# Integrate Cloud Platforms in My Enterprise

## Table of Contents

# 1 Introduction

## 1.1 Lab Goals

During this lab you will learn to use Python Flask and Bootstrap Framework to custom portal that will serve as an integration point for external vendors in form of SoftLayer and BlueBox with and internal ERP in form of the OpenSource product Odoo (formerly known as OpenERP).

The lab will show you step by step how to utilise the relevant APIs to achieve this integration.

## 1.2 Some instructions

VM operating system: Ubuntu 14.04.03
user: ibmcloud
passwd: IBM_Cloud_2016

Download this document to copy/paste the code from:
https://ibm.biz/lab3247

You can find the finalize code on the folder “/home/ibmcloud/dashboard/”, in case you want to check it and compare with your progress.

**The text on this format, means code to add to the files**

**The text on this format means code to run on the command line**

The text on this format means important information

## 1.3 Components

- Odoo. Open source ERP. Installation instructions can be found here. The lab image has Odoo pre-installed and configured with sample data, and the server can be accessed by web browser on http://localhost:8069
- Bootstrap template. The original one can be download from here , it is released under MIT license. We will use a simplified version for this lab. That can be found on “/home/ibmcloud/lab_template/” or you can download from here.
- Python
- Python/ Flask
- Pythong binding for Softlayer API
- Python binding for OpenStack API
- BlueMix Containers RestFul API

Our preferred IDE for this Lab will Geany, yes! Geany :)

# 2 The code

## 2.1 Create the python server

In order to create the python server we will use “flask” extension for python (already installed on the VM).

- Open Geany
- Create a new file

Add the needed libraries for Flask server:

```
import os #, sys
from flask import Flask, session, render_template, request
from flask.ext.session import Session
```

Add the code for the Flask session (don't really needed in this lab)

```
app = Flask(__name__)
# Check Configuration section for more details
SESSION_TYPE = 'filesystem'
app.config.from_object(__name__)
Session(app)
```

Add the code for server initialization

```
port = os.getenv('PORT', '5000')
if __name__ == "__main__":
	app.run(host='0.0.0.0', port=int(port), threaded=True, debug=True)
```

In order to add the main site (“index.html”, on the root), add this above the previous chunk of code, before the line “port = os.getenv('PORT', '5000')”:

```
@app.route('/' )
def root():
   return render_template("index.html", title = 'Projects')
```

The code for the “server.py” should look like this (the working folder to save the “server.py” is “/home/ibmcloud/lab/”):

```
## Needed Libraries

import os
from flask import Flask, session, render_template, request
from flask.ext.session import Session



## Code for the session
```

```
app = Flask(__name__)
SESSION_TYPE = 'filesystem'
app.config.from_object(__name__)
Session(app)

## Set the root folder

@app.route('/' )
def root():
   return render_template("index.html", title = 'Dashboard')

## Where the server will listen, and debug options

port = os.getenv('PORT', '5000')
if __name__ == "__main__":
        app.run(host='0.0.0.0', port=int(port), threaded=True, debug=True)
```

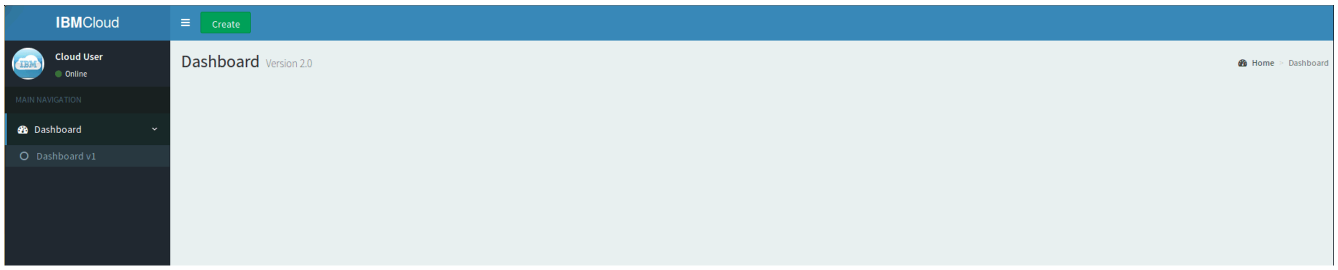
Copy the bootstrap template to the right directories, “static” folder for the static content, and “template” folder to host the “index.html”:

```
mkdir /home/ibmcloud/lab/templates
cp /home/ibmcloud/lab_template/index.html /home/ibmcloud/lab/templates/
cp -r /home/ibmcloud/lab_template/static /home/ibmcloud/lab/
```

On the working folder run:

```
python server.py
```

Check on Firefox the template is loaded, URL: [http://localhost:5000](http://localhost:5000), you should see something like this:



## 2.2 List servers and containers

As we are going to work with 3 different platforms on this lab we will create 3 libraries, to interact which each provider API we create 1 file per provider.

** Ask for the provider credentials to the lab hosts
***If we need to see any respond from the API we just need to add the line "pp(variable_with_the_respond)"

- Create a folder with the name "softlayer" and a file with Geany with the name "cci.py".
- Add the SoftLayer python binding, and "pprint" in case we want to see some answers from the API:

```
import SoftLayer
from pprint import pprint as pp
```

- Add the lines with the user&passwd provided and initialize the client:

```
user='your_user';
apikey='your_passwd';
client = SoftLayer.Client(username=user, api_key=apikey)
```

- Create a function to list the CCI in the account based on the service "Account" and the method "getVirtualGuests" , that will return this datatype

```
def getCCIs():
        ##Mask to get the right data that we will use
        object_mask = 'id,fullyQualifiedDomainName,status'
        result = client['Account'].getVirtualGuests(mask=object_mask)
        return result
```

- Create a folder with the name "openstack" and a file with Geany with the name "vm.py".
- Add the Nova OpenStack python binding,  and "pprint" in case we want to see some answers from the API:

```
from novaclient import client
from pprint import pprint as pp
```

- Initialize the client with the account details provided:

```
nova = client.Client(2,"your_user", "your_passwd", "your_tenant", "https://icos-sea.openstack.blueboxgrid.com:5001/v2.0", region_name="RegionOne", service_type="compute")
```

- Create the function to list the VMs, calling to the method list in the class Servers:

```
def getServers():
        list_servers=nova.servers.list()
        servers=[]
```

```
    for server in list_servers:
        ##we create a json for the respond
        servers.append({'id':server.id,"hostname":server.name,"status":server.status})
    return servers
```

- Create a folder with the name “docker” and a file with Geany with the name “containers.py”.
- Add the “request” library to perform the Rest request to the BlueMix Containers API,  and “pprint” in case we want to see some answers from the API:

```
import requests
from pprint import pprint as pp
```

- Add your account credentials for your BlueMix account. If you don't have any, create one for free here:

*** To find out the guid of your BlueMix account/space account, execute this on the command line

```
cf login
cf curl /v2/organizations
```

it will return something like this:

```
{
      "metadata": {
        "guid": "6f2e2826-xxxx-xxxx-xxxx-7318718261fc",
        "url": "/v2/organizations/6f2e2826-xxxx-xxxx-xxxx-7318718261fc",
        "created_at": "2015-04-22T10:48:16Z",
        "updated_at": "2015-04-22T11:04:23Z"
      },
      "entity": {
        "name": "jesus.arteche@ie.ibm.com",
        "billing_enabled": false,
        "quota_definition_guid": "d8787d01-xxxx-xxxx-xxxx-ee2576137e19",
        "status": "active",
        "quota_definition_url": "/v2/quota_definitions/d8787d01-xxxx-xxxx-xxxx-ee2576137e19",
        "spaces_url": "/v2/organizations/6f2e2826-xxxx-xxxx-xxxx-7318718261fc/spaces",
        "domains_url": "/v2/organizations/6f2e2826-xxxx-xxxx-xxxx-7318718261fc/domains",
        "private_domains_url": "/v2/organizations/6f2e2826-xxxx-xxxx-xxxx-
7318718261fc/private_domains",
        "users_url": "/v2/organizations/6f2e2826-xxxx-xxxx-xxxx-7318718261fc/users",
        "managers_url": "/v2/organizations/6f2e2826-xxxx-xxxx-xxxx-7318718261fc/managers",
        "billing_managers_url": "/v2/organizations/6f2e2826xxxx-xxxx-xxxx
-7318718261fc/billing_managers",
        "auditors_url": "/v2/organizations/6f2e2826-xxxx-xxxx-xxxx-7318718261fc/auditors",
        "app_events_url": "/v2/organizations/6f2e2826-xxxx-xxxx-xxxx-7318718261fc/app_events",
        "space_quota_definitions_url": "/v2/organizations/6f2e2826-xxxx-xxxx-xxxx-
7318718261fc/space_quota_definitions"
      }
    },
```

