vmware

Pivotal / VMware KuBo (Kubernetes on Bosh)

Lab Install Guide

Prepared by:
Technical Product Management team
VMware Cloud Native Business Unit

Version: 1.2

Document releases

Version	Description
1.0	Date: 08/15/2017 - Initial Version covering manual install of Kubo (release 0.5.0).
1.1	Date: 09/05/2017 – Adding sections: scale out K8s cluster and Harbor Integration.
1.2	Date 09/15/2017 – Updating the doc with Kubo release 0.7.0.

Table of Contents

1. Overview

This document contains information to manually install Kubo (Kubernetes on Bosh) on top of vSphere.

It is fully based on the information located here:
https://github.com/virtmerlin/doc-bosh-intro/blob/master/Readme.md

This guide assumes user has good knowledge and hands-on with vSphere. Please refer to appropriate documentation if needed.

This guide also assumes user has good understanding of Kubernetes technology and architecture. These aspects will not be covered in this document.

1.1 Acronyms

Acronym	Definition
KuBo	Kubernetes on Bosh
K8s	Kubernetes

1.2 Useful Links

These links provide detailed information about KuBo:

- Kubo Deployment:
  https://github.com/cloudfoundry-incubator/kubo-deployment
- Introduction to Bosh:
  https://github.com/virtmerlin/doc-bosh-intro/blob/master/Readme.md
- Harbor:
  https://vmware.github.io/harbor/

1.3 Binaries Versions

Binaries versions used for this lab are (latest versions available when writing this doc):

- vCenter: 6.5.0 U1
- ESXi: 6.5.0 U1
- Kubo release 0.7.0
- Harbor: 1.2.0

To download VMware vSphere (vCenter and ESXi) binaries, go to https://www.vmware.com/downloads

To download Kubo binaries, go to https://github.com/cloudfoundry-incubator/kubo-release/releases

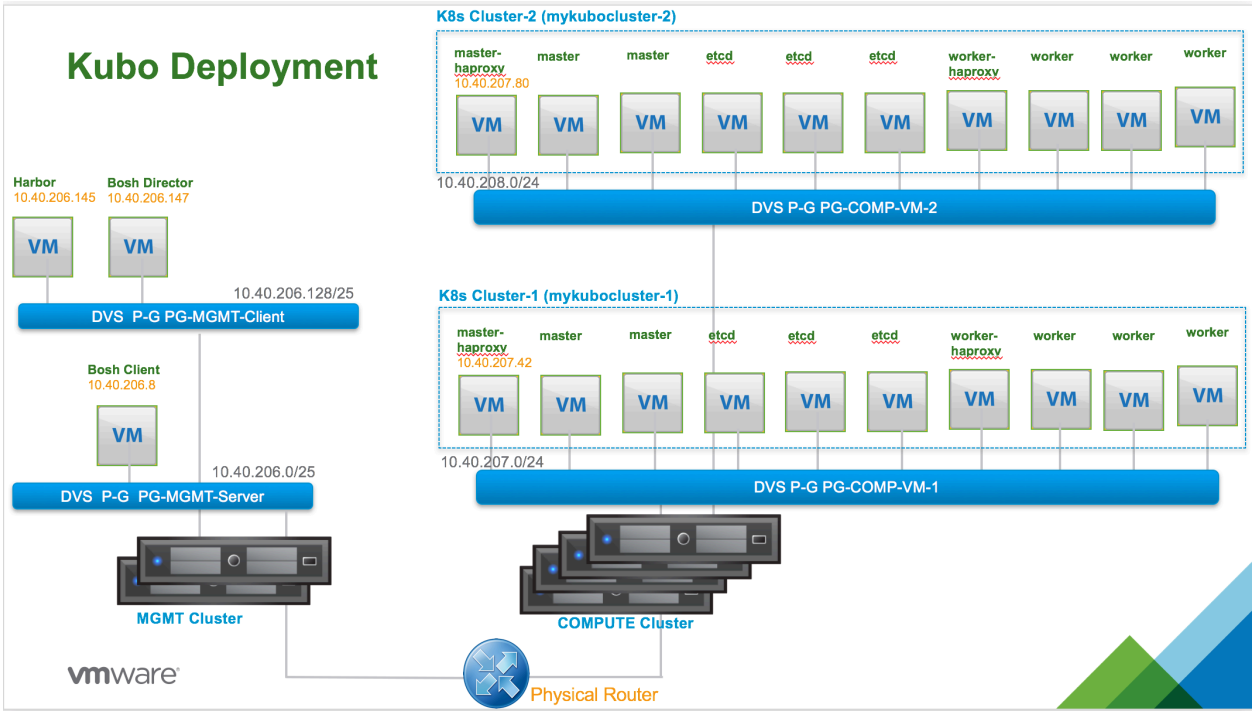
To download Harbor binaries, go to https://github.com/vmware/harbor/releases

2. Lab Topology

This section covers lab topology used for Kubo deployment.

