INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY

# Visual Workflows For Openshift

User Guide

**Sandeep Panem**
**Subho Banerjee**

**Sanchit Aggarwal**
**Vijayendra Grampurohit**

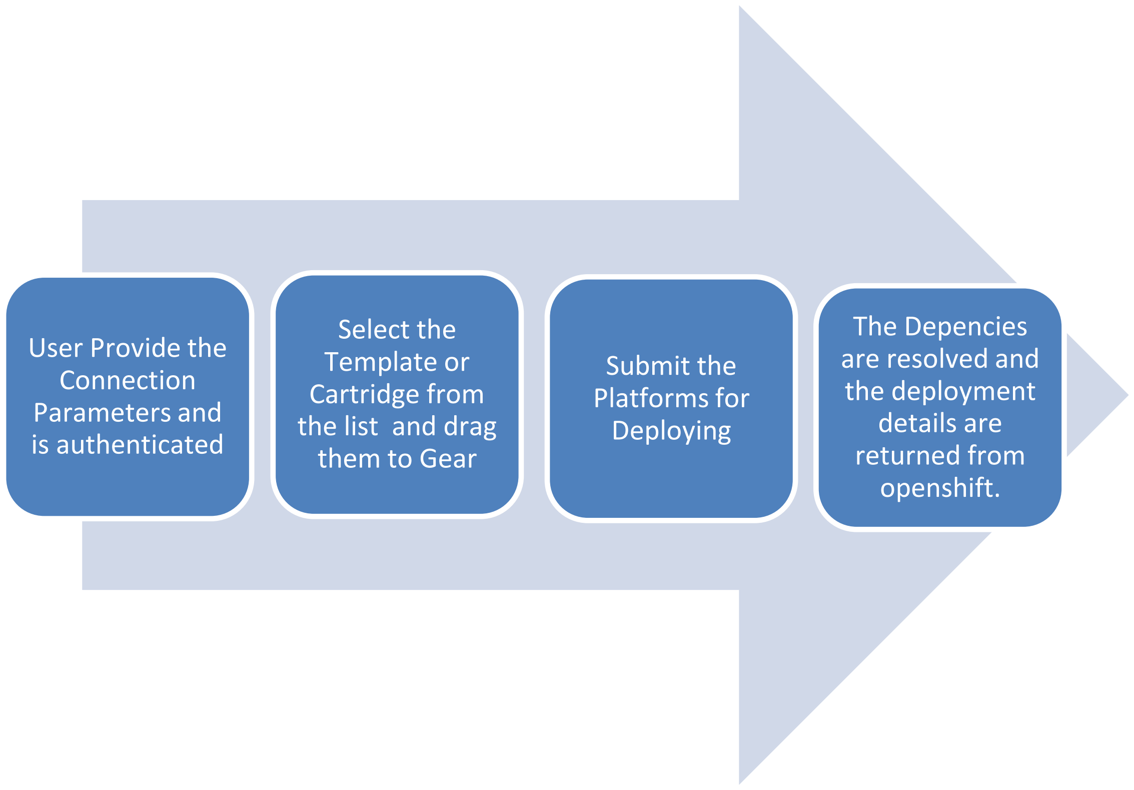
This guide contains information about the client for the OpenShift which provides an interface to the Broker-API for simple deployments as well as more complicated workflows.

## Table of Contents

## 1. Project Overview

The project's objective is to identify enterprise work-flows for deploying applications to Openshift and build a client which provides an interface to the Broker-API for simple deployments as well as more complicated workflows using:

1. Keystone authentication
2. Openshift REST API
3. Abstract interface to create your own workflows


User Provide the Connection Parameters and is authenticated
Select the Template or Cartridge from the list and drag them to Gear
Submit the Platforms for Deploying
The Depencies are resolved and the deployment details are returned from openshift.

**The above Diagram summarizes the steps followed to deploy the platform using the portal.**

## 2. Functional Requirements

To develop a portal for Openshift visual workflows with Drag and Drop features to select the cartridges and application platforms with respect to the business needs.

### 2.1 REQUIREMENTS IN SCOPE

- To automate the process of installation and deploying the platforms on the fly in cloud environment without being concerned about the order of the dependencies.
- To remove the redundancy while developing, deploying and scaling the applications manually across multiple nodes by the developers.