Using the GUID got previously for the org, we get the GUID for the space:

```
cf curl /v2/organizations/6f2e2826-xxxx-xxxx-xxxx-7318718261fc/spaces
```

```
{
  "total_results": 1,
  "total_pages": 1,
  "prev_url": null,
  "next_url": null,
  "resources": [
    {
      "metadata": {
        "guid": "66e599ab-xxxx-xxxx-xxxx-446946adfca9",
        "url": "/v2/spaces/66e599ab-xxxx-xxxx-xxxx-446946adfca9",
        "created_at": "2015-04-22T10:48:18Z",
        "updated_at": null
      },
      "entity": {
        "name": "chechu",
        "organization_guid": "6f2e2826-xxxx-xxxx-xxxx-7318718261fc",
        "space_quota_definition_guid": null,
        "allow_ssh": true,
        "organization_url": "/v2/organizations/6f2e2826-7ee0-4307-b1f4-7318718261fc",
        "developers_url": "/v2/spaces/66e599ab-xxxx-xxxx-xxxx-446946adfca9/developers",
        "managers_url": "/v2/spaces/66e599ab-xxxx-xxxx-xxxx-446946adfca9/managers",
        "auditors_url": "/v2/spaces/66e599ab-xxxx-xxxx-xxxx-446946adfca9/auditors",
        "apps_url": "/v2/spaces/66e599ab-xxxx-xxxx-xxxx-446946adfca9/apps",
        "routes_url": "/v2/spaces/66e599ab-xxxx-xxxx-xxxx-446946adfca9/routes",
        "domains_url": "/v2/spaces/66e599ab-xxxx-xxxx-xxxx-446946adfca9/domains",
        "service_instances_url": "/v2/spaces/66e599ab-xxxx-xxxx-xxxx-
446946adfca9/service_instances",
        "app_events_url": "/v2/spaces/66e599ab-xxxx-xxxx-xxxx-446946adfca9/app_events",
        "events_url": "/v2/spaces/66e599ab-xxxx-xxxx-xxxx-446946adfca9/events",
        "security_groups_url": "/v2/spaces/66e599ab-xxxx-xxxx-xxxx-446946adfca9/security_groups"
      }
    }
  ]
}
```

```
user="your_user"
passwd="your_passwd"
guid="your_space_guid"
```

- Create the function to get the authentication token:

```
def auth_token_get(user, passwd):
        ## auth url
```

```
	url = 'http://login.ng.bluemix.net/UAALoginServerWAR/oauth/token'
	## type of auth and the credentials
	body="grant_type=password&username="+user+"&password="+passwd
	## the post request
	auth=requests.post(url,params=body, headers={ 'authorization': 'Basic Y2Y6', 'accept': 
'application/json', 'content-type' : 'application/x-www-form-urlencoded' } )
	## we return it in JSON format
	return auth.json()['access_token']
```

- Create the function to list the containers based on the API calls showed in the documentation:

```
def list_containers():
	## get the auth token
	auth_token=auth_token_get(user,passwd)
	## url to get the list
	url = 'https://containers-api.ng.bluemix.net/v3/containers/json'
	## request for the list
	containers_list=requests.get(url, headers={"Accept": "application/json","X-Auth-
Token": auth_token, "X-Auth-Project-Id": guid} )
	containers=[]
	## read the list, and create  a JSON
	for container in containers_list.json():
		containers.append({'id':container['Id'],'name':container['Name'], 'status': 
container['Status']})
	## Return the list in JSON format
	return containers
```

- Open the file “server.py” created before, and add the files where we created the functions, below the libraries imported at the beginning of the file, import the library “sys” to read the files, and “json” to return the results:

```
import sys, json
sys.path.append ("softlayer")
sys.path.append ("openstack")
sys.path.append ("docker")
sys.path.append(“erp”)
import cci, vm, containers, erp  ##import the files with the functions
```

- In the server file add below main route:

```
@app.route('/' )
def root():
   return render_template("index.html", title = 'Dashboard')
```

a new route to list the cci, vms and containers:

```
@app.route('/list_vms', methods = ['GET'])
def list_vms():
	list_servers=[]
```

```
        ## get the cci from SL
        cci_softlayer=cci.getCCIs()
        ## get the VMs from BlueBox
        servers_os=vm.getServers()
        ## get the containers from Docker
        containers_bm=containers.list_containers()
        ## Create a list with all of them
        for server in servers_os:

        list_servers.append({'id':server['id'],'hostname':server['hostname'],'provider':'BlueBox',
"status":server['status']})

        for container in containers_bm:
                list_servers.append({'id':container['id'],'hostname':container['name'],"status":
container['status'],'provider':'Docker BM'})

        for server in cci_softlayer:


list_servers.append({'id':server['id'],'hostname':server['fullyQualifiedDomainName'],'provider':'
SoftLayer', "status": server['status']['name']})
        ## Return the list in JSON format
        return json.dumps(list_servers)
```

- Modify the Bootsrap template to add a table. Add these lines on the head section of the "template/index.html" file, to add the "css" files:

```
## Place them under this line:

<!-- Bootstrap Table -->
```

```
  <link rel="stylesheet" href="//cdnjs.cloudflare.com/ajax/libs/bootstrap-table/1.9.1/bootstrap-
table.min.css">
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/bootstrap3-
dialog/1.34.7/css/bootstrap-dialog.min.css">
```

Add the JS file on the body section at the end of the file "template/index.html":

```
## Place it under:
<!-- Functions for the portal -->
```

```
<script src="//cdnjs.cloudflare.com/ajax/libs/bootstrap-table/1.9.1/bootstrap-
table.min.js"></script>
<script src="//cdnjs.cloudflare.com/ajax/libs/bootstrap-table/1.9.1/locale/bootstrap-table-zh-
CN.min.js"></script>
```

We need to create a JS file to manage the table behavior, few need to include this file on the “html” file, we will create it after this:

```
## Place it under the line
<!-- Functions-->
```

```
<script src="static/js/functions.js"></script>
```

Add the table reference:

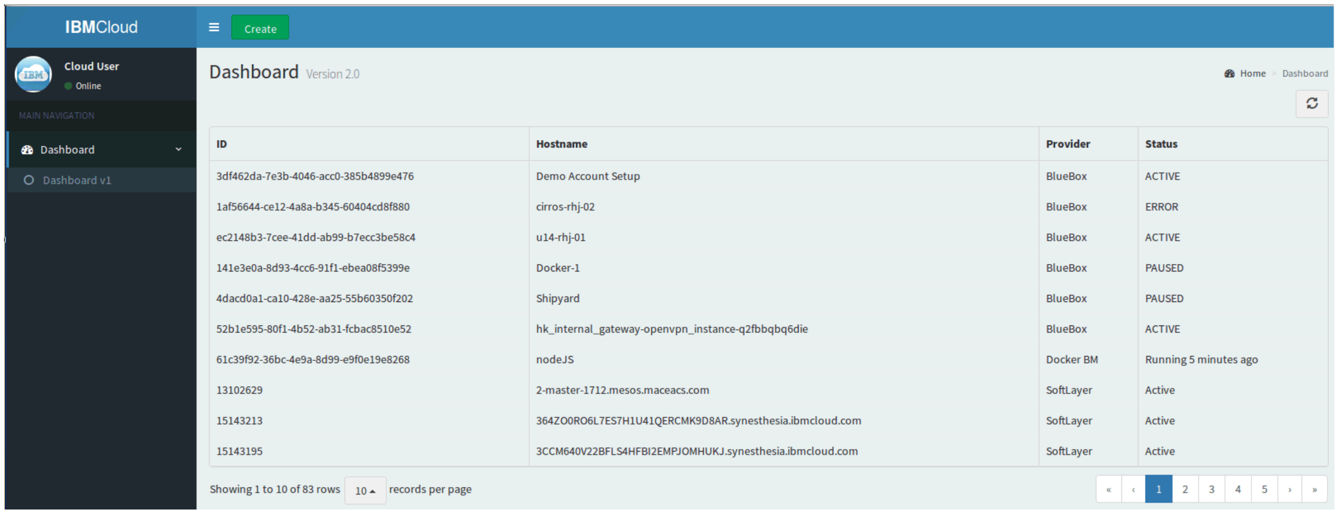
```
## place it under the lines:
    <ol class="breadcrumb">
      <li><a href="#"><i class="fa fa-dashboard"></i> Home</a></li>
      <li class="active">Dashboard</li>
     </ol>
```

```
<table id="table" data-toggle="table" data-locale="en-US"></table>
```

In Geany, create a file with the name “functions.js” in the folder “/static/js” on the working directory, and add the following code:

```
$('#table').bootstrapTable({
   id: "table",
   url: '/list_vms', // URL of the web service that lists the VMs and containers
   method: 'GET',
   pagesize: 10,
   pagination: true,
   clickToSelect: true,
   singleSelect: true,
   showRefresh: true,
   // Set the columns that we will show in the table, using the fields provided from the JSON in the
WS
   columns: [{
      field: 'id',
      title: 'ID'
   }, {
      field: 'hostname',
      title: 'Hostname'
   } , {
      field: 'provider',
      title: 'Provider'
   },{
      field: 'status',
      title: 'Status'
   }




   ]
});
```

Refresh the browser (http://localhost:5000), and you should see something like this:



| ID | Hostname | Provider | Status |
|---|---|---|---|
| 3df462da-7e3b-4046-acc0-385b4899e476 | Demo Account Setup | BlueBox | ACTIVE |
| 1af56644-ce12-4a8a-b345-60404cd8f880 | cirros-rhj-02 | BlueBox | ERROR |
| ec2148b3-7cee-41dd-ab99-b7ecc3be58c4 | u14-rhj-01 | BlueBox | ACTIVE |
| 141e3e0a-8d93-4cc6-91f1-ebea08f5399e | Docker-1 | BlueBox | PAUSED |
| 4dacd0a1-ca10-428e-aa25-55b60350f202 | Shipyard | BlueBox | PAUSED |
| 52b1e595-80f1-4b52-ab31-fcbac8510e52 | hk_internal_gateway-openvpn_instance-q2fbbqbq6die | BlueBox | ACTIVE |
| 61c39f92-36bc-4e9a-8d99-e9f0e19e8268 | nodeJS | Docker BM | Running 5 minutes ago |
| 13102629 | 2-master-1712.mesos.maceacs.com | SoftLayer | Active |
| 15143213 | 364ZO0RO6L7ES7H1U41QERCMK9D8AR.synesthesia.ibmcloud.com | SoftLayer | Active |
| 15143195 | 3CCM640V22BFLS4HFBI2EMPJOMHUKJ.synesthesia.ibmcloud.com | SoftLayer | Active |

Showing 1 to 10 of 83 rows 10 records per page

« ‹ 1 2 3 4 5 › »

## 2.3 Get details for servers and containers

Every item in IBM cloud offerings is based in an unique id per offering. We need to grab that id from the table and send it with an API request to get the details for the specific server (containers are not showed at this point).

