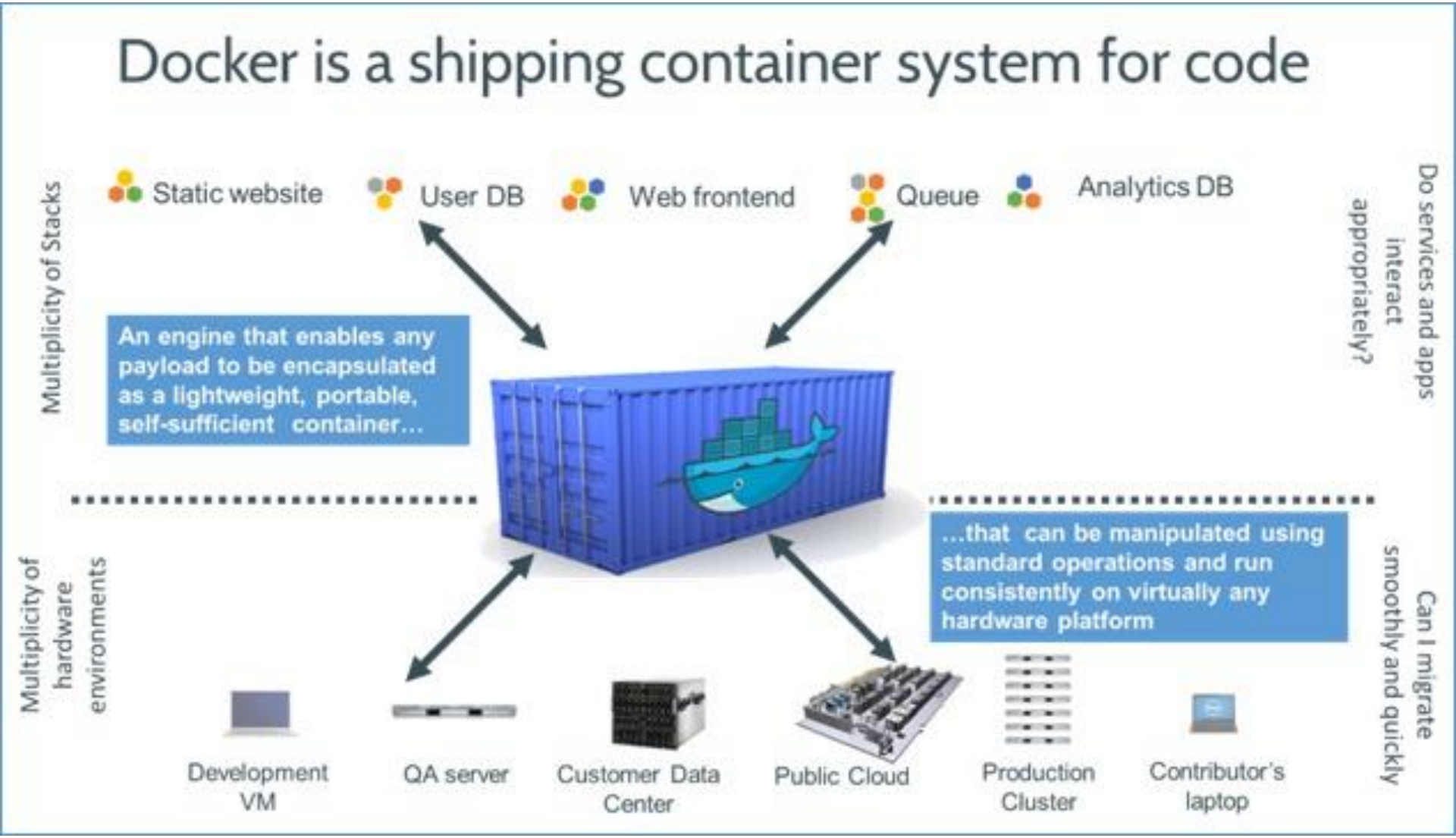
Etude de l'existant : *compileonline.com*