- To automate the process of installing, resolving the dependencies between applications, developing and deploying applications for users.
- For example for developing form processing application we need a front end with HTML/ JavaScript and Oracle/Mongo Db. as database.
- To simplify the process of installing, resolving dependencies and deploying the application for user.

### 2.2 FEATURES

Some Salient features of the Visual workflow for Openshift are:

- ***Frontend client:***

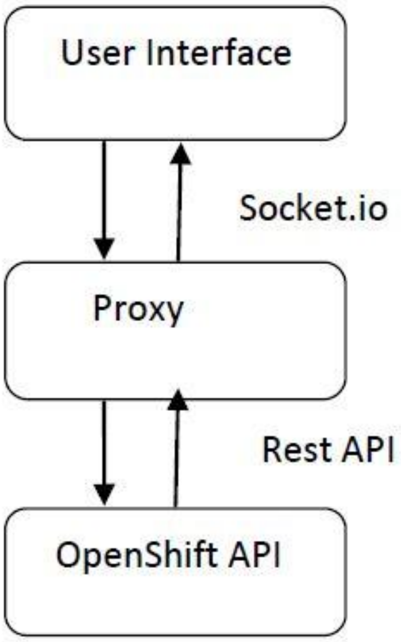
1. Designing an interface for connecting to any Openshift server.
2. Drag and Drop for selecting the cartridges and templates.
3. Ability for the user to design simple workflows using multiple API (Openshift/REST) calls

