CSCI 4140 – Tutorial 8

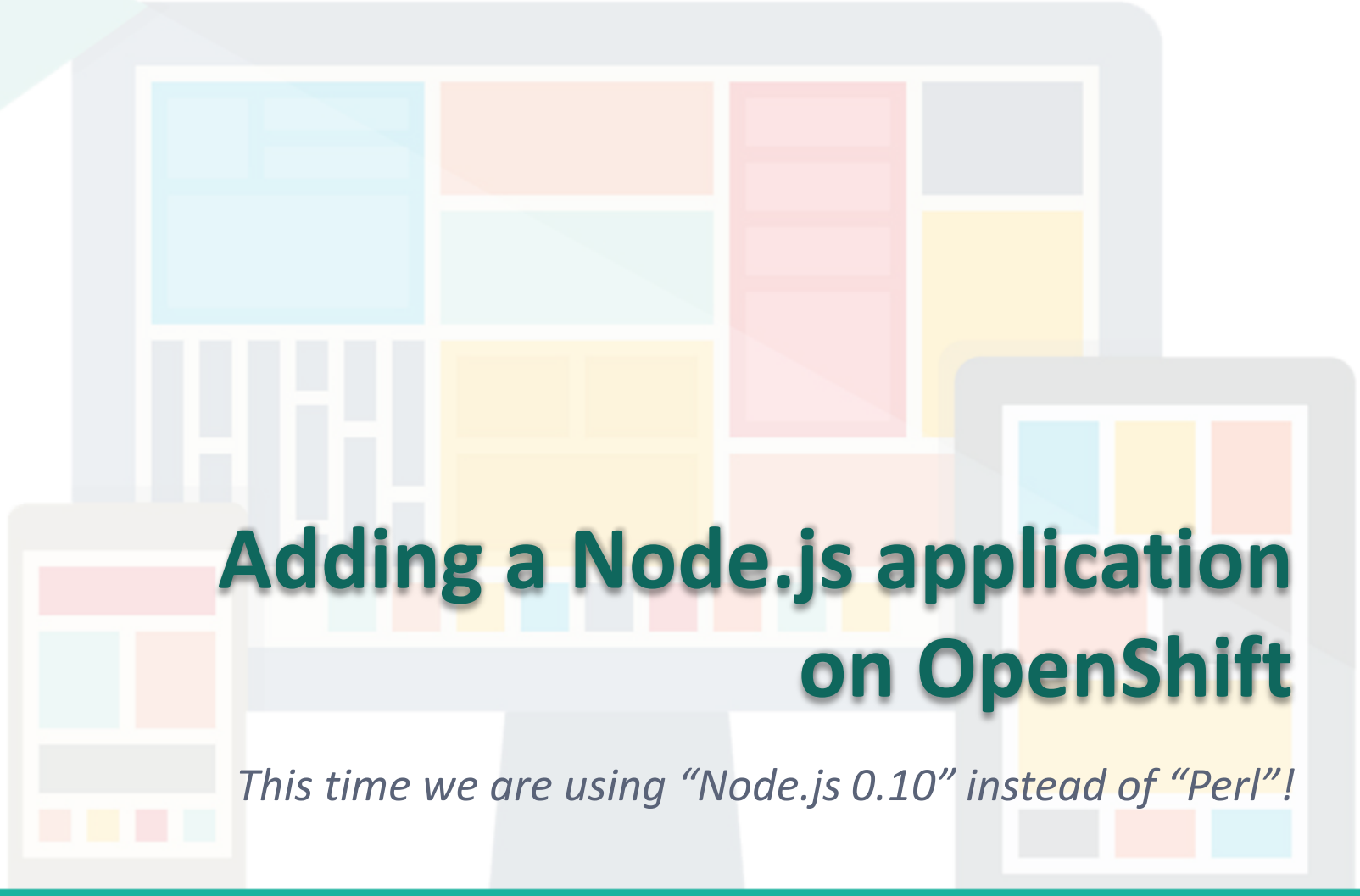
Deploying Node.js Applications on OpenShift