Etude de l'existant : *compilr.com*



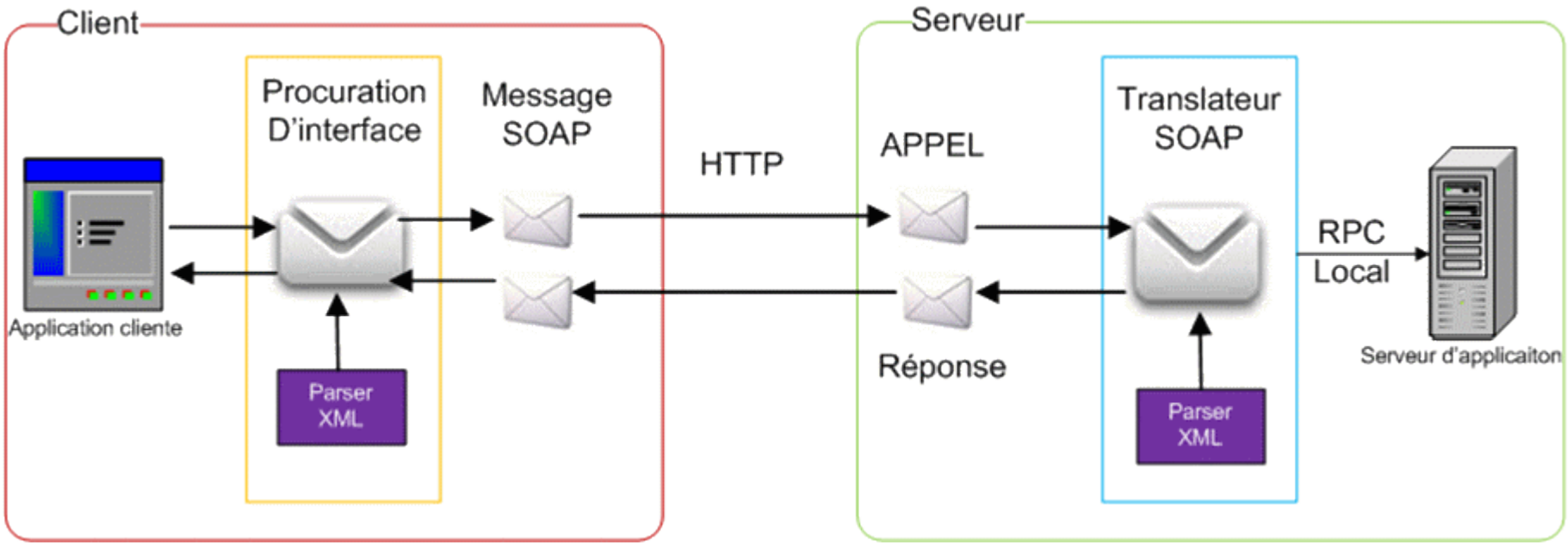
Docker & la sécurité



Docker & la sécurité

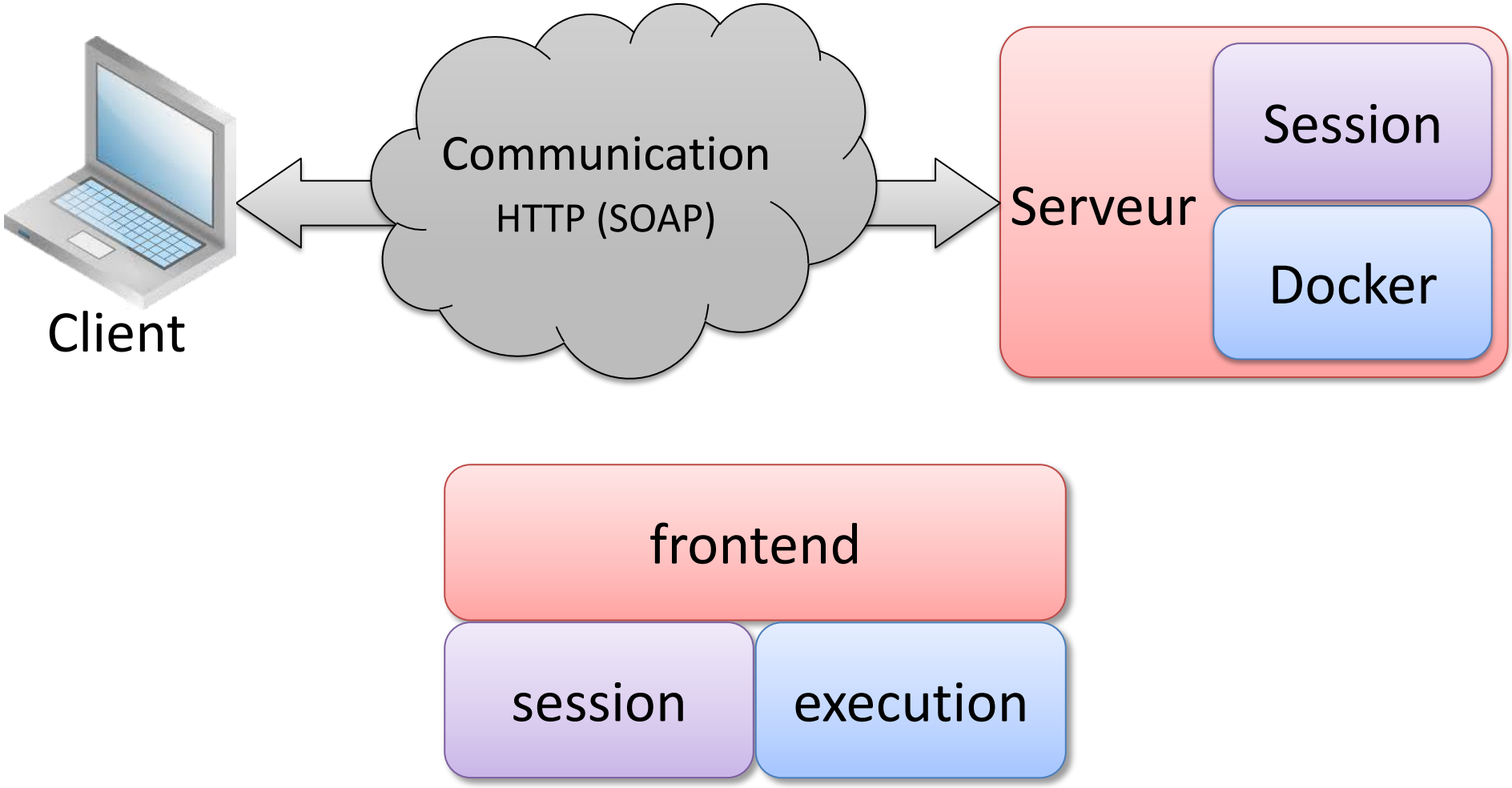
- --user=<*username*>
- --networking=<*true*|*false*>
- -m <*space*>
- --volume <*path*>
- timeout *(fonction Linux)*