- ***Middleware (proxy):***

4. Forwarding API calls from client to Openshift broker (to avoid CORS)

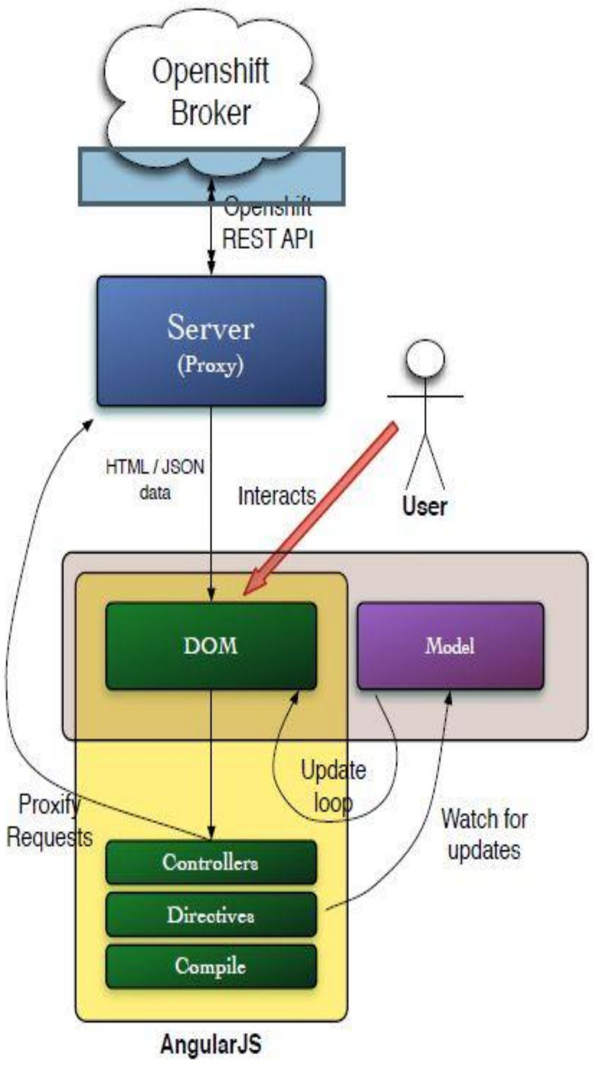
## 3. High Level Design

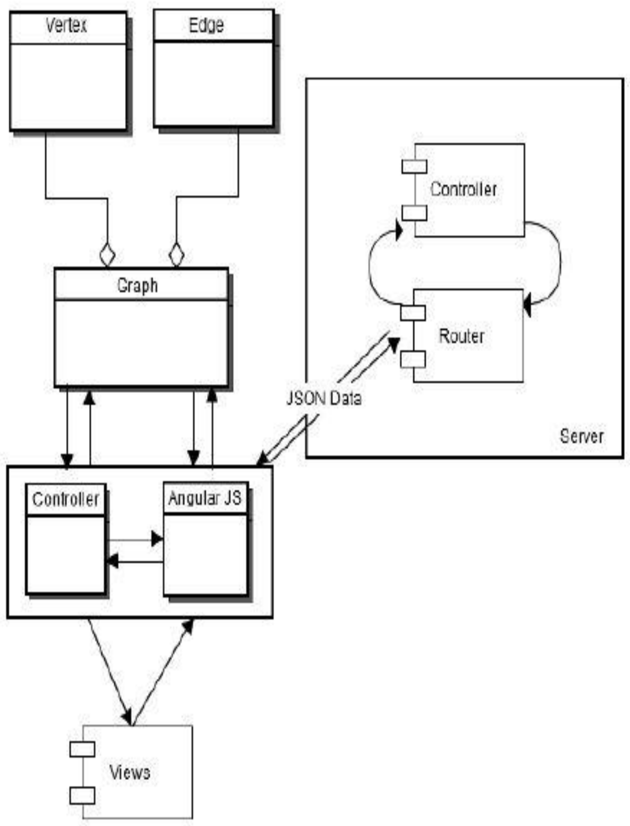
The following diagram depicts the high level design of the application.



## 4. Architecture

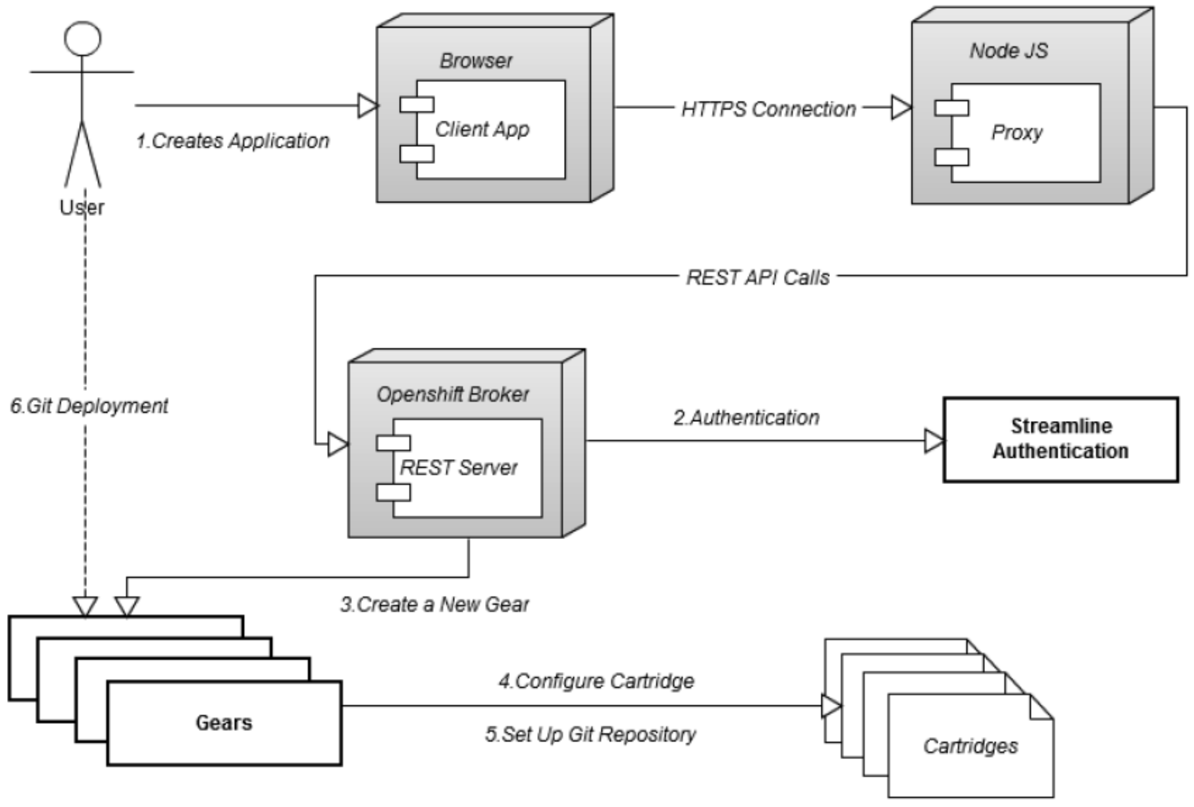
The basic description of the application architecture is depicted by the following figure:





## 5. User Interactions

The following diagram depicts the main interaction between the user and the application.