Matt YIU, Man Tung (mtyiu@cse)
SHB 118
Office Hour: Tuesday, 3-5 pm

2015.03.12

Prerequisite

- We will start with an **Express application**
  - Please follow the instructions on the tutorial slides "**Installing Node.js and Express on [Windows|Linux or Mac]**", pp. 19-21 for creating an application skeleton
  - If you don't use an Express application, you need to figure out where to configure the server's listen IP address and port number
- We will deploy the application on **OpenShift**
  - You should have an account already ☺
  - I assume that you finished all configurations for using OpenShift (e.g., adding SSH keys)

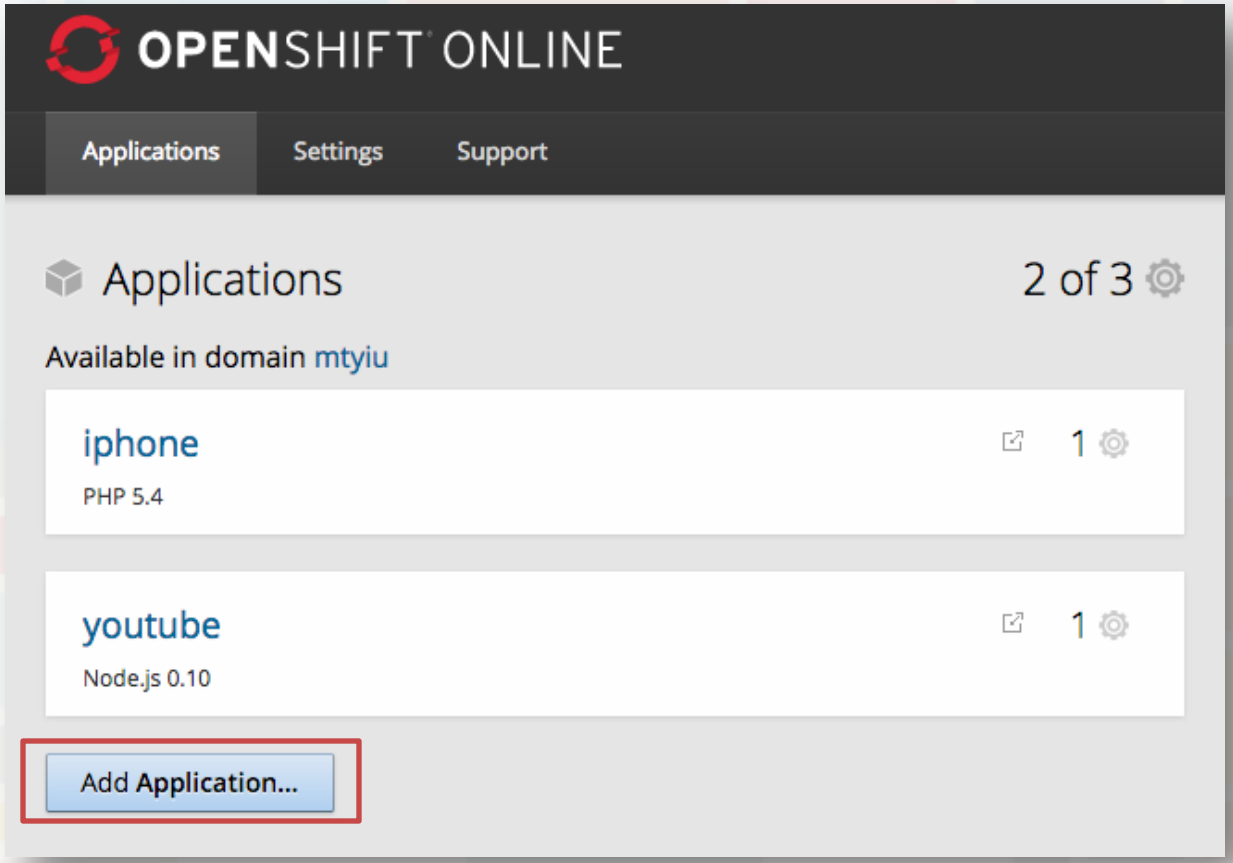


Adding a Node.js application on OpenShift

This time we are using "Node.js 0.10" instead of "Perl"!