Un Service Web (WS)



- XML & HTTP
- Indépendant du système
- Indépendant du langage
- Beaucoup de bibliothèques

Architecture



Architecture : *frontend*

RequestListener

RequestHandler

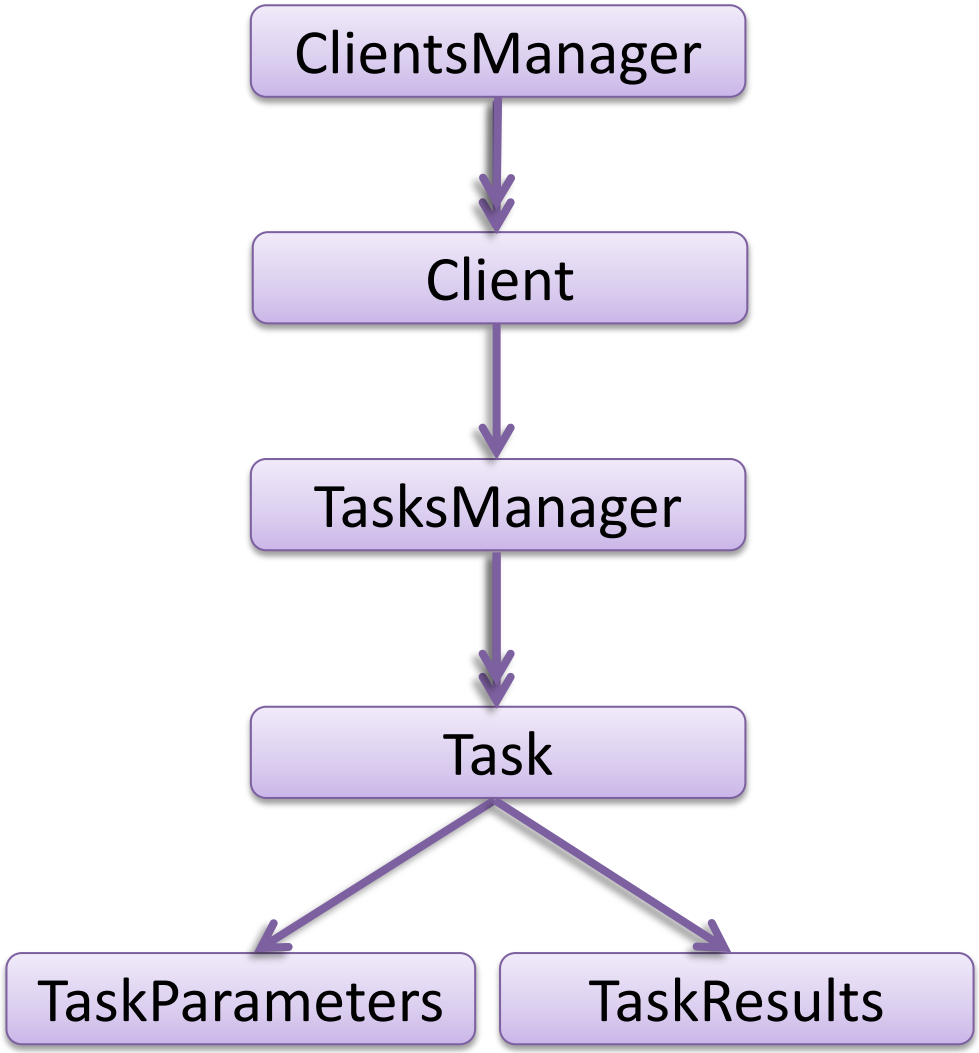
```
<message name="connect"/>
<message name="connectResponse">
  <part name="connectResult" type="tns:connectRet"/>
</message>
<message name="disconnect">
  <part name="sessionKey" type="xsd:string"/>
</message>
<message name="disconnectResponse">
  <part name="disconnectResult" type="tns:infoKey"/>
</message>
<message name="getLanguages"/>
<message name="getLanguagesResponse">
  <part name="getLanguagesResult" type="tns:getLanguagesRet"/>
</message>
```

```
@WebService(name="CompilePlatform" ,
            targetNamespace="http://pdp.compileplatform.frontend")
@SOAPBinding(style=Style.RPC)
public interface RequestListenerInterface {
    * web method to connect on the server in order to compile, interpret or
    @WebMethod
    @WebResult(partName="connectResult")
    ConnectRet connect();

    * web method to disconnect from the server , all client specific data a
    @WebMethod
    @WebResult(partName="disconnectResult")
    InfoKey disconnect(@WebParam(partName="sessionKey") String sessionKey);

    * web method to retrieve languages supported by the platform , does not
    @WebMethod
    @WebResult(partName="getLanguagesResult")
    GetLanguagesRet getLanguages();
```