- On the “cci.py” file, add the following code for the function that will list the CCI details on SoftLayer based on this method, that will return a list of this datatype object:

```
def getCCI(id_cci):
	##Mask to get the right data that we will use
	object_mask = 
'id,fullyQualifiedDomainName,operatingSystem.passwords,primaryBackendIpAddress,
primaryIpAddress, maxCpu,maxMemory,status'
	## we specify the id of the object we want to retrieve the info
	result = client['Virtual_Guest'].getObject(id=id_cci, mask=object_mask)
	## we do some string conversion
	network_fix=str(result['primaryBackendIpAddress'])+"
"+str(result['primaryIpAddress'])
	user_passwd=str(result['operatingSystem']['passwords'][0]['username'])+" /
"+str(result['operatingSystem']['passwords'][0]['password'])
	flavor_name= str(result['maxCpu'])+" vCPU, "+str(result['maxMemory'])+" GB RAM"
	## we return a Json object
	server_details= {"id": result['id'], "name": result['fullyQualifiedDomainName'],
"flavor": flavor_name,  "user_passwd": user_passwd, "image": result['operatingSystem']
['softwareLicense']['softwareDescription']['longDescription'], "networks":
network_fix,"status":result['status']['name']}
```

```
        return server_details
```

- On the “vm.py” file, add the following code for the function that will list the details of the VM in BlueBox based on this http://docs.openstack.org/developer/python-novaclient/ref/v1_1/servers.html

## we will use the function “getOptions()” to retrieve the options available on our BlueBox instance, so we can match later the id of the flavor, image,.. with  name that is human readable.

```
def getOptions():
        ## we get the flavors, images, security groups, networks and ssh keys
        list_flavors=nova.flavors.list()
        list_images=nova.images.list()
        list_sec_groups=nova.security_groups.list()
        list_networks=nova.networks.list()
        list_key_name=nova.keypairs.list()
        options=[]
        flavors=[]
        images=[]
        networks=[]
        keypairs=[]
        sec_groups = []
        for flavor in list_flavors:
                flavors.append({"id":flavor.id,"name":flavor.name})
        for image in list_images:
                images.append({"id":image.id,"name":image.name})
        for sec_group in list_sec_groups:
                sec_groups.append({"id":sec_group.id,"name":sec_group.name})
        for network in list_networks:
                networks.append({"id":network.id,"name":network.label})
        for key_pair in list_key_name:
                keypairs.append({"id":key_pair.id,"name":key_pair.name})

        options={"images":images,"flavors":flavors, "sec_groups":sec_groups, "networks":
networks, "keypairs": keypairs}
        return options
```

We get the server details using the previous function to identify the name of the options ids:

```
def getVM(id):
        ## we get the server details
        server=nova.servers.get(id)
        ## we get the options available
        options=getOptions()
        ## we match the id of the option in the server details, with a name in teh options
        sec_group_fix=""
        network_fix=""
        for option in options['flavors']:
            if option['id']==server.flavor['id']:
                flavor_name=option['name']
        for option in options['images']:
            if option['id']==server.image['id']:
                image_name=option['name']

        for sec_group in server.security_groups:
            sec_group_fix= str(sec_group['name'])+" "+str(sec_group_fix)
        for network in server.networks:
            network_fix=str(network)+" :"
            for ip in server.networks[network]:
                network_fix= str(network_fix)+" "+str(ip)
        ## we return a JSON with the info
        server_details= {"id": server.id, "name": server.name, "flavor": flavor_name,
"security_group": sec_group_fix,  "key_name": server.key_name, "image": image_name,
"networks": network_fix, "status": server.status}
        return server_details
```

- Add the new routes to the “server.py”

```
@app.route('/getServerDetails', methods = ['POST'])
def get_server_details():
        id = request.json['id']
        provider = request.json['provider']
        ## based on the provider we get details from one or another
        if str(provider)=="SoftLayer":
                details = cci.getCCI(id)
        if str(provider)=="BlueBox":
                details= vm.getVM(id)
        return json.dumps(details)
```

- Add the form to see the server details on the browser. On the file “templates/index.html” add the following code:

```
## place these lines under
    <table id="table" data-toggle="table" data-locale="en-US"></table>

    </section>
```

```
              <div id="details" class="box box-warning" hidden=true>
        <div class="box-header with-border">
          <h3 class="box-title">Server Details</h3>
        </div>
        <!-- /.box-header -->
        <div class="box-body">
          <form role="form">
            <!-- text input -->
              <div class="row">
```

```
                                                <div class="col-md-6">
                                                          <div class="form-group">
                                                                   <label>id</label>
                                                                   <input id="server_id" type="text"
class="form-control" >
                                                                   <label>Hostname</label>
                                                                   <input id="hostname" type="text"
class="form-control" >
                                                                   <label>Provider</label>
                                                                   <input id="provider" type="text" class="form-
control" >
                                                                   <label>flavor</label>
                                                                   <input id="flavor" type="text" class="form-
control" >
                                                                   <label>Status</label>
                                                                   <input id="status" type="text" class="form-
control" >
                                                         </div>
                                                </div>
                                        <div class="col-md-6">
                                                          <div class="form-group">
                                                                   <label>Image</label>
                                                                   <input id="image" type="text" class="form-
control" >
                                                                   <label>Security Group</label>
                                                                   <input id="sec_group" type="text"
class="form-control" >
                                                                   <label>Networks</label>
                                                                   <input id="networks" type="text"
class="form-control" >
                                                                   <label>Key Name</label>
                                                                   <input id="key_name" type="text"
class="form-control" >
                                                                   <label>User/Passwd</label>
                                                                   <input id="user_passwd" type="text"
class="form-control" >

                                                         </div>
                                      </div>
<!--
             <div class="form-group">
               <label>Networks</label>
               <input id="networks" type="text" class="form-control" >
             </div>
-->

           </form>
          </div>
          <!-- /.box-body -->
```

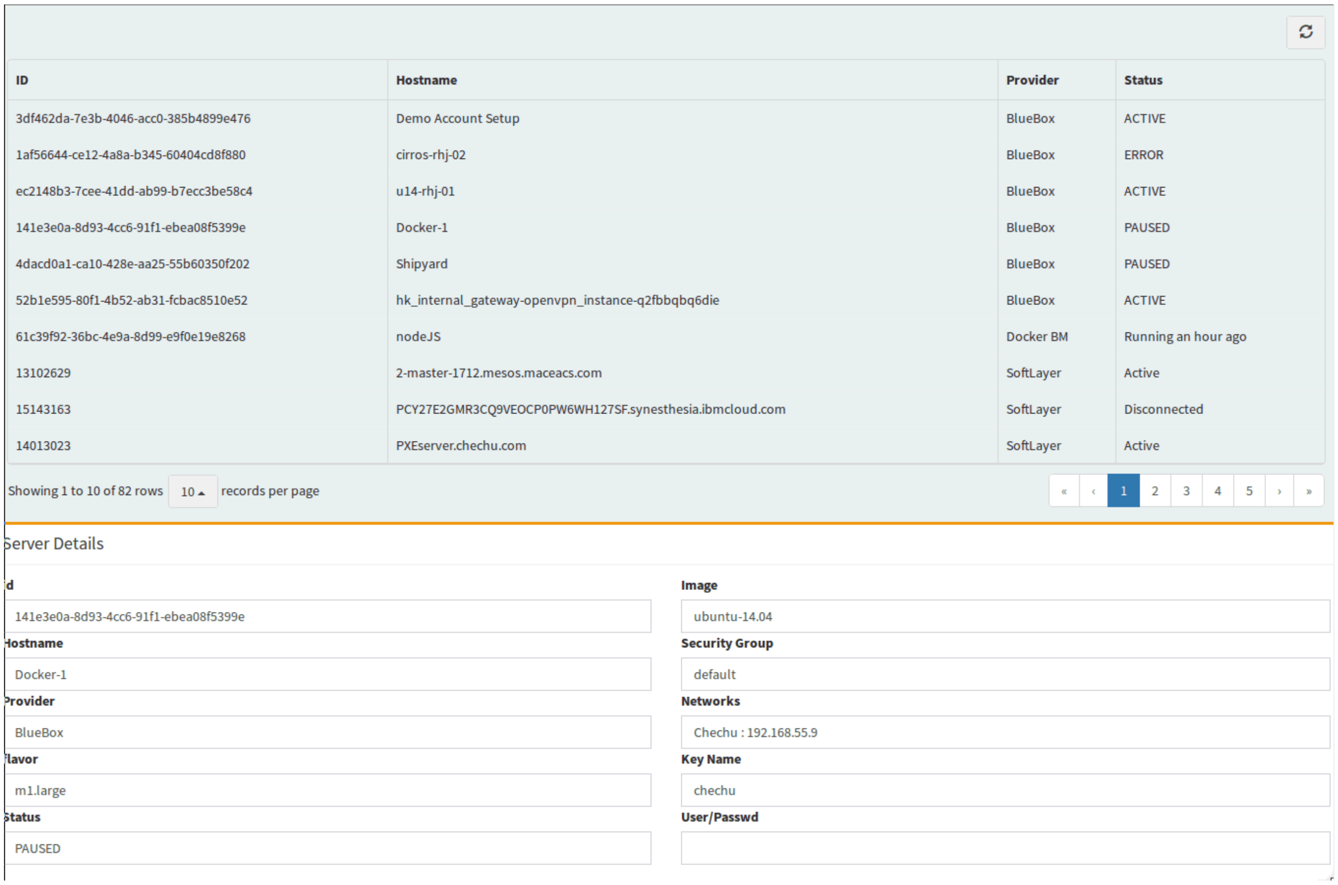
```
        </div>
        <!-- /.box -->
   <!-- Main content -->
   <!-- /.content -->
 </div>
```

