OpenStack: How do I install Cloud Foundry on Bright OpenStack?

Cloud Foundry installation on Bright Openstack

1- A working Openstack environment, API access, credentials, and a working networking configuration with metadata access is assumed.

Running the following command inside an instance is a way to check that the metadata access is functioning:

```
$ curl http://169.254.169.254
1.0
2007-01-19
2007-03-01
2007-08-29
2007-10-10
2007-12-15
2008-02-01
2008-09-01
2009-04-04
```

Instances should reach other instances on the same network using internal network.

Also, it should be possible to create volumes and attach them to the instances. It is also mandatory to have access to open ports, be able to edit and create security groups and associate with floating IP addresses.

2- The development tools are first installed on the head node or a compute node:

```
# yum groupinstall -y development
# curl -L get.rvm.io | bash -s stable
# source /etc/profile.d/rvm.sh
# rvm reload
# rvm install 2.1.0
# rvm use 2.1.0 --default
```

If installing the development tools on a compute node, then these commands must be followed up with:

```
# cmsh

% device use <Node name>

% grabimage -w
```

3- The BOSH CLI and BOSH CLI micro-plugin are now installed:

```
gem install bosh_cli bosh_cli_plugin_micro --no-ri --no-rdoc
```

BOSH can deploy and provision over cloud VMs. It was originally developed to deploy Cloud Foundry but it can also be used for provisioning of other software, eg Hadoop.

4- Special security groups rules are now created. These are for use by Cloud Foundry VMs and are to allow and control traffic on specific ports that are used by Cloud Foundry:

ssh

DIRECTION	ETHER TYPE	IP PROTOCOL	PORT RANGE	REMOTE
Egress	IPv4	Any	-	0.0.0.0/0 (CIDR)
Ingress	IPv4	UDP	68	0.0.0.0/0 (CIDR)
Ingress	IPv4	ICMP	-	0.0.0.0/0 (CIDR)
Egress	IPv6	Any	-	::/0 (CIDR)
Ingress	IPv4	TCP	22	0.0.0.0/0 (CIDR)

cf-public

DIRECTION	ETHER TYPE	IP PROTOCOL	PORT RANGE	REMOTE
Ingress	IPv4	TCP	443	0.0.0.0/0 (CIDR)
Ingress	IPv4	UDP	68	0.0.0.0/0 (CIDR)
Ingress	IPv4	TCP	80	0.0.0.0/0 (CIDR)
Egress	IPv4	Any	-	0.0.0.0/0 (CIDR)

Egress	IPv6	Any	-	::/0 (CIDR)

cf-private

DIRECTION	ETHER TYPE	IP PROTOCOL	PORT RANGE	REMOTE
Egress	IPv6	Any	-	::/0 (CIDR)
Egress	IPv4	Any	-	0.0.0.0/0 (CIDR)
Ingress	IPv4	UDP	68	0.0.0.0/0 (CIDR)
Ingress	IPv4	TCP	1-65535	bosh
Ingress	IPv4	UDP	3456-3457	0.0.0.0/0 (CIDR)

bosh

Direction	Ether Type	IP Protocol	Port Range	Remote
Ingress	IPv4	TCP	1-65535	bosh
Ingress	IPv4	TCP	25777	0.0.0.0/0 (CIDR)
Ingress	IPv4	TCP	25555	0.0.0.0/0 (CIDR)
Ingress	IPv4	TCP	25250	0.0.0.0/0 (CIDR)
Ingress	IPv4	TCP	6868	0.0.0.0/0 (CIDR)
Ingress	IPv4	TCP	4222	0.0.0.0/0 (CIDR)
Ingress	IPv4	UDP	68	0.0.0.0/0 (CIDR)
Ingress	IPv4	TCP	53	0.0.0.0/0 (CIDR)
Ingress	IPv4	UDP	53	0.0.0.0/0 (CIDR)
Egress	IPv4	Any	-	0.0.0.0/0 (CIDR)
Egress	IPv6	Any	-	::/0 (CIDR)

5- A key is now created with the name microbosh. It should be moved to the working directory, for example /cm/shared/mymanifests/

6- The manifest YML file is placed in the directory for the bosh utility to use for deployment:

- # mkdir /cm/shared/mymanifests
- # cat /cm/shared/mymanifests/manifest.yml

this is an example of how the manifest YML file should look like:

```
---
name: microbosh

network:
  type: manual
  vip: FLOATING-IP       # floating IP address
  ip: SUBNET-POOL-IP-ADDRESS     # IP from OpenStack internal network
  cloud_properties:
    net_id: NETWORK-UUID # Replace with your OpenStack internal network UUID

resources:
  persistent_disk: 20000
  cloud_properties:
    instance_type: m1.xlarge

cloud:
  plugin: openstack
  properties:
    openstack:
      auth_url: IDENTITY-API-ENDPOINT
      tenant: OPENSTACK-TENANT
      username: OPENSTACK-USERNAME
      api_key: OPENSTACK-PASSWORD
      default_key_name: microbosh
      private_key: /root/manifests/microbosh.pem
      default_security_groups: [bosh]

apply_spec:
  properties:
    director: {max_threads: 3}
    hm: {resurrector_enabled: true}
    ntp: [0.north-america.pool.ntp.org, 1.north-america.pool.ntp.org]
```