Architecture : *session*

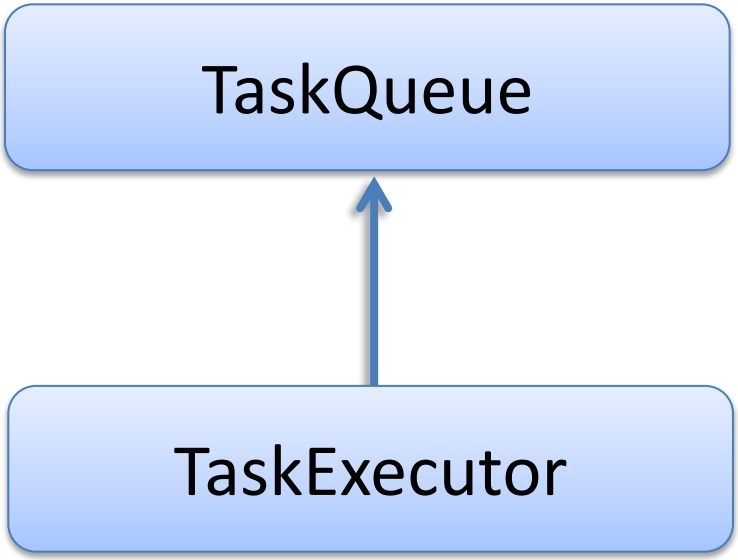


Architecture : *session, le Cleaner*

```
/**
 * Clean directories for inactive sessions and tasks
 */
private void sessionCleaner() {
    Timer timer = new Timer();
    timer.schedule(new TimerTask() {
        @Override
        public void run() {
            RequestHandler.printLog("/!\\ cleanning session(s)");
            List<ClientInterface> lstOldClients = new ArrayList<ClientInterface>();
            for(Entry<String, ClientInterface> entry : mClients.entrySet()) {
                ClientInterface client = entry.getValue();
                if (client.inactiveSince() > maxClientInactivityDuration){
                    lstOldClients.add(client);
                }
            }
            for (ClientInterface client : lstOldClients) {
                ClientsManager.this.removeClient(client);
            }

            RequestHandler.printLog("/!\\ cleanning task(s)");
            for(Entry<String, ClientInterface> entry : mClients.entrySet()) {
                ClientInterface client = entry.getValue();
                client.cleanTasks(maxTaskInactivityDuration);
            }
        }
    }, sessionCleanerTimer, sessionCleanerTimer);
}
```