– Add the following code to the file “static/js/functions.js” to add a click event on the table, and show the server details on the form:

**## add the lines on bold**

```
$('#table').bootstrapTable({
         id: "table",
   url: '/list_vms', // URL of the web service that lists the VMs and containers
   method: 'GET',
   pagesize: 10,
   pagination: true,
   clickToSelect: true,
   singleSelect: true,
   showRefresh: true,
   // Set the columns that we will show in the table, using the fields provided from the JSON in the WS
   columns: [{
      field: 'id',
      title: 'ID'
   }, {
      field: 'hostname',
      title: 'Hostname'
   } , {
      field: 'provider',
      title: 'Provider'
   },{
      field: 'status',
      title: 'Status'
   }



   ], //don't forget the ,
   onClickRow:  function (row, $element) {// event when we click in a row
      // we grab the provider and the server id
      provider = row.provider;
      id = row.id;
      $.ajax({ // we do an ajax call to the ws that will return the details
                                                      type: "POST",
                                                      url: "/getServerDetails",
                                                      data: JSON.stringify({  provider:provider, id:id}),//we
pass the provider and the server id
                                                      dataType: 'json',
                                          contentType: 'application/json',
                                                            success: function ( dataCheck){ //if the call is
successful we write the details on the fields in the form
                                                                                                                                    $
('#details').show();
                                                                                                                                    $
('#hostname').val(dataCheck.name);
                                                                                                                                    $
('#server_id').val(dataCheck.id);
                                                                                                                                    $
('#networks').val(dataCheck.networks);
                                                                                                                                    $
('#sec_group').val(dataCheck.security_group);
                                                                                                                                    $
('#image').val(dataCheck.image);
                                                                                                                                    $
('#key_name').val(dataCheck.key_name);
                                                                                                                                    $
('#user_passwd').val(dataCheck.user_passwd);
                                                                                                                                    $
('#flavor').val(dataCheck.flavor);
                                                                                                                                    $
('#provider').val(row.provider);
                                                                                                                                    $
('#status').val(row.status);

                                                            }


                                        });
   }
});
```

If we refresh the browser we can check that if we click in a row we get the server details in the bottom:



| ID | Hostname | Provider | Status |
|---|---|---|---|
| 3df462da-7e3b-4046-acc0-385b4899e476 | Demo Account Setup | BlueBox | ACTIVE |
| 1af56644-ce12-4a8a-b345-60404cd8f880 | cirros-rhj-02 | BlueBox | ERROR |
| ec2148b3-7cee-41dd-ab99-b7ecc3be58c4 | u14-rhj-01 | BlueBox | ACTIVE |
| 141e3e0a-8d93-4cc6-91f1-ebea08f5399e | Docker-1 | BlueBox | PAUSED |
| 4dacd0a1-ca10-428e-aa25-55b60350f202 | Shipyard | BlueBox | PAUSED |
| 52b1e595-80f1-4b52-ab31-fcbac8510e52 | hk_internal_gateway-openvpn_instance-q2fbbqbq6die | BlueBox | ACTIVE |
| 61c39f92-36bc-4e9a-8d99-e9f0e19e8268 | nodeJS | Docker BM | Running an hour ago |
| 13102629 | 2-master-1712.mesos.maceacs.com | SoftLayer | Active |
| 15143163 | PCY27E2GMR3CQ9VEOCP0PW6WH127SF.synesthesia.ibmcloud.com | SoftLayer | Disconnected |
| 14013023 | PXEserver.chechu.com | SoftLayer | Active |

Showing 1 to 10 of 82 rows 10 records per page

« ‹ 1 2 3 4 5 › »

Server Details

| | |
|---|---|
| Id | Image |
| 141e3e0a-8d93-4cc6-91f1-ebea08f5399e | ubuntu-14.04 |
| Hostname | Security Group |
| Docker-1 | default |
| Provider | Networks |
| BlueBox | Chechu : 192.168.55.9 |
| Flavor | Key Name |
| m1.large | chechu |
| Status | User/Passwd |
| PAUSED | |

## 2.4 Server operations

In order to pause, resume, reboot and delete the servers, we add the following lines.

- Add the following functions to the “cci.py” to add the operations: pause, start, reboot and delete. Taking as parameter the id of the CCI:

```
def pause(id):
        result = client['Virtual_Guest'].pause(id)
        return result
def play(id):
        result = client['Virtual_Guest'].resume(id)
        return result
def reboot(id):
        result = client['Virtual_Guest'].rebootSoft(id)
        return result
def delete(id):
        result = client['Virtual_Guest'].deleteObject(id)
        return result
```

- Add the following functions to the “vm.py” to add the operations: pause, start, reboot and delete.

Taking as parameter the id of the VM:

```
def pause(id):
	result=nova.servers.pause(id)
	return True
def reboot(id):
	result=nova.servers.reboot(id,reboot_type='SOFT')
	return True
def play(id):
	result=nova.servers.unpause(id)
	return True
def delete(id):
	result=nova.servers.delete(id)
	return True
```

- Add the routes to the “server.py”:

```
@app.route('/pause', methods = ['POST'])
def pause():
	id = request.json['id']
	provider = request.json['provider']
	if provider == "SoftLayer":
		result=cci.pause(id)
	if provider=="BlueBox":
		result=vm.pause(id)
	return json.dumps(result)
@app.route('/play', methods = ['POST'])
def play():
	id = request.json['id']
	provider = request.json['provider']
	if provider == "SoftLayer":
		result=cci.play(id)
	if provider=="BlueBox":
		result=vm.play(id)
	return json.dumps(result)
@app.route('/delete', methods = ['POST'])
def delete():
	id = request.json['id']
	provider = request.json['provider']
	if provider == "SoftLayer":
		result=cci.delete(id)
	if provider=="BlueBox":
		result=vm.delete(id)
	return json.dumps(result)
@app.route('/reboot', methods = ['POST'])
def reboot():
	id = request.json['id']
	provider = request.json['provider']
	if provider == "SoftLayer":
```

```
            result=cci.reboot(id)
    if provider=="BlueBox":
            result=vm.reboot(id)
    return json.dumps(result)
```

- Add the following code to the “templates/index.html” in order to show the icons for the operations on the form we created in the previous step:

```
## Place them under these lines

        <!-- /.box-header -->
        <div class="box-body">
          <form role="form">
            <!-- text input -->
              <div class="row">
```