## 6. Code Structure

The Application has the two main components Frontend Client and the Middleware (proxy).The code structure for both of them is:

### 6.1 Frontend Client Location: /public/js/

The Frontend Client is the interface which is a result of number of modules called by the **index.html**. Theses main modules of the frontend client are:

1) **UI.JS**: This module sets all the elements and features of the application. The main functions are:

- **FUNCTION: setError**

| | |
|---|---|
| Parameters | text |
| Return Values | None |
| Description | This function displays the error. |
| Calling Event | Index.html #errorplaceholder |
| Functions called by this function | None |

- **FUNCTION: showcartridges**

| | |
|---|---|
| Parameters | document |
| Return Values | None |
| Description | It generates the list of cartridges |
| Calling Event | None |
| Functions called by this function | None |

- **FUNCTION jsPlumb.draggable**

| | |
|---|---|
| Parameters | node |
| Return Values | None |
| Description | It makes the node drag gable |
| Calling Event | None |
| Functions called by this function | None |

2) **GRAPH.JS**: This module defines the functions used for making graph used in the application. It uses the JSplumb jquery to render graph vertex and node. The main functions are:

- **FUNCTION: Vertex**

| | |
|---|---|
| Parameters | div_id |
| Return Values | None |
| Description | It defines the vertex and its endpoints using jsPlummb.addEndpoint. |
| Calling Event | Graph |
| Functions called by this function | None |

- **FUNCTION: Edge**

| | |
|---|---|
| Parameters | S,t |
| Return Values | None |
| Description | It creates the edge between two vertexes (source and target) in the graph. |
| Calling Event | Graph |
| Functions called by this function | None |

- **FUNCTION: Graph**

| | |
|---|---|
| Parameters | None |
| Return Values | Vertex and Edge |
| Description | It renders the graph by calling vertex and edges. |
| Calling Event | Index.html div drawing board |
| Functions called by this function | Vertex, Edge |

3) **PAGE.JS**: This module defines the functions used for controlling the web application. It uses the Angular JS. All parameters related to connection and graph is defined in this module. The main functions are:

- **FUNCTION: App**

| | |
|---|---|
| Parameters | $scope,$http |
| Return Values | None |
| Description | This function calls internal function for controlling the elements like spinner, proxify |
| Calling Event | Index.html |

| | |
|---|---|
| Functions called by this function | Busy, proxify, errorcallback, errorcallbackfor deploy, submit, addnode, removenode, cleargraph, dragCartfrombar,acceptTokenInSubnode, CommitToken,,DeleteCartridge,deploy |

- **FUNCTION: Busy**

| | |
|---|---|
| Parameters | None |
| Return Values | None |
| Description | Show a spinner to indicate busy status |
| Calling Event | App |
| Functions called by this function | None |

- **FUNCTION: proxify**

| | |
|---|---|
| Parameters | options, successCallback, failureCallback |
| Return Values | None |
| Description | It Make call to the openshift broker |
| Calling Event | App |
| Functions called by this function | None |

- **FUNCTION: errorCallback**

| | |
|---|---|
| Parameters | data, status, headers, config |
| Return Values | None |
| Description | Generic call back to set error for all requests |
| Calling Event | App |
| Functions called by this function | None |

- **FUNCTION: errorCallbackForDeploy**

| | |
|---|---|
| Parameters | data, status, headers, config |
| Return Values | None |
| Description | Call back to set error for deployment requests |
| Calling Event | App |
| Functions called by this function | None |

- **FUNCTION: scope.submit**

| | |
|---|---|
| Parameters | data, status, headers, config |
| Return Values | None |
| Description | Authenticate user and get the list of cartridges and templates |
| Calling Event | App |
| Functions called by this function | None |

- **FUNCTION: scope.addnode**

| | |
|---|---|
| Parameters | ident |
| Return Values | None |
| Description | Add a node to the Graph |
| Calling Event | App |
| Functions called by this function | None |