- The file can be filled with information extracted from the Openstack deployment that is in use. The information is available in the OpenStack RC file. The plaintext RC file can be downloaded from the OpenStack Horizon dashboard, under

Compute tab > Access & Security > APi Access >Click on download Openstack RC file

Running “grep ^export <filename>” on the file will extract what is needed.

OpenStack: How do I install Cloud Foundry on Bright OpenStack?

7-

The stemcell is a TAR GZ file with a qcow or raw image archived inside. It is downloaded by BOSH and is used to deploy the agent. So, it is used to deploy and configure the Cloud Foundry installation and deploy the Cloud Foundry VMs. There are a lot of stemcell types, eg. aws, openstack, vmware[qcow..raw]. The pre-configured images are only made for Cloud Foundry and BOSH.

The stemcell is now downloaded:

```
# cd /cm/shared/mymanifests; bosh public stemcells
```

```
# bosh download public stemcell \   bosh-stemcell-2889-openstack-kvm-ubuntu-trusty-go_agent.tgz
```

```
# bosh micro deployment manifest.yml
WARNING! Your target has been changed to https://10.2.13.25:25555!
Deployment set to ~/manifests/manifest.yml
```

```
# bosh micro deploy \ bosh-stemcell-2881-openstack-kvm-centos-go_agent.tgz
```

```
No 'bosh-deployments.yml' file found in current directory.
Is ~/my-micro-deployment a directory where you can save state? (type 'yes' to continue): yes
Deploying new micro BOSH instance ~/my-micro-deployment/manifest.yml to 'https://10.2.13.25:25555' (type 'yes' to continue): yes
  Started deploy micro bosh
...
```

```
# bosh target https://10.2.13.25:25555
Target set to 'microbosh'
Your username: admin
Enter password: *****
Logged in as 'admin'
```

At this point, deployment of Cloud Foundry components by the Micro Agent is completed.

OpenStack: How do I install Cloud Foundry on Bright OpenStack?

Deploying Cloud Foundry over Openstack:

bosh status

Config

/home/me/.bosh_config

Director

Name bosh_openstack

URL https://10.2.13.25:25555

Version 1.2710.0 (00000000)

User admin

UUID aba7d67c-115d-4f47-8315-2268c4ab900a

CPI openstack

dns enabled (domain_name: bosh)

compiled_package_cache disabled

snapshots disabled

Deployment

not set

- A stemcell is uploaded to BOSH (it will be used for Cloud Foundry VMs):

bosh upload stemcell \ ./bosh-stemcell-2754-openstack-kvm-ubuntu-trusty-go_agent.tgz

- The Cloud Foundry release repo is cloned from git:

git clone https://github.com/cloudfoundry/cf-release.git

cd cf-release; ./update

- Spiff is now installed:

Spiff is a validation tool to check the manifests templates that are used by Cloud Foundry. It is also used later on for the manifest that is to be used for deploying this Cloud foundry installation. It can be used to catch typos and correct variables, to generate a cleaned up manifest file for deployment.

1- go is installed from https://golang.org/doc/install

2- paths are set:

export GOPATH=/usr/local/go

export PATH=$PATH:$GOPATH/bin

3- spiff is installed with:

go get github.com/cloudfoundry-incubator/spiff

Now an example manifest file can be used as a template to generate the deployment manifest file:

-# cp spec/fixtures/openstack/cf-stub.yml

- The cf-stub.yml file can be edited according to need. At the end of this document are customizations that will be required.

- # ./generate_deployment_manifest openstack cf-stub.yml > cf-deployment.yml

Now, BOSH is pointed to the deployment file that is to be used to deploy Cloud Foundry:

- # bosh deployment cf-deployment.yml

In the cf-release directory this must be uploaded to the agent. The cf-release directory contains the Cloud Foundry files and code that will be deployed.

- # bosh upload release releases/cf-***.yml ###Check the latest stable

or

bosh create release

- # bosh upload release

Now it is ready to be deployed.