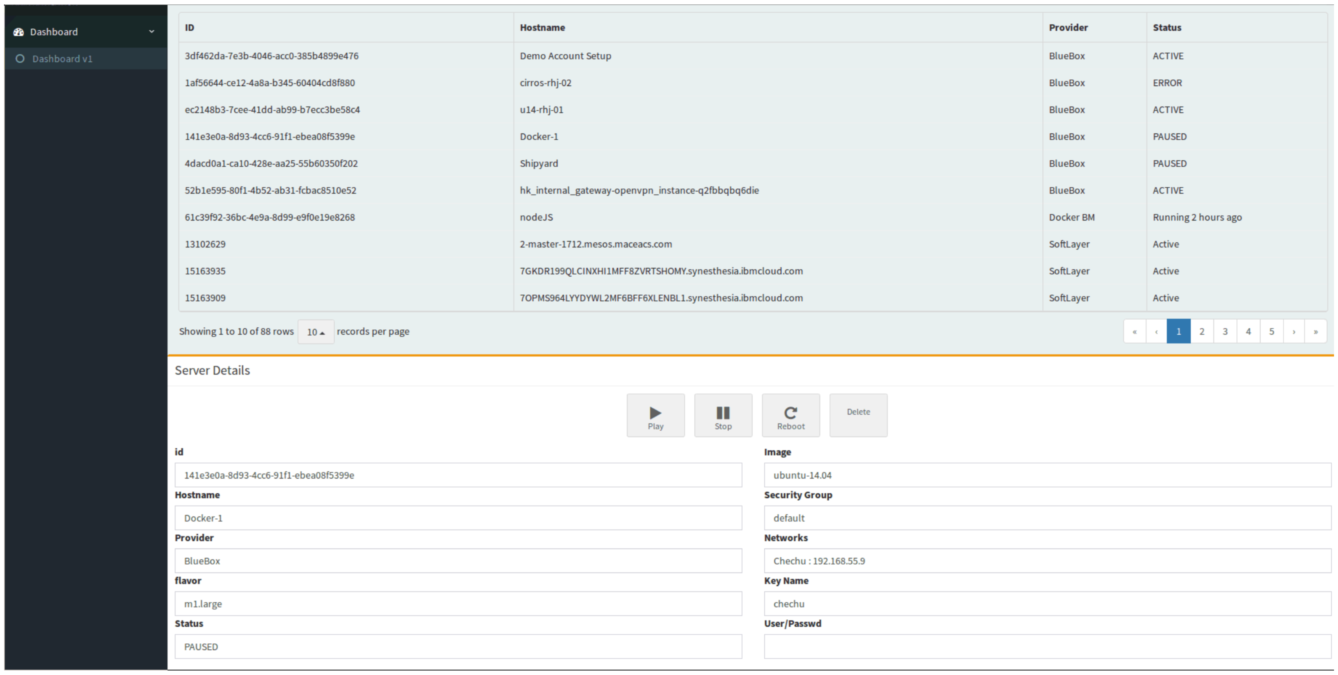
```
 <div class="col-md-12 " style="text-align:center;">
 <a id="play" class="btn btn-app" vertical-align="middle">
         <i  class="fa fa-play"></i>
                 Play
</a>
 <a id="pause" class="btn btn-app">
         <i  class="fa fa-pause"></i>
                 Stop
</a>
 <a  id="repeat" class="btn btn-app">
         <i class="fa fa-repeat"></i>
                 Reboot
</a>
 <a id="delete" class="btn btn-app">
         <i  class="fa fa-delete"></i>
                 Delete
</a>
</div>
```

- Add the event on the “static/js/functions.js” in order to execute the operation when we click the icons:

**## Place these lines add the end of the file**

```
$('#play').on('click', function (event) {
            // grab the provider and the sevrer id
            data = $('#provider').val();
            id= $('#server_id').val();
            // ajax call to perform the operation against the wb
      $.ajax({
             type: "POST",
             url: "/play",
             data: JSON.stringify({  id: id, provider: data}),
             dataType: 'json',
          contentType: 'application/json',
                          success: function ( dataCheck){
                                      $('#table').bootstrapTable('refresh');// refresh the table
                        }
            });
});
$('#pause').on('click', function (event) {
                // grab the provider and the sevrer id
                        data = $('#provider').val();
                        id= $('#server_id').val();
              $.ajax({
                          type: "POST",
                          url: "/pause",
                          data: JSON.stringify({  id: id, provider: data}),
                          dataType: 'json',
          contentType: 'application/json',
                          success: function ( dataCheck){
                                     $('#table').bootstrapTable('refresh');// refresh the table
                        }
            });
});

$('#delete').on('click', function (event) {
                        // grab the provider and the sevrer id
                        data = $('#provider').val();
                        id= $('#server_id').val();
                        // ajax call to perform the operation against the wb
              $.ajax({
                          type: "POST",
                          url: "/delete",
                          data: JSON.stringify({  id: id, provider: data}),
                          dataType: 'json',
          contentType: 'application/json',
                          success: function ( dataCheck){
                                      $('#table').bootstrapTable('refresh');// refresh the table
                        }
            });
});

$('#reboot').on('click', function (event) {
                        // grab the provider and the sevrer id
                        data = $('#provider').val();
                        id= $('#server_id').val();
                        // ajax call to perform the operation against the wb
              $.ajax({
                          type: "POST",
                          url: "/reboot",
                          data: JSON.stringify({  id: id, provider: data}),
                          dataType: 'json',
        contentType: 'application/json',
                    success: function ( dataCheck){
                             $('#table').bootstrapTable('refresh'); // refresh the table
                   }
         });
});
```

If we refresh the browser, and click on a server we will see this:



| ID | Hostname | Provider | Status |
|---|---|---|---|
| 3df462da-7e3b-4046-acc0-385b4899e476 | Demo Account Setup | BlueBox | ACTIVE |
| 1af56644-ce12-4a8a-b345-60404cd8f880 | cirros-rhj-02 | BlueBox | ERROR |
| ec2148b3-7cee-41dd-ab99-b7ecc3be58c4 | u14-rhj-01 | BlueBox | ACTIVE |
| 141e3e0a-8d93-4cc6-91f1-ebea08f5399e | Docker-1 | BlueBox | PAUSED |
| 4dacd0a1-ca10-428e-aa25-55b60350f202 | Shipyard | BlueBox | PAUSED |
| 52b1e595-80f1-4b52-ab31-fcbac8510e52 | hk_internal_gateway-openvpn_instance-q2fbbqbq6die | BlueBox | ACTIVE |
| 61c39f92-36bc-4e9a-8d99-e9f0e19e8268 | nodeJS | Docker BM | Running 2 hours ago |
| 13102629 | 2-master-1712.mesos.maceacs.com | SoftLayer | Active |
| 15163935 | 7GKDR199QLCINXHI1MFF8ZVRTSHOMY.synesthesia.ibmcloud.com | SoftLayer | Active |
| 15163909 | 7OPMS964LYYDYWL2MF6BFF6XLENBL1.synesthesia.ibmcloud.com | SoftLayer | Active |

Showing 1 to 10 of 88 rows 10 records per page « ‹ 1 2 3 4 5 › »

| | |
|---|---|
| id | Image |
| 141e3e0a-8d93-4cc6-91f1-ebea08f5399e | ubuntu-14.04 |
| Hostname | Security Group |
| Docker-1 | default |
| Provider | Networks |
| BlueBox | Chechu : 192.168.55.9 |
| flavor | Key Name |
| m1.large | chechu |
| Status | User/Passwd |
| PAUSED | |

If we click on the new icons we will see the operations being performed, and the table being refreshed.

## 2.5 Create Odoo Entry

To create an entry in the ERP, we use the out-of-the-box Odoo Equipment Module.

- Add the content to “erp/erp.py”

```
url = 'http://localhost:8069'
db = 'lab'
username = 'admin'
password = 'admin'

BLUEBOX_VENDOR_ID = 47
SOFTLAYER_VENDOR_ID = 46
CATEGORY_ID = 6 # ID of Cloud VM Inventory Category

import xmlrpclib

common = xmlrpclib.ServerProxy('{}/xmlrpc/2/common'.format(url))
uid = common.authenticate(db, username, password, {})
```

```
models = xmlrpclib.ServerProxy('{}/xmlrpc/2/object'.format(url))

# Search for all cloud VMs
def getVmInventory():
        result = models.execute_kw(db, uid, password,'hr.equipment', 'search_read',
[[['category_id', '=', 'Cloud VMs']]])
        return result

# Create a new record
def addVmInventory(name = None, vendor = None, cost = float(0.00)):
        if vendor == 'SoftLayer':
                partner_id = SOFTLAYER_VENDOR_ID
        if vendor == 'BlueBox':
                partner_id = BLUEBOX_VENDOR_ID
        vm_details = {
                'name': name,
                'category_id': CATEGORY_ID,
                'partner_id': partner_id,
                'cost': float(cost)
        }
        models.execute_kw(db, uid, password, 'hr.equipment', 'create', [vm_details])
```

## 2.6 Create server

In order to create a server we need to gather the options that each provider offer to us to provision a server.
After the server is successfully created, we call the “erp.addVmInventory” function to add the newly created VM to the ERP VM Equipment list

- Add the following function to the file “cci.py” based on this API method:

```
def createOptions():
        options = client['Virtual_Guest'].getCreateObjectOptions()
        return options
def createCCIServer(hostname, domain,processor,memory,block,os,network,location):
        ## we build the order_template needed
        parameters = {
           "hostname": hostname,
                "domain": domain,
                "startCpus": processor,
                "maxMemory": memory,
                "hourlyBillingFlag": "true",
          'operatingSystemReferenceCode' : os,
          'localDiskFlag' : False,
          'datacenter' : {'name': location}
           }
        result = client['Virtual_Guest'].createObject(parameters)
        return result
```

- Add the following function to the file “vm.py” based on the Nova API documentation:

```
## Options gathering function was cerated in the previous steps for BlueBox
```

```
def createVM(name, image_id, flavor_id,sec_group,  key_name,nic):
        nic = [{'net-id': nic}] ## network parameter needs to be specified in this format
        server = nova.servers.create(name, flavor_id, image_id, security_groups=[sec_group],
key_name=key_name,nics=nic)
        return server.name
```

