



Bluemix en Pratique

Exercice 1 - Mes premiers déploiements

Version:

1

Modification:

28-juin-15

Propriétaire:

Philippe THOMAS for Devoxx

Table des matières

Bluemix en Pratique	1
Exercice 1 - Mes premiers déploiements	1
Exercice 1.a – Déploiement d'une première application	4
Exercice 1.b – Déploiement d'une application simple Node.js avec la CLI de Cloud Foundry	10
Exercice 1.c – Utilisation de DevOps Services pour le déploiement d'une application.....	16

IBM Bluemix est une nouvelle solution d'IBM qui permet de construire, de déployer et de gérer tout type d'applications (mobile, smart devices, Internet des objets, web, big data, analytics). Bluemix est basée sur les standards ouverts du marché (Cloud Foundry) et il utilise une plate-forme Cloud (SoftLayer) pour déployer ces nouvelles applications. Parmi la longue liste de fonctionnalités de Bluemix, on notera la capacité de construire des back-end applicatifs pour les applications mobiles, le monitoring et l'analyse des performances intégrés ainsi que le dispositif de scalabilité automatique qui peut être ajouté à toute application pour lui permettre d'absorber une charge plus importante. Bluemix est réellement une plate-forme ouverte car en plus des services IBM proposés (plus de 40), elle intègre également les services des partenaires ainsi que ceux de la communauté Open Source. L'un des avantages principaux de Bluemix est de permettre la construction et le déploiement très rapide d'une application et de ses services associés (base de données SQL ou non, file d'attentes, envoi de SMS, récupération de tweets, analytics, Watson Q&A, ...).

Exercice 1.a – Déploiement d'une première application

Cet exercice vous montrera comment déployer une application avec la console Bluemix.

Dans votre navigateur, suivez le lien suivant:

<http://bluemix.net>

Ce lien vous amènera vers le site de Bluemix

The screenshot shows the IBM Bluemix homepage. At the top, there's a navigation bar with links for 'SOLUTIONS', 'CATALOG', 'PRICING', 'DOCS', 'COMMUNITY', 'SIGN UP' (in blue), and 'LOG IN'. Below the navigation bar, the main header reads 'IBM Bluemix' and 'The Digital Innovation Platform'. There's a 'GET STARTED FREE' button. Below the header, there are five main navigation tabs: 'BUILD', 'EXTEND', 'SCALE', 'INTEGRATE', and 'FEATURED'. The background features a dark, geometric pattern.

Choisissez ‘LOG IN’ en haut à droite puis saisissez votre identifiant et votre mot de passe.
Cliquez sur ‘Sign In’. Vous devriez voir une page similaire à celle-ci:

The screenshot shows the IBM Bluemix dashboard. On the left, there's a sidebar with 'Create a Space' and a list of organizations: 'f103544@fr.ibm.com' (selected), 'First', 'Mobile on Bluemix', 'rockstar', and 'wear2Rewards'. The main area has a heading 'Deploy your app the way you want' and a sub-section 'Cloud Foundry' with icons for 'CLOUD FOUNDRY', 'IBM CONTAINERS', and 'VIRTUAL MACHINES'. It shows '2.06 GB out of 18 GB of memory', '0 APP HEALTH', and '13/80 SERVICES'. A search bar at the top right says 'Type to search'.

Selectionnez le menu ‘CATALOG’ en haut de l'écran:

The screenshot shows the IBM Bluemix Catalog. On the left, there are two expandable sections: 'Category' (with items like Watson, Mobile, DevOps, etc.) and 'Support' (with IBM). The main area has a heading 'Starters // Choose a package of sample code and services, or start from scratch'. It shows a grid of 'Boilerplates' with the 'Java Cloudant Web Starter' icon highlighted with a red box. Other items include 'Internet of Things', 'JAVA', 'Node.js Cache Web St...', 'Node.js Cloudant DB ...', 'Personality Insights Jav...', 'Personality Insights No ...', 'Node-RED Starter Community', and 'Vaadin Rich Web Starter Community'. A search bar at the top says 'Type here to search'.

Selectionnez le ‘Java Cloudant Web Starter’ depuis la section Boilerplates (un boilerplate est une application toute prête avec ses services associés).

Exercice 1 - Mes premiers déploiements

The screenshot shows the IBM Bluemix Catalog interface. On the left, there's a sidebar with a search bar and a link to 'Back to All Categories'. The main area displays the 'Java Cloudant Web Starter' application, which is described as demonstrating how to use the Cloudant NoSQL DB service with the 'Liberty for Java™' runtime on the IBM Cloud. It includes a version (1.0) and type (Boilerplate) section, and a 'VIEW DOCS' button. To the right, there's a summary of the 'java liberty' service, including icons for Cloudant NoSQL DB and Monitoring and Analytics, and a 'VIEW DOCS' button. Below this is a 'Pick a plan' section showing a 'Default' plan for running one or more apps for 30 days (375 GB-hours free) at a price of €0.0526 EUR/GB-Hour. A note says it's a service plan for the IBM Bluemix Platform runtime. At the bottom right of the catalog area, there's a 'TERMS' button. The right side of the screen is occupied by a 'Create an app' form. It asks for a 'Space' (Bluemix Days), 'Name' (MyFirstDeployXX), 'Host' (MyFirstDeployXX), and 'Domain' (mybluemix.net). Under 'Selected Plans', 'Liberty for Java™ - Default' is chosen. For 'Cloudant NoSQL DB', 'Shared' is selected. For 'Monitoring and Analytics', 'Free' is selected. A warning message in a yellow triangle states: 'An instance of the Monitoring and Analytics service already exists'.

Saisir un nom pour votre application comme indiquer ci-dessus (MyFirstDeployXX). Le nom du host sera rempli automatiquement mais il doit être unique dans Bluemix car il fait parti de l'URL de l'application. Afin de s'assurer de l'unicité pendant l'exercice, utilisez vos initiales et un numéro afin d'obtenir un nom de host unique.

Appuyer sur 'CREATE'

Après un temps relativement court (environ une minute), votre application doit être active (rond vert à droite avec Your app is running).

Cliquez sur 'Overview' dans le menu à gauche. Vous pourrez alors lancer l'application dans un autre onglet en cliquant sur la route (lien en bleu sous le nom de l'application):

The screenshot shows the IBM Bluemix Overview page for the application 'MyFirstDeployXX'. The left sidebar has a 'Back to Dashboard' button and lists 'MyFirstDeployXX' under 'Overview', 'LIBERTY for Java™', 'Files and Logs', 'Environment Variables', 'Start Coding', 'Cloudant NoSQL DB', and 'Monitoring and Analytics'. The main area has a header 'MyFirstDeployXX' with a green status icon and a 'Routes: MyFirstDeployXX.mybluemix.net' link (circled in red). Below this are sections for 'LIBERTY FOR JAVA™' (Instances: 1, Memory Quota: 512, Available Memory: 12.438 GB), 'ADD A SERVICE OR API' (button), 'BIND A SERVICE OR API' (button), 'Monitoring and Analytics' (MyDevopsServiceAppCL-Mo... Free), and 'Cloudant NoSQL DB' (MyFirstDeployXX-cloudantNo... Shared). On the right, there's an 'APP HEALTH' section with 'Your app is running.' and 'RESTART' and 'STOP' buttons, and an 'ACTIVITY LOG' section showing log entries: 3/13/15 5:12 PM M103544@fr.ibm.com started MyFirstDeployXX app, 3/13/15 5:12 PM M103544@fr.ibm.com updated MyFirstDeployXX app • changed routes, and 3/13/15 5:12 PM M103544@fr.ibm.com created MyFirstDeployXX app. At the bottom right is a 'Estimate the cost of this app' button.