2.1 Kubo Deployment

The following diagram illustrates a typical KuBo deployment:



Using a same and unique Bosh Director, it is possible to deploy multiple K8s cluster instances.

In this lab, we are going to deploy exactly the same topology as shown above.

2.1.1 MGMT and COMPUTE Cluster

2 Types of cluster will be used in this deployment:

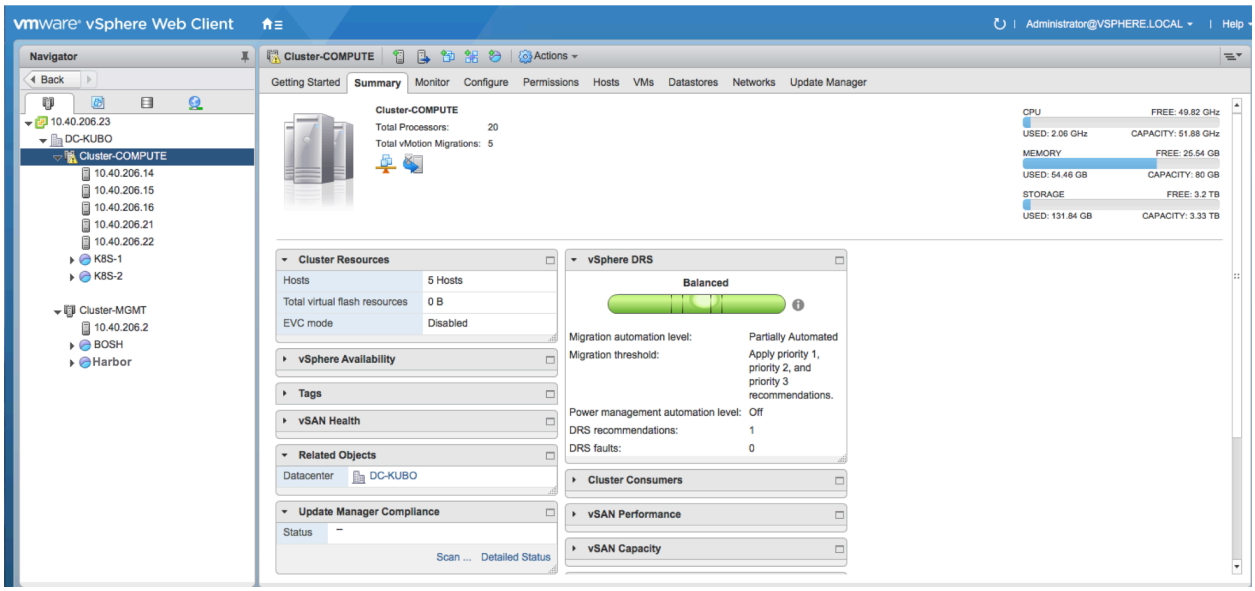
- MGMT Cluster:
    - Contains Bosh Client (Ubuntu VM) which is used to interact with Bosh Director. Bosh Client is connected to DVS Port-Group PG-Bosh-Client.

- Contains Bosh Director which is used to deploy K8s clusters (K8s Cluster-1 and Cluster-2 as shown in the above topology).
  Bosh Director is connected to DVS Port-Group PG-MGMT.
- Contains Harbor which is the Open Source private registry from VMware. Harbor supports secure or unsecure access mode and allows to perform vulnerabilities scan on stored images.

- MGMT Cluster has HA enabled to increase resiliency of the cluster so Bosh Client VM or Bosh Director VM can be restarted on a different host in case their current host face an outage.
- MGMT Cluster must have DRS enabled. Additionally, they must be configured with partially automated or fully automated modes otherwise Kubo deployment will fail.

- COMPUTE Cluster(s):
  - Contains all K8s clusters deployed by Bosh Director. Each K8s cluster can be instantiated on separate DVS Port-Groups as depicted in the diagram.
    Note: When deploying K8s cluster, user can specify multiple characteristics including number of master nodes, etcd and worker nodes. By default, the following deployment will be used: 3 masters, 3 etcd and 4 workers.
  - COMPUTE Cluster can have HA enabled if desired (mainly to protect K8s masters and etcd nodes).
  - COMPUTE Cluster must have DRS enabled. Additionally, they must be configured with partially automated or fully automated modes otherwise Kubo deployment will fail.

A snapshot of vCenter showing the 2 clusters is displayed below:



BOSH is a resource-pool in the MGMT Cluster that contains Bosh Client and Bosh Director VMs.

Harbor is a resource-pool in the MGMT Cluster that contains Harbor private registry VM.

K8S-1 and K8S-2 are resource-pools in the COMPUTE Cluster that contain K8s cluster-1 and cluster-2 VMs.