- Add the routes to the file “server.py”, one for the CCI in SoftLayer and another for the VM in BlueBox:

```
@app.route('/create_options', methods = ['POST'])
def create_options():
        provider = request.json['provider']
        if str(provider)=="SoftLayer":
                options = cci.createOptions()
        if str(provider)=="BlueBox":
                options = vm.getOptions()
        return json.dumps(options)
@app.route('/create_cci', methods = ['POST'])
def create_cci():
        location = request.json['location']
        processor = request.json['processor']
        memory = request.json['memory']
        block = request.json['block']
        network = request.json['network']
        os = request.json['os']
        hostname = request.json['hostname']
        domain = request.json['domain']
        result = cci.createCCIServer(hostname,
domain,processor,memory,block,os,network,location)
        erp.addVmInventory(name = hostname, vendor = 'SoftLayer', cost = 0.00)
        return json.dumps(result)
@app.route('/create_vm', methods = ['POST'])
def create_vm():
        processor = request.json['processor']
        network = request.json['network']
        os = request.json['os']
        hostname = request.json['hostname']
        keypair = request.json['keypair']
        sec_group = request.json['sec_group']
        result = vm.createVM(hostname,processor,os,sec_group, keypair[0],network)
        erp.addVmInventory(name = hostname, vendor = 'BlueBox', cost = 0.00)
        return json.dumps(result)
```

- Add the following code at the end of the file “static/js/functions.js” to generate the dialog window to provision a new server:

```
$('#create').on('click', function (event) { // this will trigger when click on the create server icon

htmlRemove = function(id){ // this code will help to remove html code when it will not be needed
based on teh provider selection

            var elem = document.getElementById(id);
            if (elem != null)
            elem.parentNode.removeChild(elem);
}

 BootstrapDialog.show({ //this will create the window that will pop up
                                    id: "create_server",
                                    title: "Create Server...",
                                    closable: false,
                                     buttons: [{
                    label: 'Cancel',
                    cssClass: 'btn-danger',
                    action: function(dialogRef){// this will close teh window when we click on Cancel
                        dialogRef.close();
                    }},
                    {
                    label: 'Create',
                    cssClass: 'btn-primary',
                    action: function(dialogRef){ // this is the action that will take place when click
oncreate button
                                                                        data = $('#provider_select').val(); // grab the provider
selection
                                                                        if(data=='SoftLayer'){ // if SoftLayer is selected
                                                                                    //grab these data for the provisioning
                                    processor = $("#CPU option:selected").val();
                                                                                    memory = $('#memory option:selected').val();
                                                                                    block = $('#block_devices
option:selected').val();
                                                                                    os = $('#OS option:selected').val();
                                                                                    network = $('#network option:selected').val();
                                                                                    datacenter = $('#datacenter
option:selected').val();
                                                                                    hostname = $('#server_name').val();
                                                                                    domain = $('#domain').val();
                                                                                    $.ajax({// perform the ajax call to provision
anew server
                                                                                      type: "POST",
                                                                                      url: "/create_cci",
                                                                                      data: JSON.stringify({  processor: processor,
location: datacenter, memory: memory, block: block, network: network, os: os, hostname:
hostname, domain: domain}),
                                                                                dataType: 'json',
                                                                contentType: 'application/json',
                                                                                success: function ( dataCheck)

           {

           BootstrapDialog.show({// show a windows uinforming the server was created
                                                                    message: 'Server created!'
                                                                });
                                                                              }



                                                         });
                              }
                              //same behaviour for Bluebox
                              if(data=='BlueBox'){
                                  processor = $("#CPU option:selected").val();
                                                                              os = $('#OS option:selected').val();
                                                                              network = $('#network option:selected').val();
                                                                              hostname = $('#server_name').val();
                                                                              keypair = $('#keypair').val();
                                                                              sec_group = $('#sec_groups').val();
                                                                              $.ajax({
                                                                                type: "POST",
                                                                                url: "/create_vm",
                                                                                data: JSON.stringify({  processor: processor,
network: network, os: os, hostname: hostname,  sec_group:sec_group[0], keypair:keypair}),
                                                                                dataType: 'json',
                                                                contentType: 'application/json',
                                                                                success: function ( dataCheck)

           {

           BootstrapDialog.show({
                                                                    message: 'Server created!'
                                                                });
                                                                              }



                                                         });
                              }
                   }

                   },
                   ],
           message: function(dialog) { // Add the html code that will appear in the provisioning
window, fields, button...
```

```
            var code = '<div class="box box-default"> <div class="box-header with-border"> \
                                                            <h3 class="box-title">Create Server...</h3> \
                                                            <div class="box-tools pull-right">\
          </div>\
        </div>\
        <!-- /.box-header -->\
        <div class="box-body">\
          <div class="row">\
            <div class="col-md-6">\
              <div class="form-group">\
                <label>Provider</label>\
                <select id="provider_select" class="form-control select2" style="width: 100%;">\
                  <option selected="selected">Select one...</option>\
                  <option>SoftLayer</option>\
                  <option>BlueBox</option>\
                </select>\
                 <label>CPU</label>\
                <select class="form-control select2" id="CPU" multiple="CPU" data-
placeholder="Select a CPU" style="width: 100%;">\
                </select>\
                <label>OS</label>\
                <select class="form-control select2" id="OS" multiple="Operating System" data-
placeholder="Select a OS" style="width: 100%;">\
                </select>\
                <label>Network</label>\
                <select class="form-control select2" id="network" multiple="Network data-
placeholder="Select a Network" style="width: 100%;">\
                </select>\
                 <label>hotname</label>\
                 <input id="server_name" type="text" name="hostname">\
                </select>\
              </div>\
              </div>\
               <div class="col-md-6">\
              <div id="group" class="form-group">\
                <div id="Option_1" \
                </div>\
              </div>\
              </div>\
              </div>\
              </div>\
              </div>\
              <script src="static/js/functions.js"></script>'
                                    return code;}
        });
})
```

```
$("#provider_select").on('change', function (event) {// this function will gather the options for
ecah provider when selected
  data = $('#provider_select').val();
  $.ajax({// ajax call to get the options
                          type: "POST",
                          url: "/create_options",
                          data: JSON.stringify({  provider: data}),
                          dataType: 'json',
          contentType: 'application/json',
                          success: function ( dataCheck){
                                                                                    if(data=='SoftLayer')// if softlayer is selected we
do some arrangements
                                                                                                            {
                                                                                                            htmlRemove('sec_group_div');//
we remove options that we dont need for SoftLayer
                                                                                                            htmlRemove('key_name_div');
                                                                                                            // we add the option we need for
SoftLayer
                                                                                                            var html_softlayer =  '<div
id="datacenter_div" \

             <label>location</label>\
                                                                                                                                <select
class="form-control select2" id="datacenter" multiple="Location" data-placeholder="Select a
Datacenter" style="width: 100%;">\
                                                                                                                                </select>\
                                                                                                                                </div>\
                                                                                                                                <div
id="memory_div" label=Memory"\

<label>Memory</label>\
                                                                                                                                <select
class="form-control select2" id="memory" multiple="Memory" data-placeholder="Select a
Memory" style="width: 100%;">\
                                                                                                                                </select>\
                                                                                                                                </div>\
                                                                                                                                <div
id="block_div" label=Block Devices"\
                                                                                                                                 <label>Block
Devices</label>\
                                                                                                                                <select
class="form-control select2" id="block_devices" multiple="Block Devices" data-
placeholder="Select a storage" style="width: 100%;">\
                                                                                                                                </select>\
                                                                                                                                </div>\
                                                                                                                                <div
id="domain_div" label=Domain"\
             <label>Domain</label>\

             <input id="domain" type="text" name="domain">\

            </select>\

            </div>'
                                                                    // we add teh html code with the
options to teh windows html code

            document.getElementById("Option_1").innerHTML = html_softlayer;
                                                                    $
(dataCheck).each(function(index, value){

            $(value.datacenters).each(function(index, value){

                        $('#datacenter').append($('<option>', {

                                    value: value.template.datacenter.name,

                                    text : value.template.datacenter.name

                        }));

            });

                                                                    });
                                                                    $
(dataCheck).each(function(index, value){

            $(value.processors).each(function(index, value){

                        $('#CPU').append($('<option>', {

                                    value: value.template.startCpus,

                                    text : value.itemPrice.item.description

                        }));

            });

                                                                    });
                                                                    $
(dataCheck).each(function(index, value){

            $(value.memory).each(function(index, value){

                        $('#memory').append($('<option>', {
```

```
                        value: value.template.maxMemory,

                        text : value.itemPrice.item.description

                })); 

        });

                                                                        });
                                                                        $
(dataCheck).each(function(index, value){

        $(value.blockDevices).each(function(index, value){

                $('#block_devices').append($('<option>', {

                        value: value.template.blockDevices[0].diskImage.capacity,

                        text : value.itemPrice.item.description

                })); 

        });

                                                                        });
                                                                        $
(dataCheck).each(function(index, value){

        $(value.operatingSystems).each(function(index, value){

                $('#OS').append($('<option>', {

                        value: value.template.operatingSystemReferenceCode,

                        text : value.itemPrice.item.description

                })); 

        });

                                                                        });
                                                                        $
(dataCheck).each(function(index, value){

        $(value.networkComponents).each(function(index, value){

                $('#network').append($('<option>', {
```

```
                              value: value.template.networkComponents[0].maxSpeed,

                              text : value.itemPrice.item.description

                    }));

          });

                                                                                          });
                                                                                }
                                                                      if(data=='BlueBox'){// same behaviour
forBlueBox selection


                                                                                          htmlRemove('memory_div');
                                                                                          htmlRemove('block_div');
                                                                                          htmlRemove('domain_div');
                                                                                          htmlRemove('datacenter_div');
                                                                                          var html_sec_group =  '<div
id="sec_groups_div" \

          <label>Security Groups</label>\
                                                                                                                        <select
class="form-control select2" id="sec_groups" multiple="Security Groups" hidden data-
placeholder="Select a security groups" style="width: 100%;">\
                                                                                                                        </select>\
                                                                                                                        </div>\
                                                                                                                        <div
id="key_name_div" \

          <label>KeyPairs</label>\
                                                                                                                        <select
class="form-control select2" id="keypair" multiple="KeyPairs" hidden data-
placeholder="Select a KeyPairs" style="width: 100%;">\
                                                                                                                        </select>\
                                                                                                                        </div>';


          document.getElementById("Option_1").innerHTML = html_sec_group;
                                                                                          $
(dataCheck).each(function(index, value){

          $(value.flavors).each(function(index, value){

                    $('#CPU').append($('<option>', {

                              value: value.id,

                              text : value.name
```

```
                }));

        });

                                                                        });
                                                                        $
(dataCheck).each(function(index, value){

        $(value.images).each(function(index, value){

                $('#OS').append($('<option>', {

                        value: value.id,

                        text : value.name

                }));

        });

                                                                        });
                                                                        $
(dataCheck).each(function(index, value){

        $(value.networks).each(function(index, value){

                $('#network').append($('<option>', {

                        value: value.id,

                        text : value.name

                }));

        });

                                                                        });

                                                                        $
(dataCheck).each(function(index, value){

        $(value.sec_groups).each(function(index, value){

                $('#sec_groups').append($('<option>', {

                        value: value.id,

                        text : value.name
```