L'application s'affiche dans un onglet différent et montre une liste de fichiers (un seul au départ) qui sont stockés dans une base NoSQL Cloudant (sur le cloud).

En cliquant sur le lien ‘Sample.txt’, vous visualiserez le contenu du document dans la base NoSQL.

Dans cet exemple simple, nous avons utilisé un boilerplate (modèle d’application) qui contient un runtime Java et deux services (une base de données NoSQL Cloudant et un dispositif de surveillance – Monitoring and Analytics).

En complément de cet exercice, vous pouvez regarder les informations associées à ces deux services.

Tout d’abord, cliquez sur Dashboard en haut de la page. Puis dans la section ‘Services’, vous verrez vos deux services. Choisissez celui qui s’appelle ‘Cloudant NoSQL DB’. Chaque service dispose d’une console associée.

Puis cliquez sur ‘LAUNCH’ en haut à droite :

The Cloudant NoSQL Database service adds JSON data to your Mobile and Web applications, accessible via easy-to-use RESTful HTTP/S APIs.

Face of IaaS **Powerful search, sync and more**

LAUNCH

L'écran suivant apparaît dans un nouvel onglet du navigateur : il s'agit de la console spécifique du service Cloudant. On notera entre autre la base de donnée (sample_nosql_db).

Name	Size	# of Docs	Update Seq
sample_nosql_db	202 bytes	1	2

En cliquant sur le nom de la base de donnée, on accède aux documents de cette base.

Permissions Changes All Documents All Design Docs

id "1427807906516"

```
{
  "_id": "1427807906516",
  "_rev": "2-06d225eab5b4628b443766c8a280c863",
  "value": {
    "rev": "2-06d225eab5b4628b443766c8a280c863"
  },
  "key": "1427807906516"
}
```

Enfin, en cliquant sur le crayon à droite, on accède aux documents de la base :

Save Cancel

```

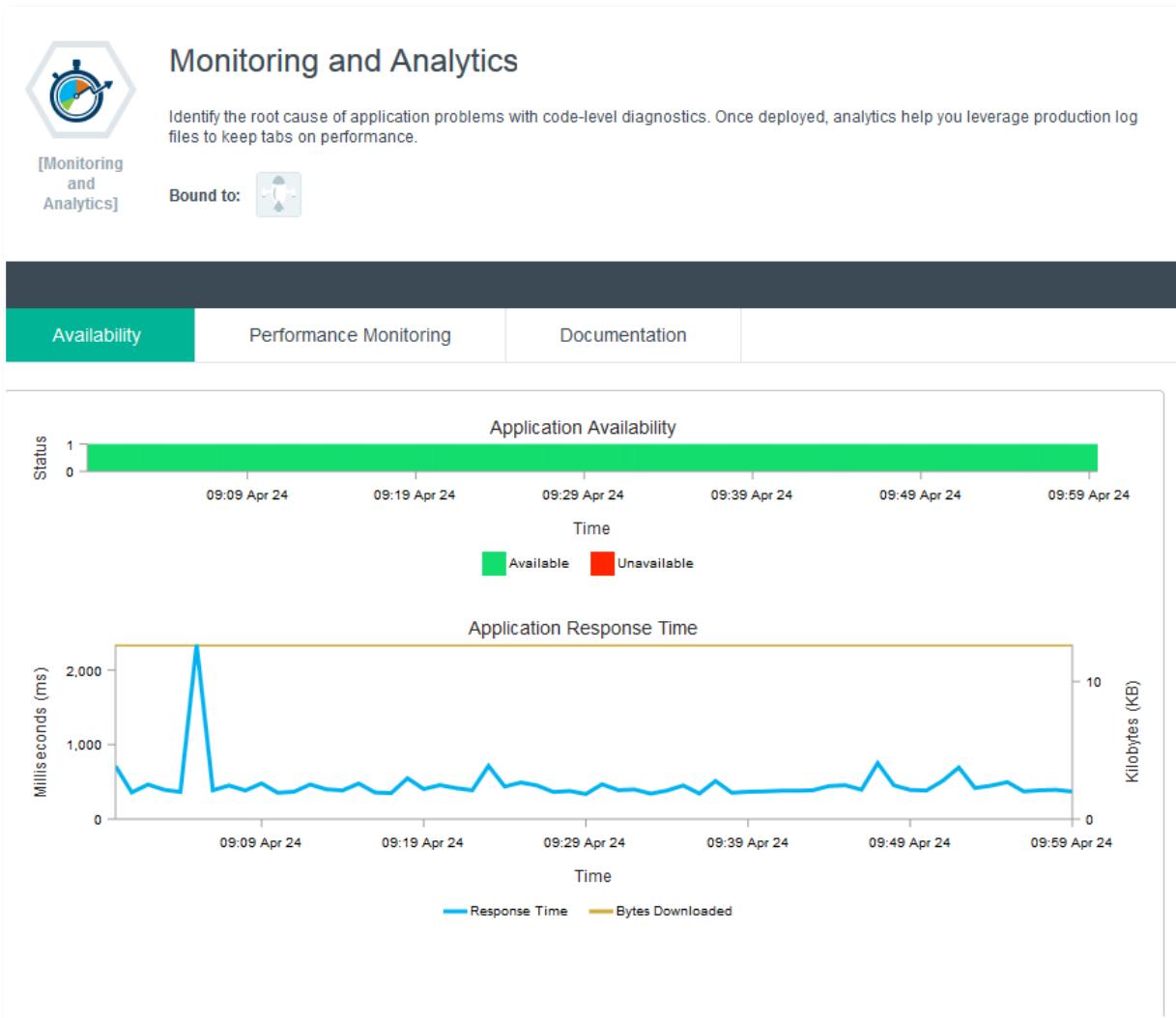
1 <
2   "_id": "1427807906516",
3   "_rev": "2-06d225eab5b4628b443766c8a280c863",
4   "creation_date": "Tue Mar 31 13:18:26 UTC 2015",
5   "name": "Sample category",
6   "value": "List of sample files",
7   "-attachments": {
8     "Sample.txt": {
9       "content_type": "text/plain",
10      "revpos": 2,
11      "digest": "md5-tkNL+d1yoxX4H1QKkjnrHA==",
12      "length": 24,
13      "stub": true
14    }
15  }

```

View Attachments Upload Attachment Clone Document Delete

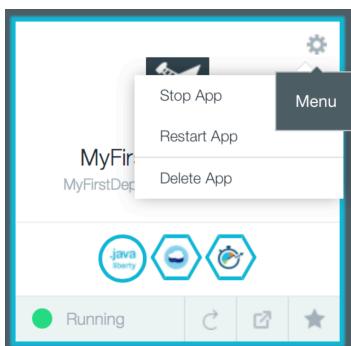
Fermez cet onglet puis revenir sur le Dashboard Bluemix. Choisir l'autre service intitulé Monitoring and Analytics.