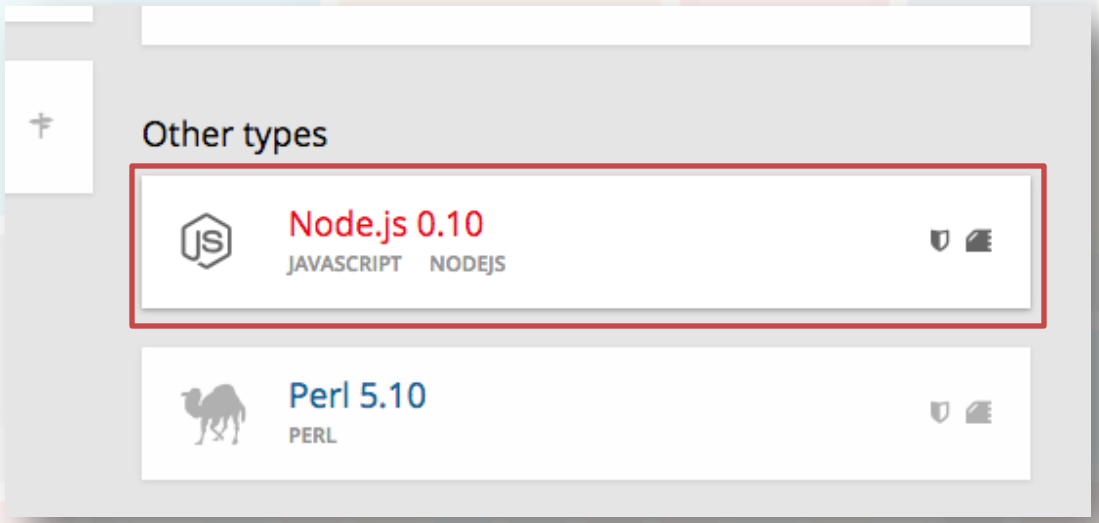
Step 1. Add Application

- Login to your OpenShift console and click "**Add Application…**"



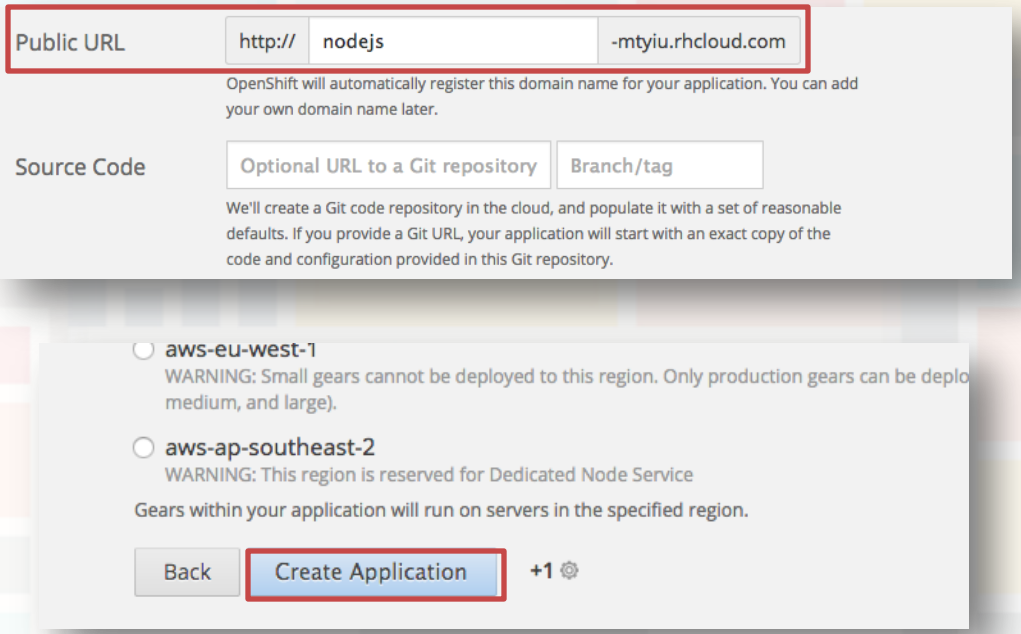
Step 2. Choose a type of application

- Select "**Node.js 0.10**"



Step 3. Configure the application

- Remember to change the **public URL** of your application
- Keep default settings for other configurations
- Click "**Create Application**"



Warning: We focus on adapting Express applications in the following steps!