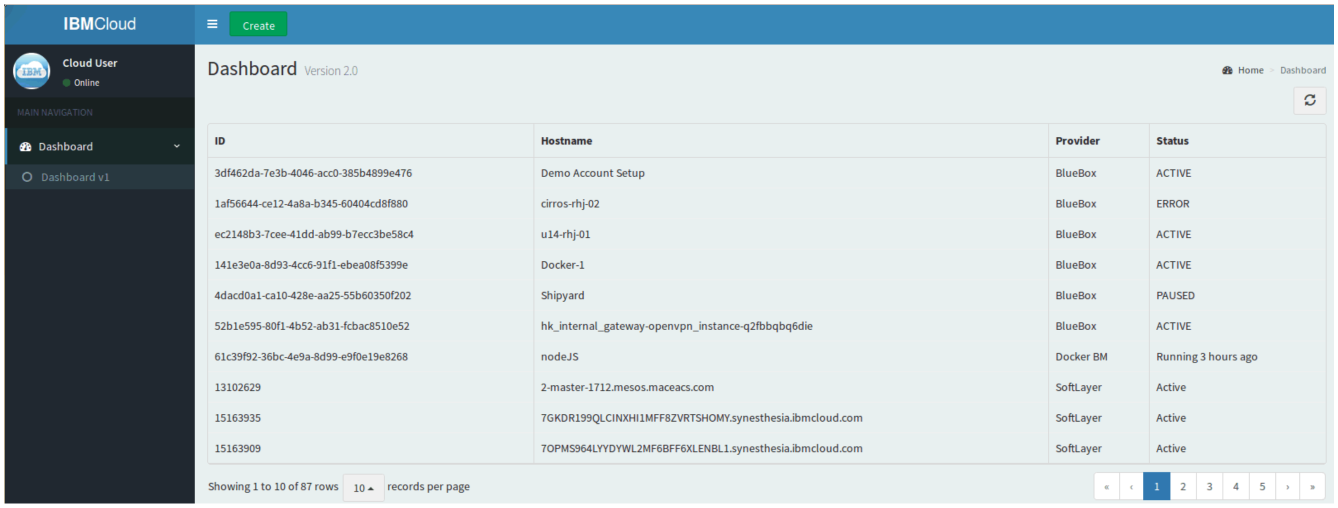
```
                    }));

          });

                                                                                         });
                                                                                         $
(dataCheck).each(function(index, value){

          $(value.keypairs).each(function(index, value){

                    $('#keypair').append($('<option>', {

                              value: value.id,

                              text : value.name

                    }));

          });
                                                                                         });

                                                                               }

}
});

});
```

- Add the following line to create the provisioning server button, to the file “templates/index.html”.

```
# place it under the lines
   <!-- Header Navbar: style can be found in header.less -->
   <nav class="navbar navbar-static-top" role="navigation">
     <!-- Sidebar toggle button-->
     <a href="#" class="sidebar-toggle" data-toggle="offcanvas" role="button">
       <span class="sr-only">Toggle navigation</span>
     </a>
```

```
<td><button id="create" class="btn btn-block btn-success"
type="button">Create</button></td>
```

Refresh the browser and you will be able to provision a new server:

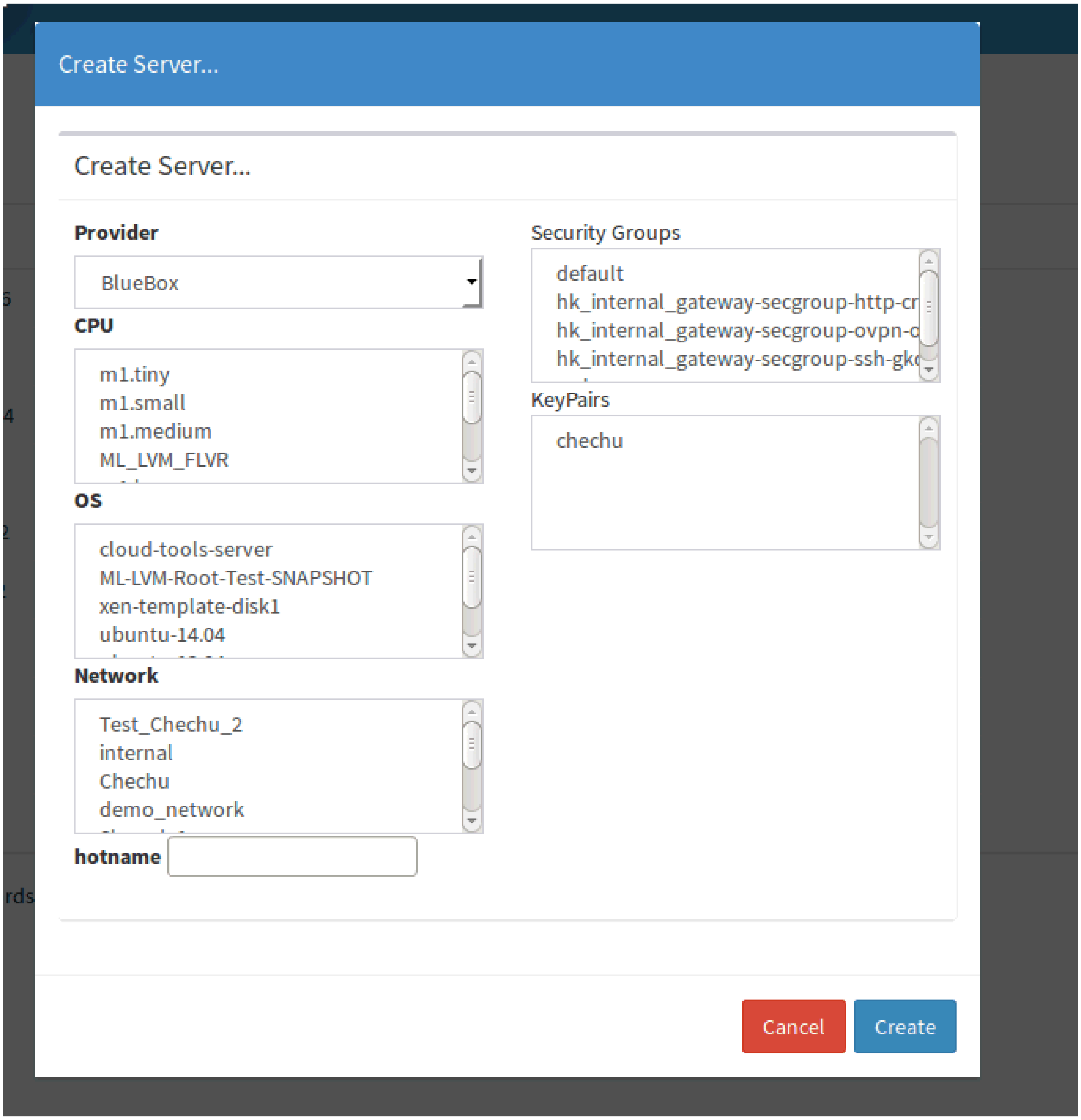


Dashboard Version 2.0

| ID | Hostname | Provider | Status |
|---|---|---|---|
| 3df462da-7e3b-4046-acc0-385b4899e476 | Demo Account Setup | BlueBox | ACTIVE |
| 1af56644-ce12-4a8a-b345-60404cd8f880 | cirros-rhj-02 | BlueBox | ERROR |
| ec2148b3-7cee-41dd-ab99-b7ecc3be58c4 | u14-rhj-01 | BlueBox | ACTIVE |
| 141e3e0a-8d93-4cc6-91f1-ebea08f5399e | Docker-1 | BlueBox | ACTIVE |
| 4dacd0a1-ca10-428e-aa25-55b60350f202 | Shipyard | BlueBox | PAUSED |
| 52b1e595-80f1-4b52-ab31-fcbac8510e52 | hk_internal_gateway-openvpn_instance-q2fbbqbq6die | BlueBox | ACTIVE |
| 61c39f92-36bc-4e9a-8d99-e9f0e19e8268 | nodeJS | Docker BM | Running 3 hours ago |
| 13102629 | 2-master-1712.mesos.maceacs.com | SoftLayer | Active |
| 15163935 | 7GKDR199QLCINXHI1MFF8ZVRTSHOMY.synesthesia.ibmcloud.com | SoftLayer | Active |
| 15163909 | 7OPMS964LYYDYWL2MF6BFF6XLENBL1.synesthesia.ibmcloud.com | SoftLayer | Active |

Showing 1 to 10 of 87 rows 10 records per page

« ‹ 1 2 3 4 5 › »


Create Server...

Create Server...

**Provider**

BlueBox

**CPU**

m1.tiny
m1.small
m1.medium
ML_LVM_FLVR

**OS**

cloud-tools-server
ML-LVM-Root-Test-SNAPSHOT
xen-template-disk1
ubuntu-14.04

**Network**

Test_Chechu_2
internal
Chechu
demo_network

**hotname**

Security Groups

default
hk_internal_gateway-secgroup-http-cr
hk_internal_gateway-secgroup-ovpn-c
hk_internal_gateway-secgroup-ssh-gk

KeyPairs

chechu

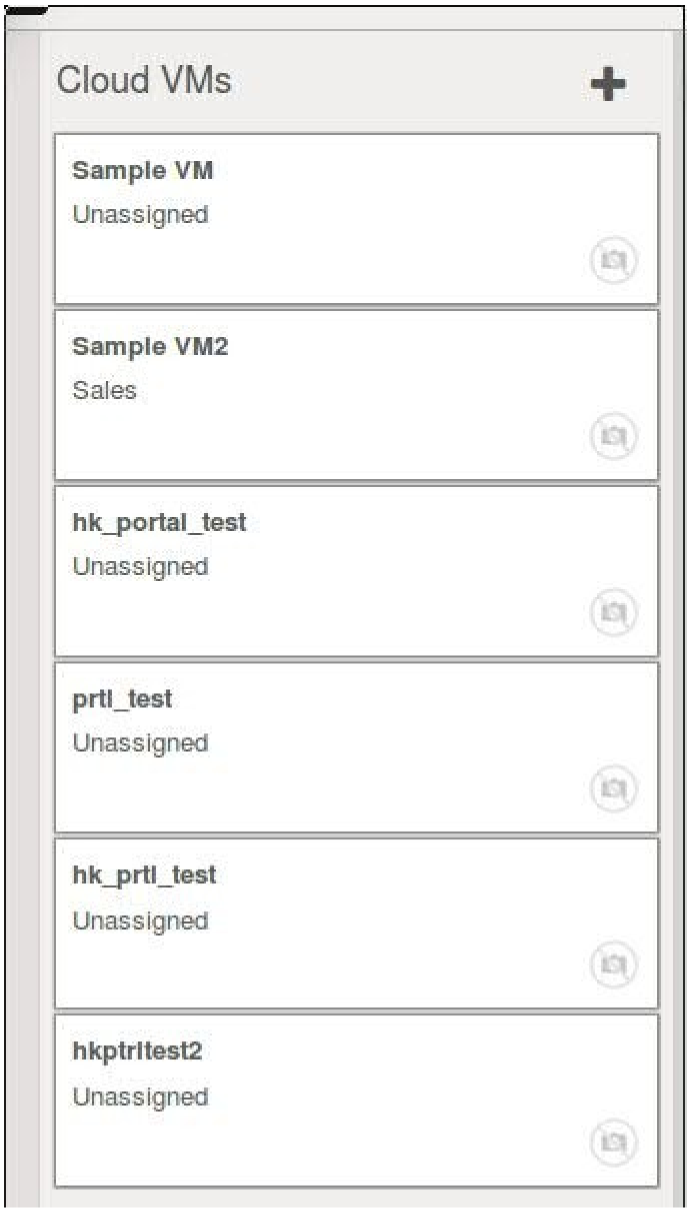
## 2.7 View the new server in Odoo

When you provision a new server in the portal, you will see the new server appearing in the ERP Equipment Module

- Open the URL http://localhost:8069
  The username / password is admin / admin if requested

- Navigate to the "Equipments" module at the top navigation bar.

You will see your newly provisioned VM in the “Cloud VMs” category, together with the existing sample data



**Sample VM**
Unassigned

**Sample VM2**
Sales

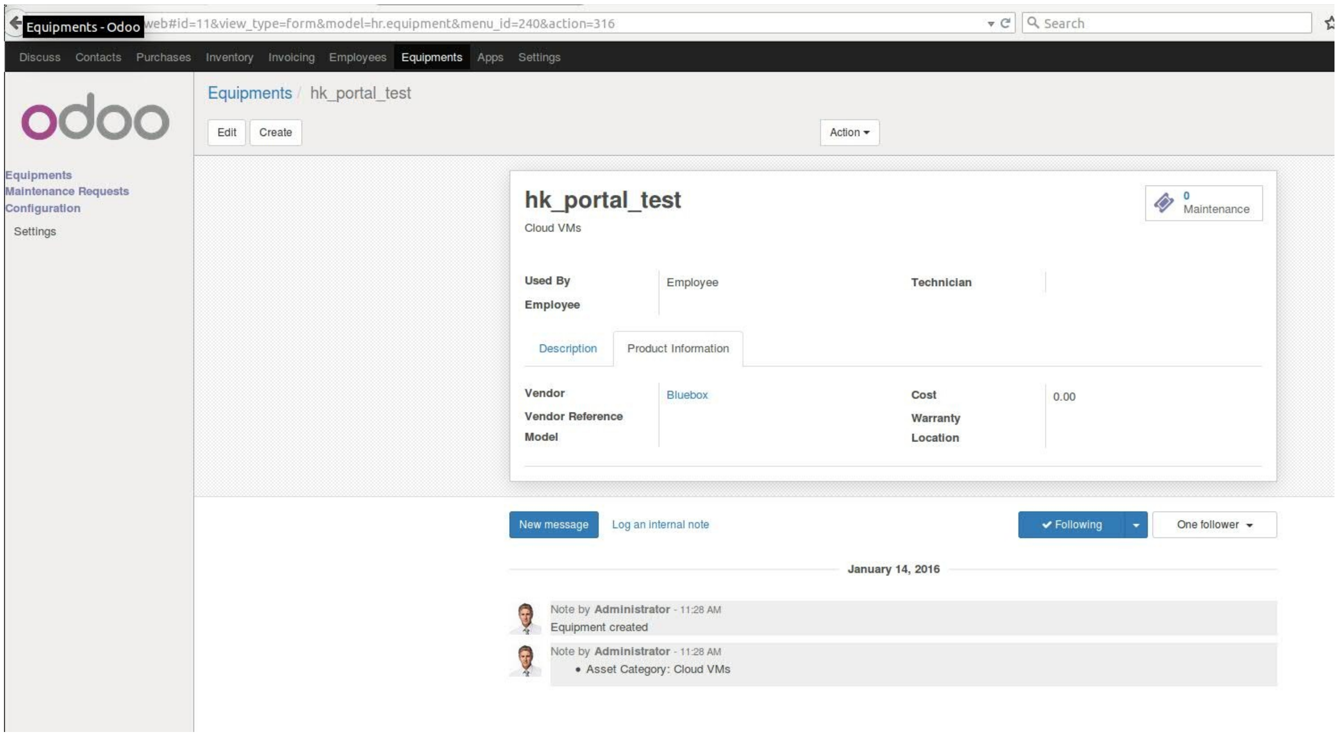
**hk_portal_test**
Unassigned

**prtl_test**
Unassigned

**hk_prtl_test**
Unassigned

**hkptrltest2**
Unassigned

Each VM will have details including the Cloud Vendor



Equipments / hk_portal_test

**hk_portal_test**

Cloud VMs

0 Maintenance

| | | | |
|---|---|---|---|
| Used By | Employee | Technician | |
| Employee | | | |

Description | Product Information

| | | | |
|---|---|---|---|
| Vendor | Bluebox | Cost | 0.00 |
| Vendor Reference | | Warranty | |
| Model | | Location | |

January 14, 2016

Note by Administrator - 11:28 AM
Equipment created

Note by Administrator - 11:28 AM
- Asset Category: Cloud VMs