La console du service est activée automatiquement. Elle montre des statistiques de disponibilité et de performance de l'application.

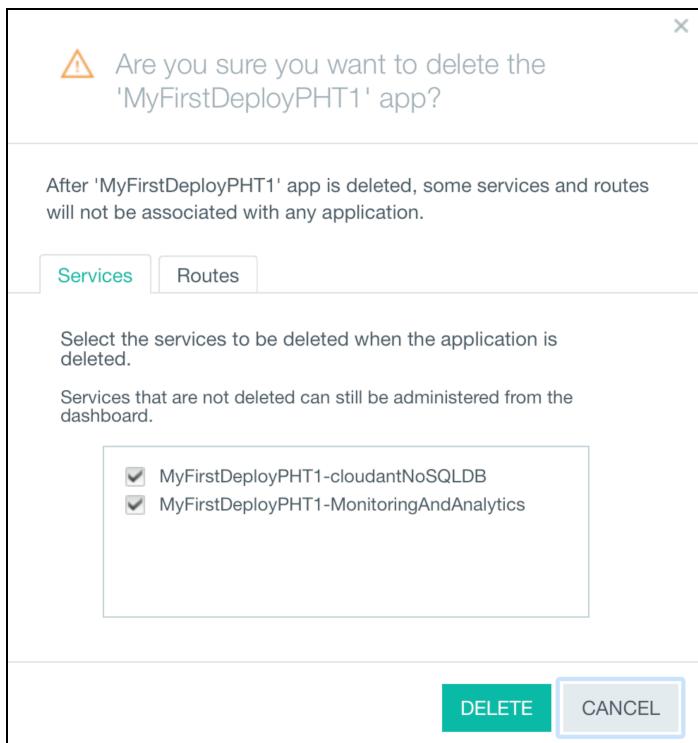


Nous allons supprimer notre application.

Cliquez sur le Dashboard puis sur votre application, cliquer sur la roue dentelée en haut à droite :



Vous pouvez cliquer sur 'Delete App' puis sur 'Delete' :



Pour supprimer les services et les routes associés, il ne faut pas décocher les services et les routes dans le menu.

Exercice 1.b – Déploiement d'une application simple Node.js avec la CLI de Cloud Foundry

Dans cet exercice, nous allons déployer une nouvelle application en JavaScript. Le code que nous allons utiliser est très simple (c'est le 'Hello World' de la page d'accueil du site Node.js) :

```
var http = require('http');
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/plain'});
  res.end('Hello World\n');
}).listen(3000);
```

Afin de faire tourner cette application sur Bluemix dans le cloud, nous allons apporter quelques petites modifications à ce code et créer un fichier app.js dans un répertoire de votre choix:

```
var http = require('http');
var appport = process.env.VCAP_APP_PORT || 3000;
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/plain'});
  res.end('Hello World\n');
}).listen(appport);
```

Le premier changement concerne le port utilisé par Bluemix pour notre application. Donc nous pouvons choisir quel port va être utilisé si nous tournons ce programme dans Bluemix (utilisation de la variable VCAP_APP_PORT) ou bien sur une machine dédiée (3000). De la même façon, il existe d'autres variables qui caractérisent notre application. Voir cette URL pour toutes les variables :

<http://docs.cloudfoundry.org/devguide/deploy-apps/environment-variable.html>

L'autre changement consiste en la création d'un fichier de gestion des dépendances Node.js : ce fichier doit être créé dans le même répertoire et s'appelle package.json. Remplacer XX avec vos initiales.

```
{
  "name": "NodeTestXX",
  "version": "0.0.1",
  "description": "Sample Node application"
}
```

Finalement, on se retrouve avec deux fichiers app.js et package.json dans un répertoire :

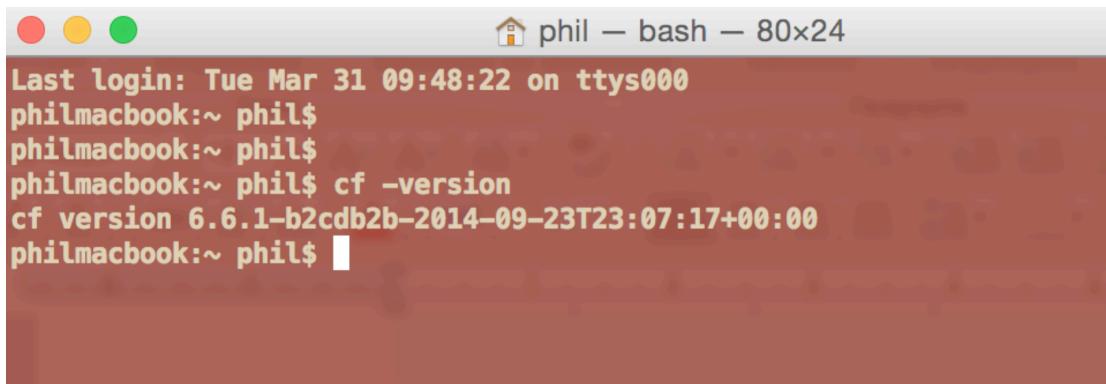
Name	Date Modified
app.js	24 Mar 2015 16:59
package.json	24 Mar 2015 17:02

Vous pouvez ouvrir une fenêtre de ligne de commande (cmd sous Windows) ou un terminal sur les autres OS.

La première chose à faire lorsqu'on veut faire un déploiement depuis sa machine vers Bluemix, il faut se connecter à distance sur le serveur Cloud Foundry de Bluemix.