Adapting existing Node.js applications to run on OpenShift

Forget about the sample code provided by OpenShift!

Step 1. Include a package.json file

- All Node.js applications should include a **package.json** file in the **root of their project**
  - Since we are using **Express application generator** to create our application skeleton, this file is automatically generated
- Remember to change the path of the startup file
  - Note that I don't use the default script (**bin/www**)
  - Let's change it to "**server.js**"

```
{
  "name": "nodejs-openshift",
  "version": "0.0.0",
  "private": true,
  "scripts": {
    "start": "node server.js"
  },
  "dependencies": {
    "express": "~4.10.6",
    "body-parser": "~1.10.1",
    "cookie-parser": "~1.3.3",
    "morgan": "~1.5.1",
    "serve-favicon": "~2.2.0",
    "debug": "~2.1.1",
    "jade": "~1.8.2"
  }
}
```

Sample package.json file

Step 2. Edit the startup script (`app.js`)

- OpenShift's Node.js cartridge automatically publishes **server connection information** to your application's environment via the following environment variables:
  - `OPENSHIFT_NODEJS_PORT`
  - `OPENSHIFT_NODEJS_IP`
- The startup script should read configuration details from the system environment
- Now edit the startup script…

Step 2. Edit the startup script (app.js)

```
var express = require( 'express' );
var app = express();

var server_port = process.env.OPENSHIFT_NODEJS_PORT || 8000;
var server_ip_address = process.env.OPENSHIFT_NODEJS_IP || '127.0.0.1';

app.get( '/', function ( req, res ) {
        res.send( 'Hello World!' );
} );

var server = app.listen( server_port, server_ip_address, function () {
        var host = server.address().address;
        var port = server.address().port;
        console.log( 'Listening at http://%s:%s', host, port );
} );
```

openshift/server.js

Step 2. Edit the startup script (app.js)

```
var express = require( 'express' );
var app = express();

var server_port = process.env.OPENSHIFT_NODEJS_PORT || 8000;
var server_ip_address = process.env.OPENSHIFT_NODEJS_IP || '127.0.0.1';

app.get(
        r
} );

var server
        v
        v
        console.log( 'Listening at http://%s:%s', host, port );
} );
```

Store the IP address and port number in variables.
It first tries to read the environment variables. If they do not exist, use "8000" as the port number and "127.0.0.1" as the IP address.
This is important to ensure your project's portability!
Since your local machine does not have these two environment variables, "127.0.0.1:8000" will be used for your development environment.

openshift/server.js

Step 2. Edit the startup script (app.js)

```
var express = require( 'express' );
var app = express();

var server_port = process.env.OPENSHIFT_NODEJS_PORT || 8000;
var server_ip_address = process.env.OPENSHIFT_NODEJS_IP || '127.0.0.1';

app.get( '/', function ( req, res ) {
        res.send( 'Hello World!' )
} );

var server = app.listen( server_port, server_ip_address, function () {
        var host = server.address().address;
        var port = server.address().port;
        console.log( 'Listening at http://%s:%s', host, port );
} );
```

Configure the server to listen at "<server_ip_address>:<server_port>".

openshift/server.js

Step 2. Edit the startup script (app.js)

```
var express = require( 'express' );
var app = express();

var server_port = process.env.OPENSHIFT_NODEJS_PORT || 8000;
var server_ip_address = process.env.OPENSHIFT_NODEJS_IP || '127.0.0.1';

app.get( '/', function ( req, res ) {
        res.send( 'Hello World!' );
} );

var server = app.listen( server_port, server_ip_address, function () {
        var host = server.address().address;
        var port = server.address().port;
        console.log( 'Listening at http://%s:%s', host, port );
} );
```

openshift/server.js

Reminder: This application does not involve Socket.IO configuration for simplicity! Follow the instructions on the tutorial notes for using Socket.IO.