- **FUNCTION: scope.addnode**

| | |
|---|---|
| Parameters | ident |
| Return Values | None |

| | |
|---|---|
| Description | Add a node to the Graph with given identity value |
| Calling Event | App |
| Functions called by this function | None |

- **FUNCTION: scope.removenode**

| | |
|---|---|
| Parameters | ident |
| Return Values | None |
| Description | Remove a node from the Graph |
| Calling Event | App |
| Functions called by this function | None |

- **FUNCTION: scope.cleargraph**

| | |
|---|---|
| Parameters | None |
| Return Values | None |
| Description | Delete the graph completely |
| Calling Event | App |
| Functions called by this function | None |

- **FUNCTION: scope.dragCartFromBar**

| | |
|---|---|
| Parameters | item, list |
| Return Values | src: list, item: item |
| Description | It Start the drag event for dragging object from cartridge list |
| Calling Event | App |
| Functions called by this function | None |

- **FUNCTION: scope.acceptTokenInSubnode**

| | |
|---|---|
| Parameters | targetArray, token |
| Return Values | None |
| Description | It Checks if drag target is acceptable |
| Calling Event | App |
| Functions called by this function | None |

- **FUNCTION: scope.commitTokenInSubnode**

| | |
|---|---|
| Parameters | to, token |
| Return Values | None |
| Description | It  Adds cartridge to vertex |
| Calling Event | App |
| Functions called by this function | None |

- **FUNCTION: scope.deleteCartridge**

| | |
|---|---|
| Parameters | cartridge, vertex |
| Return Values | None |
| Description | It  Deletes cartridge from vertex |
| Calling Event | App |
| Functions called by this function | None |

- **FUNCTION: scope.deploy**

| | |
|---|---|
| Parameters | None |
| Return Values | None |

| | |
|---|---|
| Description | It  Deploy the graph to a openshift broker |
| Calling Event | App |
| Functions called by this function | None |

## 6.2 Middleware (Proxy)           Location: /Routes/proxy.js

The Middleware forwards the request from the frontend Client interface to the **Openshift Broker.** The main module of the middleware is:

2) **PROXY.JS**: This module enables the Cross-origin Resource sharing (**CORS)** between frontend client and Openshift broker. CORS is a mechanism that allows a web page to make XMLHttpRequests to another domain. The main functions is:
   - **FUNCTION: exports.proxify**

| | |
|---|---|
| Parameters | req, res |
| Return Values | None |
| Description | This function forward the request from client to Openshift broker |
| Calling Event | http request |
| Functions called by this function | None |

## 6.3 Configuration Files

This application was built to be used with the FreeShift or MegaShift service provided on the **Red Hat Cloud**. However, it can be used with any deployment of the **Openshift Origin**. If however, your deployment of Openshift is tweaked to include non-standard cartridges and templates, you will be required to make changes in the following configuration files –

1) **public/conf/images.json –**
   a. This file controls the images that are used to display cartridges on the screen.

```
{
    "Node.js 0.6": "/img/icons/nodejs.png",
    "Zend Server 5.6": "/img/icons/zend.png",
    "Ruby 1.9": "/img/icons/ruby.png",
    "JBoss Application Server 7.1": "/img/icons/jboss_as7.png",
    "Python 2.6": "/img/icons/python.png",
    "Jenkins Server 1.4": "/img/icons/jenkins.png",
    "Ruby 1.8": "/img/icons/ruby.png",
    "JBoss Enterprise Application Platform 6.0":"/img/icons/jboss.png",
    "PHP 5.3": "/img/icons/php.png"
}
```

b. The file stores a hash with the name of the cartridge as key and the location of the corresponding image as value.
c. The images that are shipped with the code are placed in the /public/img/icons folder. However, you may give a link to a generic image on the internet.

**2) public/conf/templates.json –**

a. This file controls the images that are used to display templates on the screen.

```
{
    "CakePHP": "/img/icons/cakephp.png",
    "CakePHP (TEST)": "/img/icons/cakephp_Test.png",
    "Django": "/img/icons/django.png",
    "Django (Test)": "/img/icons/django_Test.png"
}
```

b. The file stores a hash with the name of the template as key and the location of the corresponding image as value.
c. The images that are shipped with the code are placed in the **/public/img/icons** folder. However, you may give a link to a generic image on the internet.