Vérifier que vous avez bien installé la ligne de commande Cloud Foundry en tapant la commande :

```
cf -version
```



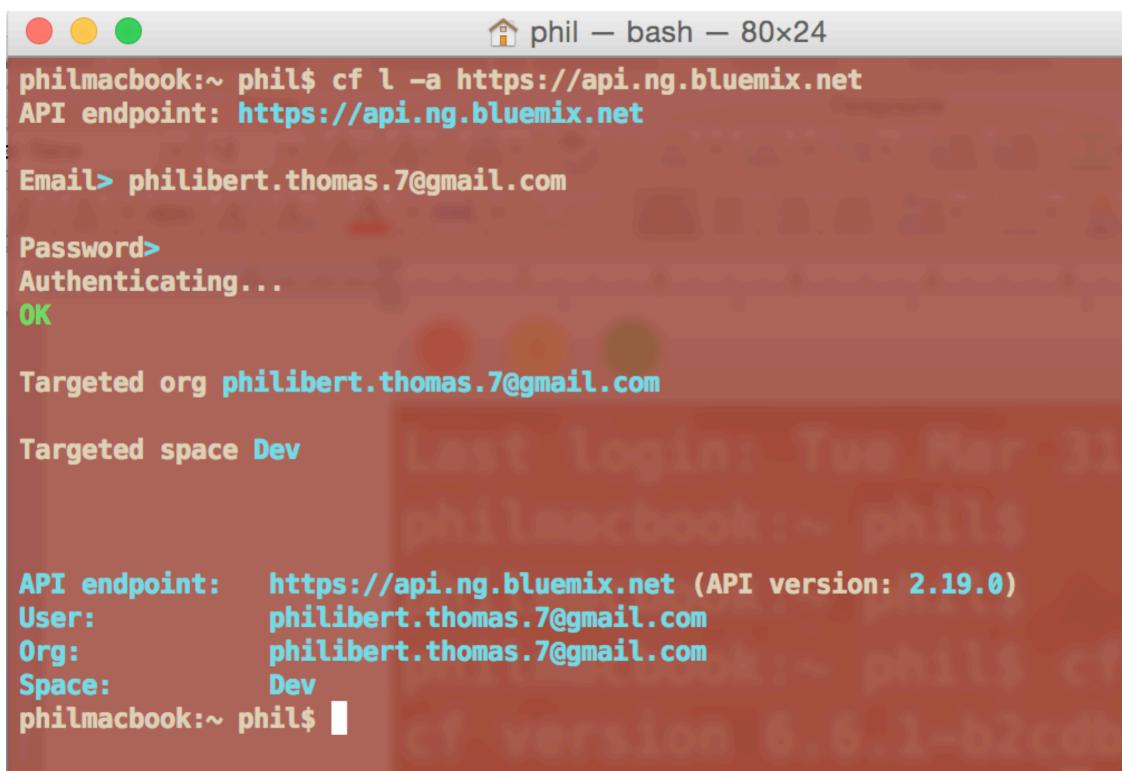
```
Last login: Tue Mar 31 09:48:22 on ttys000
philmacbook:~ phil$ 
philmacbook:~ phil$ 
philmacbook:~ phil$ cf -version
cf version 6.6.1-b2cdb2b-2014-09-23T23:07:17+00:00
philmacbook:~ phil$ 
```

A screenshot of a Mac OS X terminal window titled "phil – bash – 80x24". The window shows a command-line session where the user has run the "cf -version" command. The output indicates that the Cloud Foundry version is 6.6.1-b2cdb2b-2014-09-23T23:07:17+00:00. The terminal interface includes standard Mac OS X window controls (red, yellow, green buttons) and a scroll bar.

Ensuite, il faut se connecter à Bluemix en utilisant la bonne URL :

```
cf l -a https://api.ng.bluemix.net      (Region: US South)
cf l -a https://api.eu-gb.bluemix.net    (Region: United Kingdom)
```

Il faudra remplir l'email, le mot de passe et quelque fois l'espace.



```
philmacbook:~ phil$ cf l -a https://api.ng.bluemix.net
API endpoint: https://api.ng.bluemix.net

Email> philibert.thomas.7@gmail.com

Password>
Authenticating...
OK

Targeted org philibert.thomas.7@gmail.com

Targeted space Dev

API endpoint: https://api.ng.bluemix.net (API version: 2.19.0)
User: philibert.thomas.7@gmail.com
Org: philibert.thomas.7@gmail.com
Space: Dev
philmacbook:~ phil$ 
```

A screenshot of a Mac OS X terminal window titled "phil – bash – 80x24". The window shows a session where the user has run the "cf l -a https://api.ng.bluemix.net" command. It prompts for an email ("Email>"), which is entered as "philibert.thomas.7@gmail.com". It then prompts for a password ("Password>"), followed by "Authenticating..." and "OK". After authentication, it displays the targeted organization ("Targeted org philibert.thomas.7@gmail.com") and space ("Targeted space Dev"). Finally, it shows the API endpoint and user information ("API endpoint: https://api.ng.bluemix.net (API version: 2.19.0)", "User: philibert.thomas.7@gmail.com", "Org: philibert.thomas.7@gmail.com", "Space: Dev"). The terminal interface includes standard Mac OS X window controls and a scroll bar.

Il faut maintenant faire le déploiement de l'application (sans aucun service). Remplacer XX par vos initiales.

```
cd mydir
cf push NodeTestXX -c 'node app.js'      (on Linux or OSX)
cf push NodeTestXX -c "node app.js"        (on Windows)
```

```
philmacbook:sample phil$ cd lab1
philmacbook:lab1 phil$ cf push NodeTestPHT -c 'node app.js'
Creating app NodeTestPHT in org philibert.thomas.7@gmail.com / space Dev as philibert.thomas.7@gmail.com...
OK

Creating route nodetestpht.mybluemix.net...
OK

Binding nodetestpht.mybluemix.net to NodeTestPHT...
OK

Uploading NodeTestPHT...
Uploading app files from: /Users/phil/sample/lab1
Uploading 547, 2 files
OK

Starting app NodeTestPHT in org philibert.thomas.7@gmail.com / space Dev as philibert.thomas.7@gmail.com...
OK
----> Downloaded app package (4.0K)
----> Node.js Buildpack Version: v1.14-20150309-1555
    TIP: Specify a node version in package.json
----> Defaulting to latest stable node: 0.10.36
----> Installing IBM SDK for Node.js from cache
----> Checking and configuring service extensions
----> Installing dependencies
    npm WARN package.json NodeTestPHT@0.0.1 No repository field.
    npm WARN package.json NodeTestPHT@0.0.1 No README data
----> Cleaning up node-gyp and npm artifacts
----> A 'Procfile' is not found and the npm start script is undefined.
    TIP: Specify your start script in 'package.json' or 'Procfile'.
----> Building runtime environment
----> Checking and configuring service extensions
----> Installing App Management
----> WARN: App Management (Development Mode) cannot be installed because the start script cannot be found.
    TIP: Specify your start script in 'package.json' or 'Procfile'.
----> Node.js Buildpack is done creating the droplet

----> Uploading droplet (6.7M)

0 of 1 instances running, 1 starting
0 of 1 instances running, 1 starting
1 of 1 instances running
```

On peut voir toutes les étapes du déploiement de notre application :