Architecture : *execution*



Architecture : *exécution, les langages*

```
public LangInternal(int idLang,
                    String langName,
                    String compiler,
                    boolean interpreted,
                    String extension,
                    String compileCommand,
                    String executionCommand)
{
    super(idLang, langName, compiler);
```

```
/**
 * construct a well formed compilation command
 * @param fileName the file name to use
 * @param compileOptions compilation options
 * @return a well formed compilation command depending on this language
 * (e.g {@code gcc -std=c99 src.c})
 */
public String generateCompileCommand(String fileName, String compileOptions){
    return compileCommand.replaceAll(opt, compileOptions).replaceAll(src, fileName);
}
```

```
languages[0] = new LangInternal(1 , "c" , "gcc " , false , ".c",
                                "gcc " + LangInternal.opt + " " + LangInternal.src,
                                "./a.out");
languages[2] = new LangInternal(3 , "python" , "python " , true , ".py",
                                null,
                                "python " + LangInternal.opt + " " + LangInternal.src);
languages[3] = new LangInternal(4 , "java" , "javac " , false , ".java",
                                "javac " + LangInternal.opt + " " + LangInternal.src,
                                "java "  + LangInternal.src);
```

```
Pattern.compile("^public[ \t\n\r]+class[ \t\n]+[a-zA-Z][a-zA-Z0-9_]*");
```

Tests : *unitaires*

- JUnit
  - Le package session
- Problème détecté:
  - Gestion du temps

```
@Before
public void setUp()
{
    c = new Client("11001");
    String src = "#include <stdio.h>"
                +"void main(void){"
                +"  printf(\"Hello world !!\n\");"
                +"}";
    t1 = new Task(c, 1, src, "" , false, null, null);
    t2 = new Task(c, 1, src, "" , true, "", "");
    t3 = new Task(c, 1, src, "" , true, null, null);
}
```

```
@Test
public void testInactiveSince() {
    long time1 = t1.inactiveSince();
    try {
        Thread.sleep(2000);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
    long time2 = t1.inactiveSince();
    assertTrue(time1 < time2);
    t1.inActivity();
    long time3 = t1.inactiveSince();
    assertTrue(time2 > time3);
}
```

Tests : *unitaires*

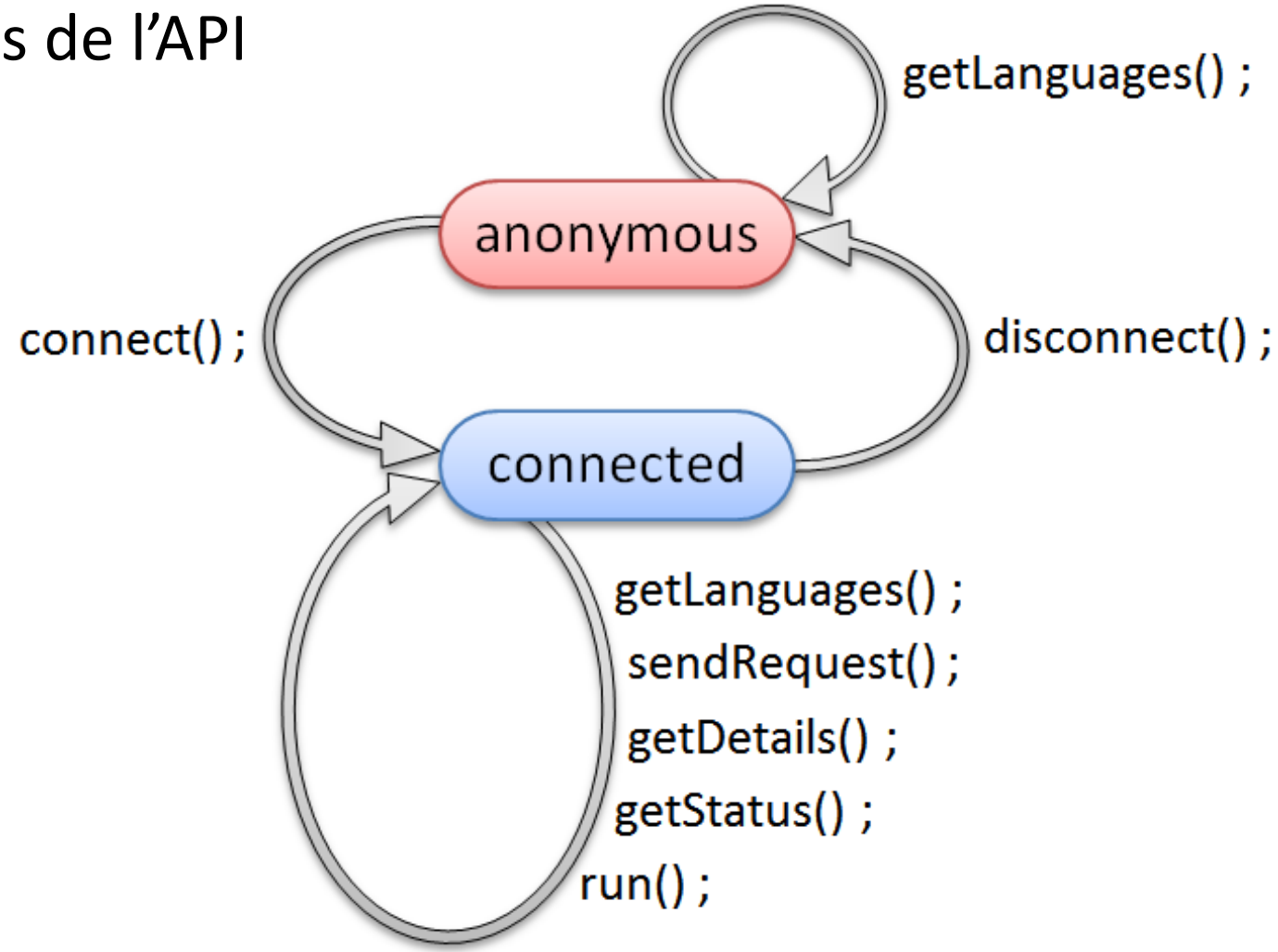
- Séparateurs dans les chemins

```
@Test
public void testKill() {
    ClientInterface myClient = new Client("1111111");
    TaskInterface t = new Task(myClient, 0, "test", "test", false, "test",
            "test");
    String oldDir = myClient.getPath();
    TaskInterface t1 = new Task(myClient, 0, "test", "test", false, "test",
            "test");
    TaskInterface t2 = new Task(myClient, 0, "test", "test", false, "test",
            "test");
    TaskInterface t3 = new Task(myClient, 0, "test", "test", false, "test",
            "test");
    TaskInterface t4 = new Task(myClient, 0, "test", "test", false, "test",
            "test");
    myClient.kill();
    File dir = new File(oldDir);

    assertFalse(dir.exists());
}
```