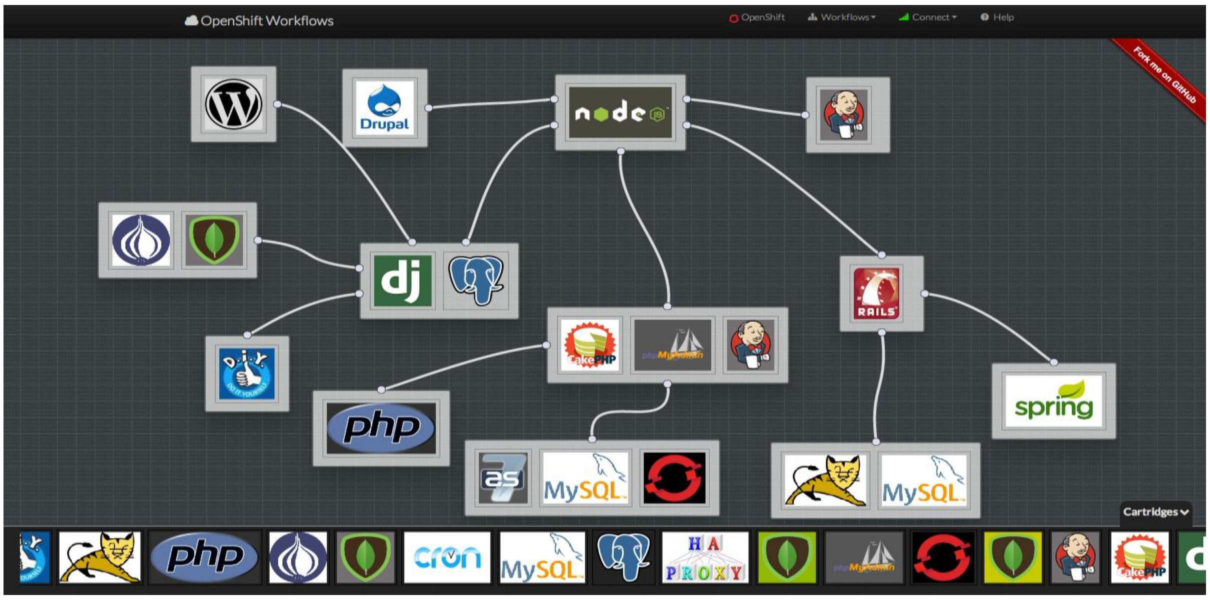
**3) public/conf/rules.json –**

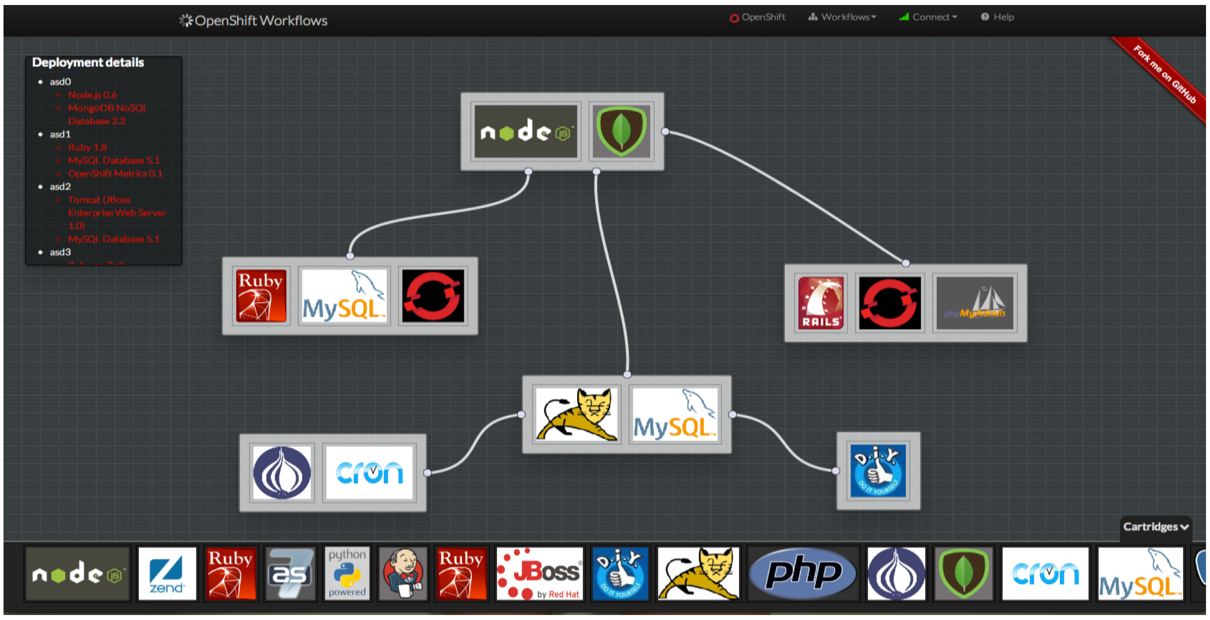
a. This file defines dependencies between cartridges and templates, so that a semantically correct workflow can be defined.

```
{
    "cartridge": {
        "rockmongo-1.1": ["mongodb-2.2"],
        "10gen-mms-agent-0.1": ["mongodb-2.2"],
        "phpmyadmin-3.4": ["mysql-5.1"]
    },
    "template": {
        "cakephp": ["mysql-5.1"],
        "drupal": ["mysql-5.1"],
        "rails": ["mysql-5.1"],
        "wordpress": ["mysql-5.1"]
    }
}
```