Step 3. `git commit` and push to OpenShift

- Now back to the root directory of your application
- In case you did not create the Git repository…

```
$ git init
Initialized empty Git repository in
/Users/mtyiu/Development/nodejs-openshift/.git/
$ git add .
```

- Commit your code changes:

```
$ git commit -a -m "<Your commit message>"
```

- Add OpenShift to the remote of the Git repository:

```
$ git remote add origin
ssh://abcdefghijklmnopqrstuvwx@nodejs-
mtyiu.rhcloud.com/~/git/nod
```

Find the remote URL on your OpenShift console.

Step 3. `git commit` and push to OpenShift

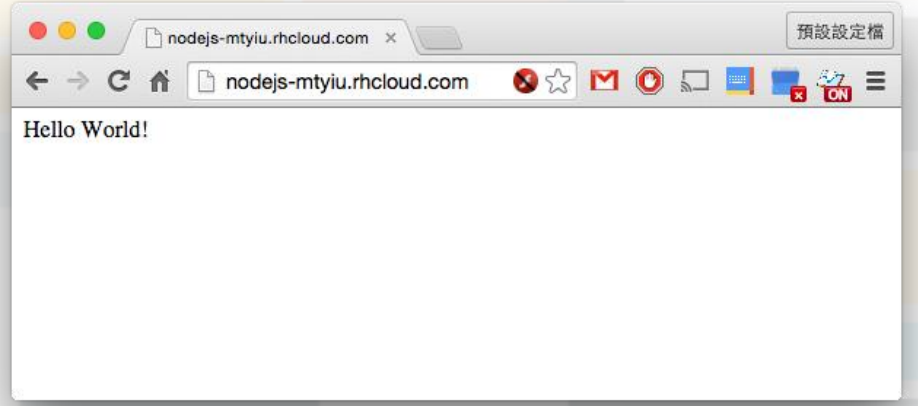
- We are ready to push the code to OpenShift:

```
$ git push -f origin
```

Do you notice the "`-f`" flag? It forces a commit on a remote ref.

Since OpenShift provides a sample Node.js application on the repository, you will not be able to commit your code changes without this flag.

- When it is done, you can visit your website using the public URL you set before:



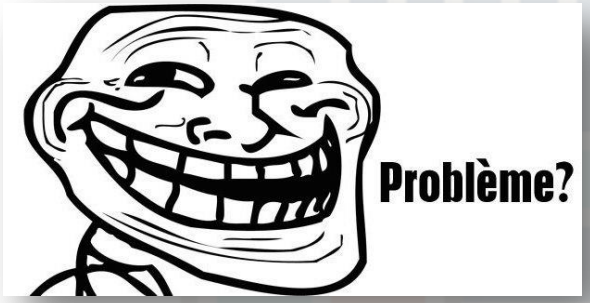
Working with database?

- **Warning:** It is not necessary to use database for Assignment 2!
- In case you want to use it in your project…
  - Node.js works best with **MongoDB**
    - Read https://blog.openshift.com/run-your-nodejs-projects-on-openshift-in-two-simple-steps/ for more details
  - It may also work with MySQL (though I didn't try)
    - Google yourself ☺
    - Remember to use environment variables to get the MySQL configuration strings

Add action hook script?

- If you need to run a script during deployment, you can follow the instructions on p. 26, Tutorial 1 by Jimmy SINN
  - http://appsrv.cse.cuhk.edu.hk/~ltsinn/csci4140-2015spring/tutorial1_openshift.pdf

Useful reference:
Node.js Application Hosting –
https://developers.openshift.com/en/node-js-overview.html



Problem?

Say "Hi Man Tung" to find me in Facebook group!