Tests : *fonctionnels*

- Tests de l'API



Tests : *fonctionnels*

- Client Python
  - API suds

- Problème résolu :
  - 1 requête = 1 port
  - Système d'identification par IP/Port invalide

Tests : *fonctionnels*

- Test du cleaner

```
Server successfully loaded!
>>> Sat Apr 12 15:13:15 CEST 2014:
        getLanguages() => Languages sending
>>> Sat Apr 12 15:13:15 CEST 2014:
        connect() => Client connection :1b32145b2b4b5bf55a60f1d8d7ca1144febef1b7
...

>>> Sat Apr 12 15:13:16 CEST 2014:
        sendRequest() => Client successfully sends a request : 1b32145b2b4b5bf55a60f1d8d7ca1144febef1b7 : 0
>>> Sat Apr 12 15:13:16 CEST 2014:
        connect() => Client connection :ce6d3ad3d398871235f558e6f4eedd4903d0
...

>>> Sat Apr 12 15:13:17 CEST 2014:
        sendRequest() => Client successfully sends a request : ce6d3ad3d398871235f558e6f4eedd4903d0 : 0
>>> Sat Apr 12 15:13:17 CEST 2014:
        session deleted : /tmp/ce6d3ad3d398871235f558e6f4eedd4903d0
>>> Sat Apr 12 15:13:17 CEST 2014:
        disconnect() => Client successfully disconnected : ce6d3ad3d398871235f558e6f4eedd4903d0
```

```
kikouyou@kikouyou-K53SC:~$ tree /tmp/1b32145b2b4b5bf55a60f1d8d7ca1144febef1b7/
/tmp/1b32145b2b4b5bf55a60f1d8d7ca1144febef1b7/
└── 0
    ├── a.out
    └── srcFile.c
```

Tests : *fonctionnels*

- Test du cleaner

```
>>> Sat Apr 12 15:13:17 CEST 2014:
        disconnect() => Client successfully disconnected : ce6d3ad3d398871235f558e6f4eedd4903d0
>>> Sat Apr 12 15:18:01 CEST 2014:
        /!\ cleanning session(s)
>>> Sat Apr 12 15:18:01 CEST 2014:
        session deleted : /tmp/1b32145b2b4b5bf55a60f1d8d7ca1144febef1b7
>>> Sat Apr 12 15:18:01 CEST 2014:
        /!\ cleanning task(s)
```

```
kikouyou@kikouyou-K53SC:~$ ls /tmp/
hsperfdata_kikouyou  hsperfdata_root  pulse-ivq7AOC14aj0  suds
```