- Création de l'application dans Bluemix
- Création de la route (URL de l'application)
- Association de la route avec l'application (Binding)
- Chargement du code depuis le poste de travail vers Bluemix (Uploading)
- Démarrage de l'application avec Node.js et vérification des dépendances
- Finalement l'instance de l'application est démarrée

```
App started

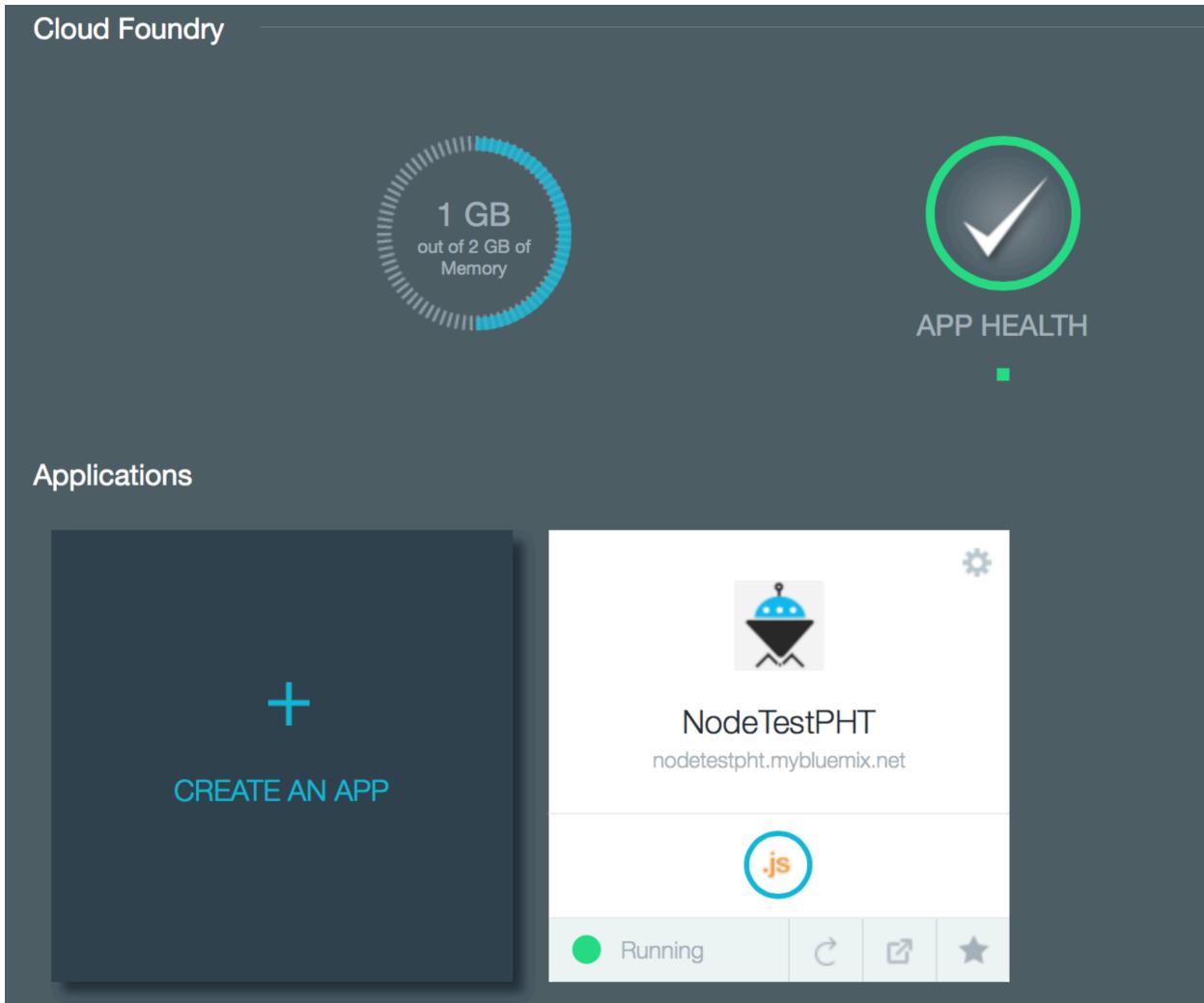
Showing health and status for app NodeTestPHT in org philibert.thomas.7@gmail.com / space Dev as philibert.thomas.7@gmail.com...
OK

requested state: started
instances: 1/1
usage: 1G x 1 instances
urls: nodetestpht.mybluemix.net

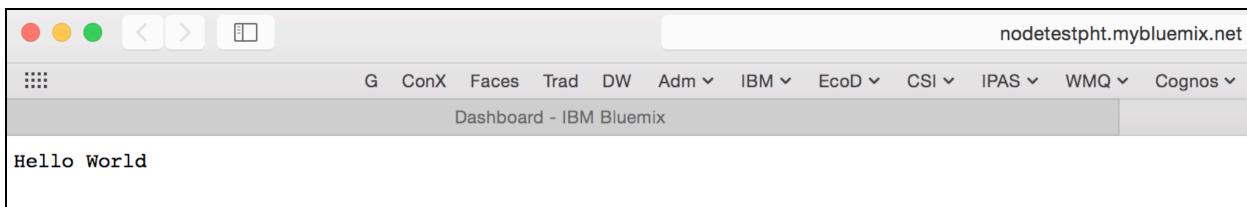
      state  since          cpu    memory      disk
#0  running  2015-04-01 08:52:43 AM  0.0%  11.5M of 1G  24.9M of 1G
philmacbook:lab1 phil$
```

On notera l'activité de l'application par ses métriques : CPU, memory, disk.

En basculant vers la console Bluemix, on notera l'apparition d'une nouvelle application NodeTestXX dans le Dashboard :



Finalement, vous pourrez tester votre nouvelle application en utilisant son URL ou en cliquant sur sa route.



On notera que par défaut l'application utilise 1Go de mémoire.
Il est possible de contrôler certains paramètres de démarrage de l'application grâce à des paramètres sur la ligne de commande **cf push** ou bien en utilisant un fichier **manifest.yml**. Des explications plus précises sont fournies sur le site de Cloud Foundry :

<http://docs.cloudfoundry.org/devguide/deploy-apps/manifest.html>

Voici un exemple de fichier **manifest.yml** que vous pourrez créer sur notre exemple (ne pas oublier de remplacer XX par vos initiales) :

```
applications:
- name: NodeTestXX
host: NodeTestXX
command: node app.js
memory: 128M
```

Nous avons maintenant trois fichiers dans notre répertoire :

Name	Date Modified	Size	Kind
app.js	Today 08:41	217 bytes	JavaScri
manifest.yml	Today 12:06	92 bytes	Subli...c
package.json	Today 08:42	88 bytes	JSON

Redéployer notre application :

```
cd mydir
cf push NodeTestXX -c 'node app.js'      (on Linux or OSX)
cf push NodeTestXX -c "node app.js"       (on Windows)
```

Une fois déployée, vous pourrez constater que Bluemix a défini une taille mémoire de 128 Mo au lieu de 1 Go.

```
App started
Showing health and status for app NodeTestXX in org philibert.thomas.7@gmail.com / space Dev as philibert.thomas.7@gmail.com...
OK

requested state: started
instances: 1/1
usage: 128M x 1 instances
urls: nodetestpht.mybluemix.net

state      since          cpu    memory      disk
#0  running   2015-04-01 12:12:18 PM  0.0%  11.1M of 128M  24.9M of 1G
philmacbook:lab1 phil$
```

Il peut être intéressant de faire un tour des commandes de Cloud Foundry (remplacer APP avec le nom de votre application) :

```
cf a                                (lister toutes les applications)
cf start APP
cf stop APP
cf delete APP
cf logs APP --recent
cf events APP

cf env APP                         (lister les variables VCAP de l'app)
cf s                               (lister les services dans un espace donné)
cf m                               (lister le catalogue des services)
```

On supprimera finalement cette application afin d'avoir un peu plus de mémoire pour le prochain exercice.