b. This configuration file has two parts, the cartridge hash, which stores the dependencies of cartridges, ie."Rockmongo-1.1": ["mongodb-2.2"] means that RockMongo cannot be installed without MongoDB installed first.
c. The second part of this configuration file, the template hash, stores the embedded cartridges that are installed along with the template, ie."cakephp": ["mysql-5.1"] means that CakePHP installs MySQL as one of its embedded cartridges.

## 7. User Interface





Deployment details
- asd0
  - Node.js 0.6
  - MongoDB NoSQL Database 2.2
- asd1
  - Ruby 1.8
  - MySQL Database 5.1
  - OpenShift Metrics 0.1
- asd2
  - Tomcat (JBoss Enterprise Web Server 1.0)
  - MySQL Database 5.1
- asd3

## 8. Error and Problems

- Resolving the issue of "cross-domain" requests between the frontend client and the Openshift broker which was forbidden by web browsers because of the same origin security policy, by making **a proxify function** to enable CORS.
- Issue of not allowing addition of duplicate cartridges on the same gear to be installed.
- Resolving the issue of not showing the application data if application fails.
- Issue of deleting the application on failed deployment.
- Issue of setting timeout for adding cartridges.
- Resolving the error of displaying deployment data.
- Issue of adding rule so that empty nodes cannot be deployed
- Resolving the issue of dependencies for templates
- Addition of rules for template addition to graph while deployment of templates
- Resolving the issue of deleting a node on screen resetting the graph.
- To check all dependencies when cartridges are added
- To do the basic validation of dependencies in graphs.
- Issue of deploying multiple cartridges to a gear.

## 9. Installation Steps

1) **Get the latest source code**

```
$ git clone https://github.com/subszero/openshift-workflows.git
```

2) **Install NodeJS and NPM from here**
3) **Install all dependencies**

```
$ npm install
```

4) **Generate new SSL certificates in the certs/ folder**
5) **Run the server and open the page in a browser**

```
$ node app.js -b
```

## 10. Future Work

- We would like to extend the application to managing deployed apps as well. Particularly, setting auto-scale parameters and controlling the lifecycle.

- We would also like to add a mechanism by which the user can directly deploy his code using our portal rather than having to synchronize Git repositories manually.

## 11. References

### 11.1 Openshift

- Openshift Architecture Specification
- Openshift User Guide
- Openshift REST API Guide

### 11.2 Technologies used in the backend

- NodeJS
- ExpressJS ( + Embedded JavaScript )
- RequestJS
- CommanderJS

### 11.3 Technologies used in the frontend

- HTML5 Rocks
- jQuery ( + jQuery-UI )
- Bootstrap
- modernizr
- AngularJS ( + Angular-UI + Angular-jQuery-UI )
- jsPlumb
- SpinJS