Tests : *performance*

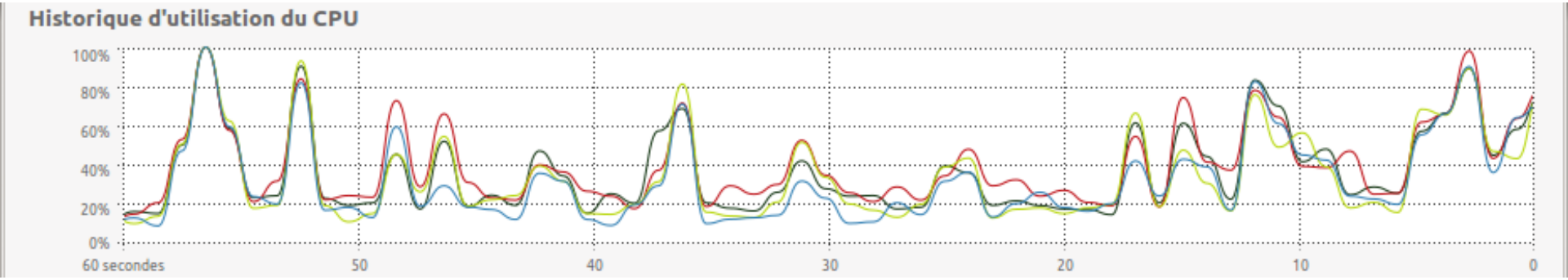
- Test de vitesse de traitement
  - Même algorithme dans 4 langages

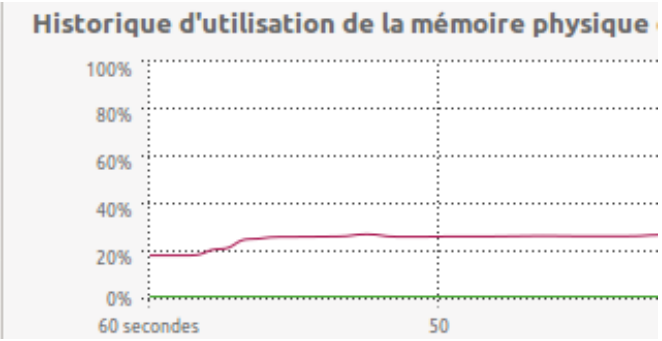
```
python_code = """import sys
i = int(sys.argv[1])
for j in range(0 , i) :
    print "Hello world from Docker , it rocks!, i'm the "  + str(j) + " iteration"
line = sys.stdin.readlines();
print "the string is %s" %line
"""
```

	Temps de transfert	Temps de compilation	Temps d'exécution
Java	13	1236	544
C	15	577	456
C++	15	576	479
Python	8	525	522
Moyennes (ms)	12.75	728.5	500.25
Répartition du temps (%)	1.03	58.68	40.29

Tests : *performance*

- Test de montée en charge
  - 1 Serveur
  - 3 Clients distants ouvrant 50 sessions de 32 requêtes.
  - Soit : 4800 requêtes lancées simultanément !





- Problème détecté et résolu:
  - Collisions du Hash de clé de session.