Exercice 1.c – Utilisation de DevOps Services pour le déploiement d'une application

Pour travailler avec Bluemix, vous avez plusieurs possibilités pour éditer et développer votre application : il est possible d'utiliser votre éditeur favori, un simple Notepad ou TextPad, Eclipse, Visual Studio ou encore un IDE Web. IBM DevOps Services est un parfait exemple d'environnement de développement collaboratif comprenant un IDE Web qui permet de développer une application depuis un repository Git et permet ainsi de la déployer dans Bluemix. Comme nous le verrons, DevOps Services présente bien d'autres avantages comme le suivi de votre code, la collaboration, la gestion du commit, la planification du projet ...

Pour cet exercice, vous devez avoir un compte DevOps Service sur :
<https://hub.jazz.net>

Dans le catalogue de Bluemix, choisir le boilerplate ‘Node.js Cache Web Starter’.



Choisir un nom pour cette application : ‘NodeJSDevOpsXX’ avec XX qui correspondent à vos initiales par exemple :

A screenshot of the Bluemix application creation interface. On the left, there's a summary of the selected service: 'Node.js Cache Web Starter' (version 0.0, type Boilerplate). In the center, there's a summary of the service: 'SDK for Node.js™', 'Data Cache', and 'Monitoring and...'. Below this, a description states: 'Develop, deploy, and scale server-side JavaScript® apps with ease. The IBM SDK for Node.js™ provides enhanced performance, security, and serviceability.' There are 'VIEW DOCS' and 'CREATE' buttons. On the right, there's a form for creating the app, including fields for 'Name' (NodeJSDevOpsXX), 'Host' (NodeJSDevOpsXX), 'Domain' (mybluemix.net), and 'Selected Plans' (SDK for Node.js™: Default, Data Cache: Free, Monitoring and Analytics: Free). The 'CREATE' button is at the bottom right.

Une fois cette application déployée et démarrée, cliquer sur ‘ADD GIT’ en haut à droite :

Assurer vous que la case ‘Populate ...’ est bien cochée :

Cliquer sur ‘EDIT CODE’ pour accéder à votre code dans DevOps Services (on notera au passage le lien du repository GIT) :

Ce projet utilise Node.js et le framework Express.js.

En ouvrant le fichier package.json dans l'éditeur de texte de DevOps, vous verrez toutes les dépendances avec Express, ejjs ... et node.

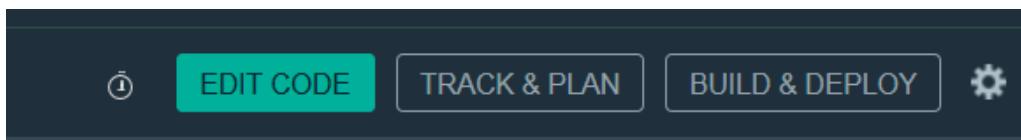
The screenshot shows the IBM DevOps interface with the project 'phthon | NodeJSDevOpsXX 1'. The left sidebar lists files like .git, launchConfigurations, public, routes, views, .configignore, .gitignore, app.js, License.txt, manifest.yml, and package.json. The package.json file is selected and shown in the main editor area. The code content is:

```

1  {
2    "name": "Node-CacheApp",
3    "version": "0.0.1",
4    "description": "A sample nodejs app using Elastic Caching Service for Bluemix",
5    "scripts": {
6      "start": "node app.js"
7    },
8    "dependencies": {
9      "express": "3.2.6",
10     "ejjs": "*",
11     "querystring": "0.2.0"
12   },
13   "engines": {
14     "node": "0.10.26"
15   },
16   "repository": {}
17 }
18

```

Les applications Node.js n'ont pas besoin de compilation avant le déploiement. Lorsqu'une modification est faite, il suffit de cliquer sur 'Build & Deploy'.



Si vous sélectionnez l'onglet 'BUILD & DEPLOY', puis l'option 'Configure Stage' dans le 'Build Stage' ...

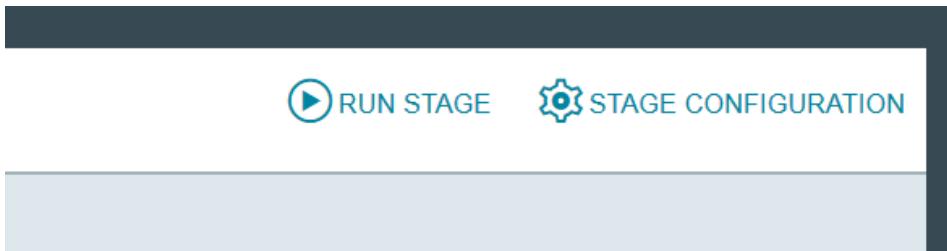
The screenshot shows the 'Pipeline: All Stages' view. It displays two stages: 'Build Stage' and 'Deploy Stage'. A context menu is open over the 'Build Stage' section, showing options 'Configure Stage' and 'Delete Stage'. The 'Build Stage' section shows 'INPUT' (Last Input: Not yet run), 'JOBS' (NO JOBS RUN, with a 'Build' button), and 'LAST EXECUTION RESULT' (No results). The 'Deploy Stage' section shows 'JOBS' (NO JOBS RUN, with a 'Deploy to Bluemix Days' button) and 'LAST EXECUTION RESULT' (No results).

... vous verrez que le builder est de type 'Simple' (contrairement aux langages compilés qui indique le nom du compilateur) :

Vous constaterez également que le fonctionnement est orienté ‘build and deploy automatically’ dès que les changements sont poussés dans le pipeline.
Cliquer sur la flèche gauche dans en haut de “Stage Configuration” pour revenir sur la page de pipeline.

Pour tester votre déploiement, cliquer sur BUILD au milieu du Build Stage (on va faire un premier déploiement).

Puis sur ‘Run Stage’



Après un instant, le ‘Build and Deploy’ seront exécutés avec succès :

Stage Job History

Build Stage

JOB EXECUTION ORDER

Build

Build 1 Success 2 hours ago

INPUT [Add starter application package](#) TRIGGERED BY phthom DURATION 5 seconds

DEPLOYED TO day (US South)

LOGS CHANGES ARTIFACTS

```
Started by user phthom
Building remotely on jenkins-build-slave-5659f343385c (*.Build) in workspace /home/jenkins/workspace/86ddce5b-88e3-535a-c0f2-5-8101-e04517ae05c2
Cloning the remote Git repository
Cloning repository https://hub.jazz.net/git/phthom/NodeJSDevOpsXX.1
Fetching upstream changes from https://hub.jazz.net/git/phthom/NodeJSDevOpsXX.1
using .gitcredentials to set credentials
Checking out Revision 4b8941b7e5768bd7894044650af37020f6cf4d87 (detached)
First time build. Skipping changelog.
Uploading artifacts ...
UPLOAD SUCCESSFUL
Total time: 0 seconds
Finished: SUCCESS
```

En cliquant sur le retour en haut à gauche (flèche)

The screenshot shows a pipeline interface with two stages: Build Stage and Deploy Stage.