- # bosh deploy

The vms option can be used to verify that the deployment was successful:

bosh vms

```
+-----------------------------+---------+-------------------+---------------+
| Job/index                   | State   | Resource Pool     | IPs           |
+-----------------------------+---------+-------------------+---------------+
| nfs_server/0                | running | nfs_server        | 10.141.0.151  |
| ccdb/0                      | running | ccdb              | 10.141.0.152  |
| cloud_controller/0          | running | cloud_controller  | 10.141.0.153  |
| collector/0                 | running | collector         | 10.141.0.154  |
| health_manager/0            | running | health_manager    | 10.141.0.155  |
| nats/0                      | running | nats              | 10.141.0.156  |
| router/0                    | running | router            | 10.141.0.157  |
| syslog/0                    | running | syslog            | 10.141.0.158  |
| uaa/0                       | running | uaa               | 10.141.0.159  |
| uaadb/0                     | running | uaadb             | 10.141.0.160  |
| dea/0                       | running | dea               | 10.141.0.161  |
| saml_login/0                | running | saml_login        | 10.141.0.162  |
+-----------------------------+---------+-------------------+---------------+
```

OpenStack: How do I install Cloud Foundry on Bright OpenStack?

Deployment manifest customization:

DEPLOYMENT MANIFEST STUB CONTENTS	EDITING INSTRUCTIONS
director_uuid: DIRECTOR_UUID	# bosh status will view the BOSH Director UUID.
meta: openstack: net_id: net_id auth_url: auth_url tenant: openstack_tenant username: openstack_username api_key: openstack_api_key security_groups: [] floating_static_ips: - 0.0.0.0	Set net_id with an Openstack network UUID. You can get it from the Dashboard > Netowrks Set auth_url with the OpenStack keystone server. You can get it from the RC file. Set Openstack_tenant, openstack_username, and openstack_api_key with your OpenStack credentials. (RC File) Set cf or default or any other security groups you used to deploy by MicroBOSH to the security_groups array. Set the **floating_static_ips:**0.0.0.0 with a static IP address from your external network. You can allocate an IP using the dashboard > compute> access&security> floating IPs.
networks: - name: cf1 subnets: - cloud_properties: static: - 0.0.0.0 - 0.0.0.25	Set the **cf1 and static:**0.0.0.0 - 0.0.0.25 IP address range with a range of at least 26 IP addresses on your private network.
properties: cc: droplets: droplet_directory_key: the_key buildpacks: buildpack_directory_key: bd_key staging_upload_user: username staging_upload_password: password bulk_api_password: password db_encryption_key: the_key	Set **droplet_directory_key:**the key with the directory used to store droplets. Set **buildpack_directory_key:**bd_key with the directory used to store buildpacks. Set **staging_upload_user:**username with the account user name used to upload files to the Cloud Head node. Set **staging_upload_password:**password with the password of the account used to upload files to the Cloud Head node. Set **bulk_api_password:**password with the password used to access the bulk_api.

OpenStack: How do I install Cloud Foundry on Bright OpenStack?

	Set **db_encryption_key:**the_key with a secure key you generate used to encrypt database values.

ccdb:
  roles:
  - name: ccadmin
    password: admin_password
    tag: admin
	Replace the **roles: name:**ccadmin with the admin user name used to connect to the Cloud Head node.
	
	Set admin_password with the admin password.

databases:
  roles:
  - name: ccadmin
    password: ccadmin_password
  - name: uaaadmin
    password: uaaadmin_password
	Set ccadmin with the admin user name used to connect to the Cloud Head database.
	
	Set the ccadmin_password with the admin password.
	
	Set uaaadmin with the admin user name used to connect to the UAA database.
	
	Set uaaadmin_password with the admin password.

dea_next:
  disk_mb: 2048
  memory_mb: 1024
	Do not change these values.

domain: example.com	Replace cm.cluster or with your domain.
loggregator_endpoint: shared_secret: loggregator_endpoint_secret	Set the **share_secret:**loggregator_endpoint_secret with a secure secret.

nats:
  user: nats_user
  password: nats_password
	Set nats_user and nats_password with a secure user name and password for NATS access.

router:
  status:
    user: router_user
    password: router_password
	Set router_user and router_password with a secure user name and password for router access.

uaa:
  admin:
    client_secret: admin_secret
  batch:
    username: batch_username
    password: batch_password
	Set batch_username and batch_password with a secure user name and password.
	
	Generate secure keys for each secret.

OpenStack: How do I install Cloud Foundry on Bright OpenStack?

cc: client_secret: cc_client_secret clients: app-direct: secret: app-direct_secret developer_console: secret: developer_console_secret login: secret: login_client_secret notifications: secret: notification_secret servicesmgmt: secret: service_mgmt_secret space-mail: secret: space-mail_secret support-services: secret: support-services_secret 	
jwt: verification_key: vk signing_key: sk 	Set vk with an RSA Public Key. Set sk with an RSA Private Key.
scim: users: - admin|fakepassword|scim.write,scim.read,openid,cloud_controller.admin	Set fakepassword with an admin password.
uaadb: roles: - name: uaaadmin password: admin_password tag: admin 	Set uaaadmin with the admin user name used to connect to the UAA database. Set admin_password with the admin password.

Unique solution ID: #1280
Author: Frank Furter
Last update: 2015-07-20 19:23