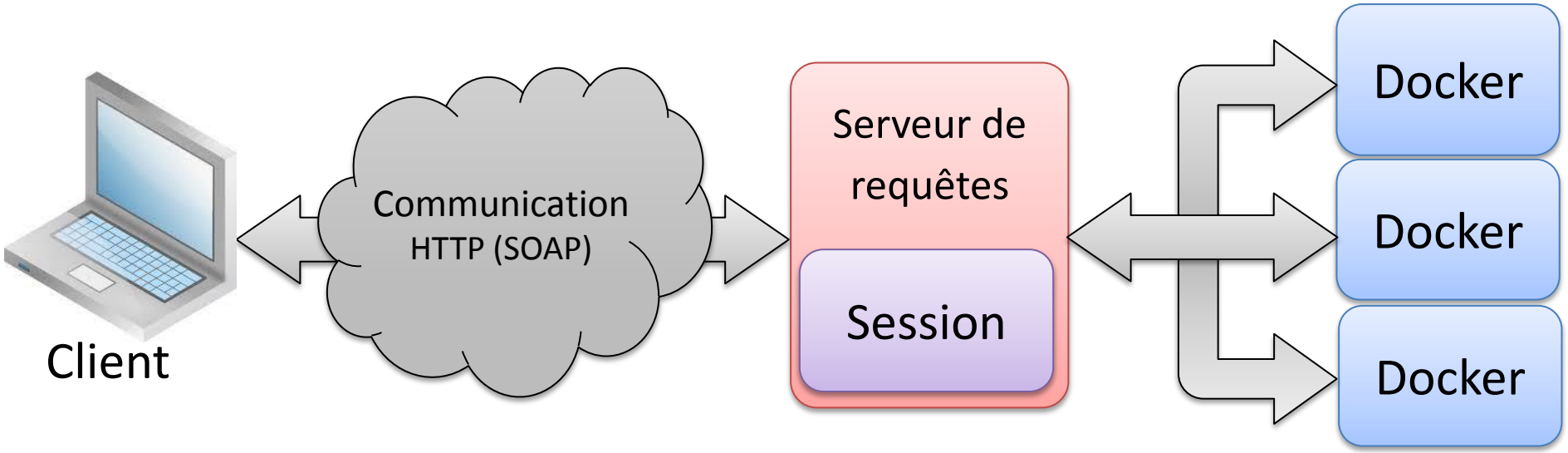
Améliorations possibles

- Java Native Interface
  - Déléguer la partie Docker à du code C.

- API Java Logging
  - Utiliser la classe java dédiée à la gestion des logs.

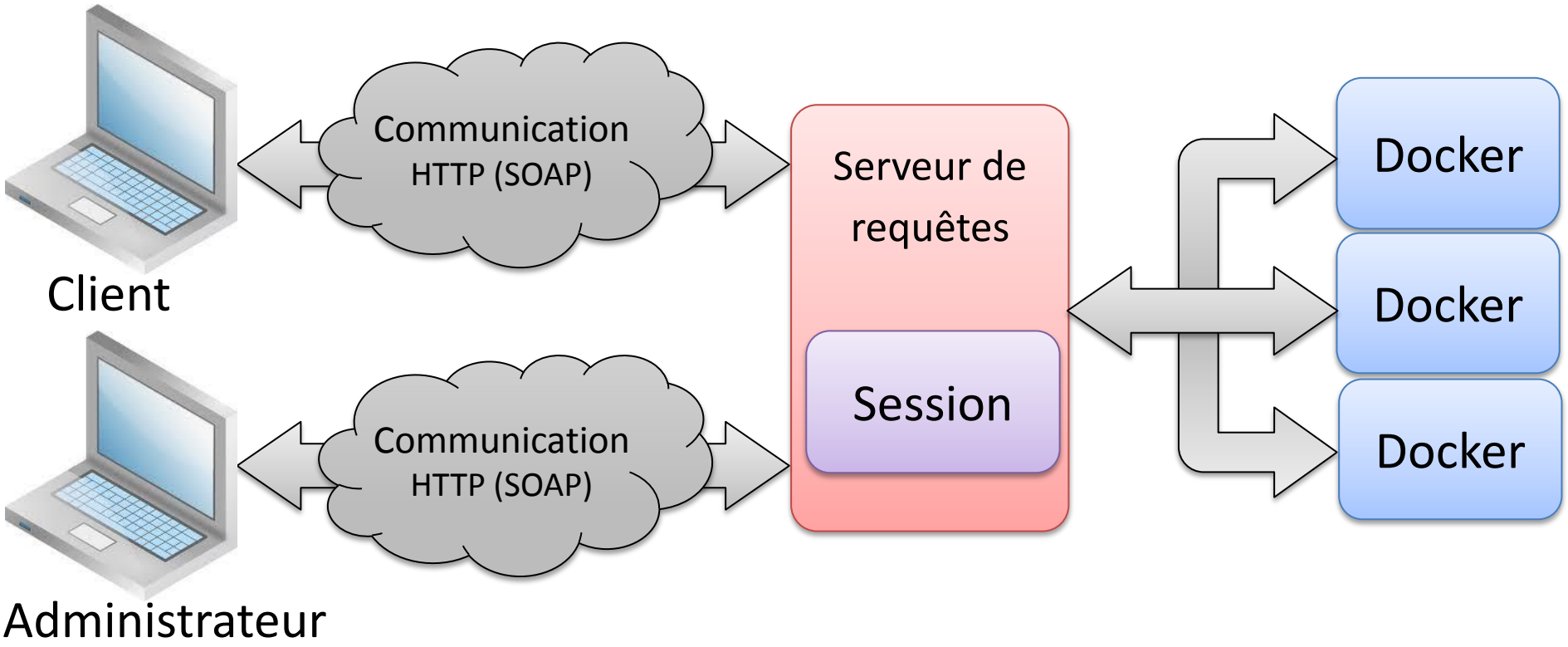
Version 2

- Plusieurs serveurs physiques docker



Version 3

- Administration à chaud



Merci pour votre attention !