Build Stage:

- INPUT:** Git URL (Last commit by Philippe Thomas, 2 hr ago), Add starter application package.
- JOB:** Build Succeeded 2 hr ago (status bar: JOBS COMPLETED SUCCESSFULLY).
- LAST EXECUTION RESULT:** Build 1 (with a build icon).

Deploy Stage:

- INPUT:** Stage: Build Stage / Job: Build, Last Input: Build 1.
- JOB:** Deploy to day Succeeded 2 hr ago (status bar: JOBS COMPLETED SUCCESSFULLY).
- LAST EXECUTION RESULT:** NodeJSDevOpsXX (NodeJSDevOpsXX.mybluemix.pvt.fr) (with a deployment icon), View runtime log.

Nous allons ensuite modifier le code en ouvrant avec l'éditeur le fichier suivant routes/cache.js. A la fin du fichier, trouver les fonctions putCache et removeCache. Modifier les messages suivants par d'autres messages:

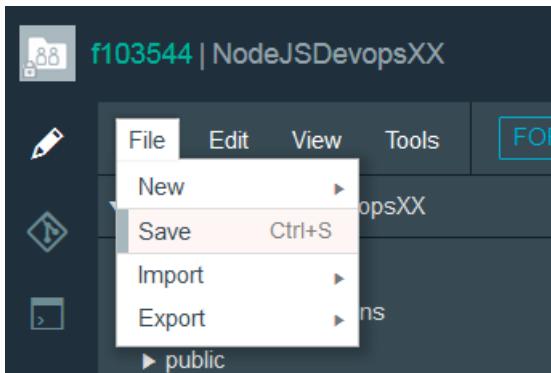
Put successfully. => Stored it
Remove successfully. => Deleted it

```

53 exports.putCache = function(req, res) {
54   var key = req.query.key;
55   var value = req.query.value;
56   wxsclient.put(key, value, function() {
57     res.json({
58       value : "Stored it."
59     });
60   });
61 };
62
63 exports.removeCache = function(req, res) {
64   var key = req.params.key;
65   wxsclient.remove(key, function() {
66     res.json({
67       value : "Deleted it."
68     });
69     console.log('finished remove');
70   });
71 };

```

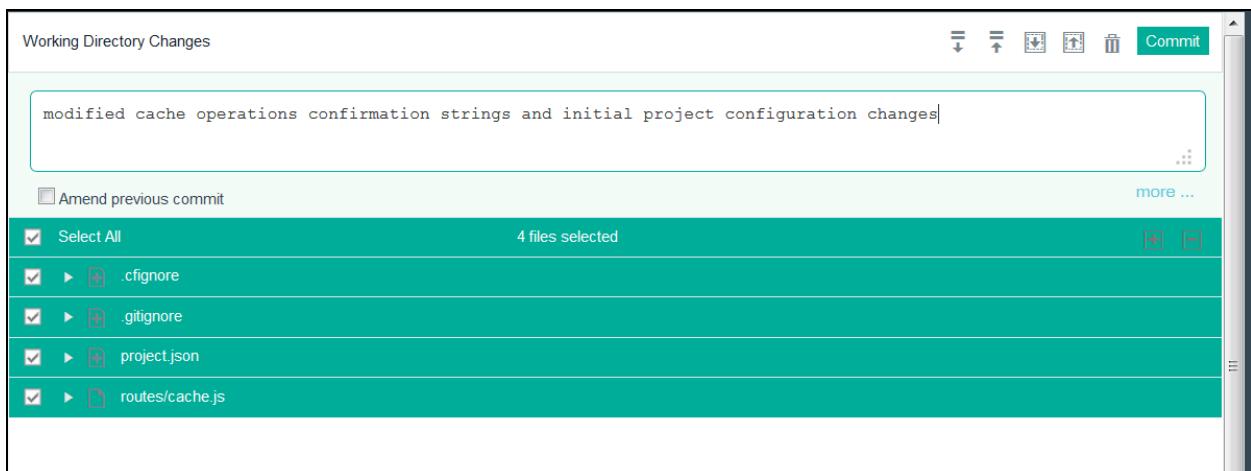
Sauvegarder le fichier.



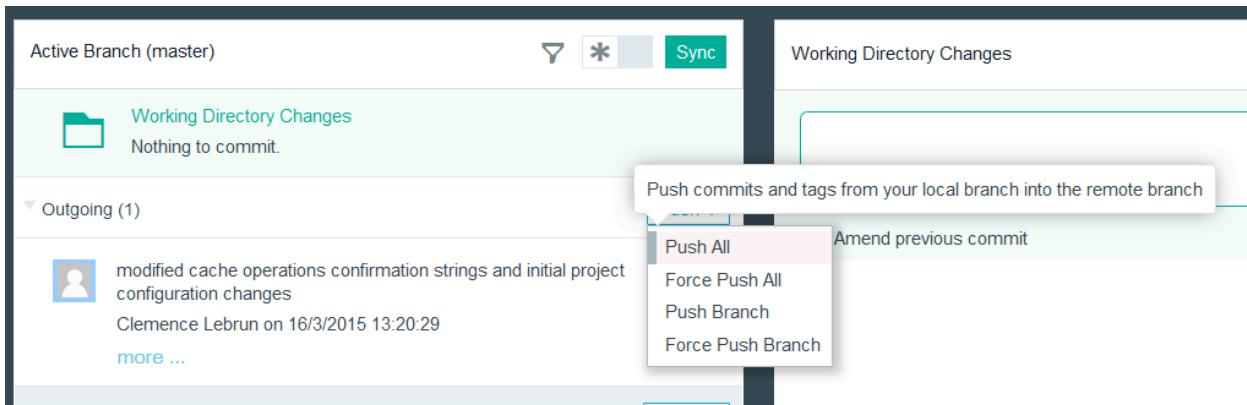
Puis aller sur la gestion du repository GIT (dans la barre de gauche)



Choisir 'Select all' puis 'Commit' en haut à droite afin d'accepter toutes les modifications.
Ne pas oublier le commentaire.



Puis pousser tous les changements en utilisant la flèche de Push puis 'Push All'.



En basculant rapidement vers ‘BUILD & DEPLOY’ en haut à droite, on pourra voir le deuxième déploiement.

Build Stage		Deploy Stage	
INPUT	Git URL	INPUT	View logs and history
Last commit by Philippe Thomas	2 min ago	Stage: Build Stage / Job: Build	Last Input: Build 2
JOB	View logs and history	JOB	View logs and history
JOBS COMPLETED SUCCESSFULLY		JOBS COMPLETED SUCCESSFULLY	
Build	Succeeded 2 min ago	Deploy to day	Not run
LAST EXECUTION RESULT		LAST EXECUTION RESULT	
Build 2	View runtime log	NodeJSDevOpsXX	NodeJSDevOpsXX.mybluemix.pvt.fr

Après un court temps de déploiement, on aura un message de déploiement réussi.

On pourra vérifier rapidement que le changement a été pris en compte en se connectant à l’application par l’URL qui se trouve en bas du Deploy Stage (lien en bleu).



Une fois dans l’application, ajouter des valeurs, modifier les et supprimer les afin de constater le changement dans les messages.

Grid Operations:

Key: test

Value: key

Stored it

Get **Put** **Delete**

Avant de supprimer l'application dans Bluemix, on pourra visualiser quelques fonctionnalités intéressantes de DevOps Services :

En retournant sur 'EDIT CODE', on ne manquera pas de noter l'éditeur de texte (assez proche de Sublime Text 2).

The screenshot shows the Bluemix DevOps Services interface with the 'Edit Code' tab selected. On the left, the project structure is displayed, including .git, launchConfigurations, public (with js and stylesheets), routes (cache.js, index.js), views, configignore, .gitignore, and app.js. The app.js file is currently open in the editor. The code in app.js is as follows:

```

1  /* jshint node:true */
2  /**
3   * Module dependencies.
4   */
5  var express = require('express'),
6      routes = require('./routes'),
7      http = require('http'),
8      path = require('path');
9
10 var app = express();
11
12 // all environments
13 app.set('port', process.env.PORT || 3000);
14 app.set('views', __dirname + '/views');
15 app.set('view engine', 'ejs');
16 app.use(express.static(path));
17 app.use(express.favicon());
18 app.use(express.logger('dev'));
19 app.use(express.bodyParser());
20 app.use(express.methodOverride());
21 app.use(app.router);
22 app.use(express.static(path.join(__dirname, 'public')));
23
24 // development only
25 if ('development' === app.get('env')) {
26   app.use(express.errorHandler());
27 }
28
29 app.get('/', routes.index);
30 app.get('/cache/:key', cache.getCache);
31 app.put("/cache", cache.putCache);
32 app.delete("/cache/:key", cache.removeCache);
33
34 http.createServer(app).listen(app.get('port'), function(){
35   console.log('Express server listening on port ' + app.get('port'));
36 });

```

En tapant un peu de code, on pourra remarquer le dispositif d'auto-complétion :

The screenshot shows the code editor with the cursor on the 'http' object in line 34. A dropdown menu displays several methods and properties available for the 'http' object, including 'all(path)', 'co', 'defaultConfiguration()', 'disable(setting)', 'disabled(setting)', 'enable(setting)', 'enabled(setting)', and 'engine(ext, fn)'. This demonstrates the integrated code completion feature.

Sur certains fichiers, il y a des warnings :

Exercice 1 - Mes premiers déploiements

A screenshot of the IBM DevOps Services interface. On the left, a file browser shows a project structure for 'phthom | NodeJSDevOpsXX 1'. The 'public' folder contains 'index.js', 'stylesheets', 'routes', and 'views'. The 'index.js' file is selected and shown in the main code editor area. The code editor highlights a syntax error at line 32: 'if (operation == "get") {'. A yellow warning icon is placed next to the '==' operator, indicating a typo where '===' was expected.

```
13
14     return null;
15
16
17     function sendRequest(operation) {
18         var key = document.getElementById('key').value;
19         if(key === ''){
20             document.getElementById('echo').innerHTML = 'Please input key.';
21             document.getElementById('key').focus();
22             return;
23         }
24         var value = document.getElementById('value').value;
25         document.getElementById('echo').innerHTML = '';
26
27         var xhr = createXHR();
28         xhr.onreadystatechange = function() {
29             if (xhr.readyState == 4) {
30                 var result = JSON.parse(xhr.responseText);
31                 var value = result.value;
32                 if (operation == "get") {
33                     document.getElementById('echo').innerHTML = "No entry is found.";
34                     document.getElementById('key').value = "";
35                     document.getElementById('value').value = "";
36                 } else if (operation == "set") {
37                     document.getElementById('echo').innerHTML = "Entry updated successfully.";
```

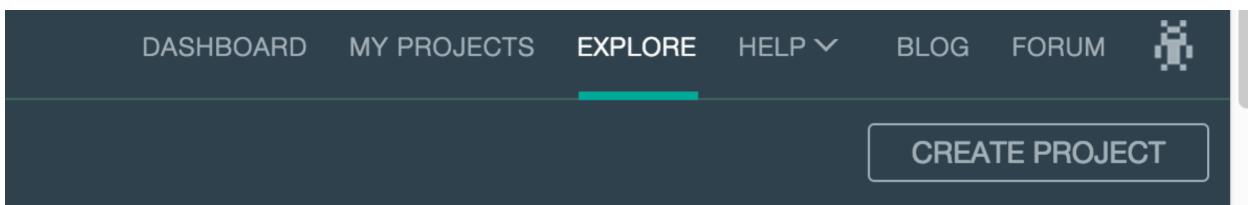
Ici par exemple, on s'attend à un '===' au lieu du '==' (glisser le curseur sur le triangle jaune).

Enfin, la possibilité de voir le fichier manifest.yml : on notera que celui-ci contient également la définition des services.

A screenshot of the IBM DevOps Services interface showing the 'manifest.yml' file content. The file defines an application with various configurations. It includes sections for 'applications', 'services', and specific host details like disk quota, host name, command, path, domain, instances, and memory.

```
1 applications:
2   - services:
3     - NodeJSDevOpsXX-DataCache
4     - NodeJSDevOpsXX-MonitoringAndAnalytics
5   disk_quota: 1024M
6   host: NodeJSDevOpsXX
7   name: NodeJSDevOpsXX
8   command: node app.js
9   path: .
10  domain: mybluemix.pvt
11  instances: 1
12  memory: 512M
```

En cliquant sur le menu 'EXPLORE' en haut à droite, vous pourrez parcourir tous les projets qui sont en cours d'élaboration dans DevOps Services.



On peut également faire des recherches sur ces projets :

Exercice 1 - Mes premiers déploiements

The screenshot shows the IBM Bluemix DevOps Services dashboard. On the left, a sidebar lists categories: ALL, ACTIVITY, EDITOR'S PICKS, POPULAR, and TWEETS. The main area displays a feed of items:

- OLIVER RODRIGUEZ | TwitterDemo**
OLIVER RODRIGUEZ created a public project a few seconds ago
- JazzHub** @JazzHub
New video featuring @Lauren_Schaefer! #DevOps Made Easy: IBM #Bluemix #DevOpsServices <http://t.co/eZOVAWCqbD>
13 minutes ago
- antomm | robot-scanner**
Have you heard the buzz about iBeacons? iBeacons are Bluetooth low-energy devices that can interact with mobile devices. This iOS robot app uses [ibeacon technology with Bluemix to create a robust guidance system](#) [Open](#)

A sidebar on the right is titled "DEVELOPERWEEK CONFERENCE + FESTIVAL 2015" and features the "IBM® DevOps Services" logo with the subtitle "IBM Coding Environment".

On supprimera finalement cette application dans Bluemix afin d'avoir un peu plus de mémoire pour le prochain exercice.