2.1.2 Virtual Networking characteristics

Bosh Client and Bosh Director are located on different DVS Port-Groups for security reasons (FW rules can be applied to protect traffic between the 2 VMs).

Each deployment of K8s cluster is using a separate DVS Port-Group for network isolation purposes. This design is recommended here; however, this is not mandatory.

2.1.3 Virtual Storage characteristics

Bosh Client, Bosh Director and K8s cluster instances can be installed on top of any datastore:

Local datastore, NFS datastore, vSAN datastore, and so on.

For the purpose of this document, we are going to use a NFS datastore.

2.2 Pre-requisites in term of IP connectivity

Pre-requisites in term of IP connectivity:

- Bosh Client should be able to reach Bosh Director.
- Bosh Director should be able to reach vCenter.
- Bosh Director should be able to reach K8s cluster VMs (any of them).

2.3 KuBo Deployment Overview

Here are the high-level steps to deploy KuBo:

KuBo Deployment – High Level Steps

1. Deploy Bosh Client
2. Deploy Bosh Director (using Bosh Client)
3. Deploy K8s clusters (from Bosh Director)

One or multiple K8s clusters can be deployed using the same Bosh Director.

Installation and configuration of vSphere is outside of this scope. Please refer to available materials on vmware.com if needed.

3. Kubo Installation

Caution:

We noticed that when copying/paste commands listed in this document, some issues may arise because of the formatting rendering. So double check the resulting copy/paste to make sure you get the whole command line and no new line or spaces are added while it shouldn't.

For instance, in the command below:

```
/usr/local/bin/bosh create-env bosh.yml \
--state=mystate.json \
```

there should be no space after the '\'.

Another typical format rendering issue is the hyphen character. Word sometimes convert it to '– '.
A command like "**bosh -v**" must use the hyphen character.

3.1 Deploy Bosh Client

Instantiate a VM for this purpose. In this lab, we are going to use Ubuntu 16.04 as OS.

If you plan to use different OS (like Windows or Mac OS), please refer to this link:
http://bosh.io/docs/cli-v2.html

- Install Bosh CLI:

```
root@bosh-client:~# wget https://s3.amazonaws.com/bosh-cli-artifacts/bosh-cli-2.0.28-linux-amd64
--2017-08-04 21:45:27--  https://s3.amazonaws.com/bosh-cli-artifacts/bosh-cli-2.0.28-linux-amd64
Resolving s3.amazonaws.com (s3.amazonaws.com)... 52.216.161.5
Connecting to s3.amazonaws.com (s3.amazonaws.com)|52.216.161.5|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 27190334 (26M) [binary/octet-stream]
Saving to: 'bosh-cli-2.0.28-linux-amd64'

bosh-cli-2.0.28-linux-
amd64                100%[==========================================================================================>]  25.93M   457KB/s    in 49s

2017-08-04 21:46:16 (545 KB/s) - 'bosh-cli-2.0.28-linux-amd64' saved [27190334/27190334]
```

```
root@bosh-client:~# ls
bosh-cli-2.0.28-linux-amd64
```

```
root@bosh-client:~# chmod +x ./bosh-cli-2.0.28-linux-amd64
```

```
root@bosh-client:~# mv bosh-cli-2.0.28-linux-amd64 /usr/local/bin/bosh
```

Check:

```
root@bosh-client:~# bosh -v

version 2.0.28-cb77557-2017-07-11T23:04:21Z

Succeeded
```

- Install OS specified dependencies for bosh create-env:

```
root@bosh-client:~# sudo apt-get install -y build-essential zlibc zlib1g-dev ruby ruby-dev openssl libxslt-dev libxml2-dev libssl-dev libreadline6 libreadline6-dev libyaml-dev libsqlite3-dev sqlite3
```

Check:

```
root@bosh-client:~# ruby -v
ruby 2.3.1p112 (2016-04-26) [x86_64-linux-gnu]
```

3.2 Deploy Bosh Director

- Git clone Bosh deployment:

```
root@bosh-client:~# git clone https://github.com/cloudfoundry/bosh-deployment

Cloning into 'bosh-deployment'...
remote: Counting objects: 1051, done.
remote: Compressing objects: 100% (26/26), done.
remote: Total 1051 (delta 9), reused 17 (delta 3), pack-reused 1021
Receiving objects: 100% (1051/1051), 183.08 KiB | 0 bytes/s, done.
Resolving deltas: 100% (569/569), done.
Checking connectivity... done.
```

```
root@bosh-client:~# ls
bosh-deployment
```

Check:

```
root@bosh-client:~# cd bosh-deployment/

root@bosh-client:~/bosh-deployment# ls

aws              bosh-lite.yml     credhub.yml  external-ip-not-recommended-uaa.yml  hm              local-
dns.yml  README.md     test.sh      virtualbox
azure            bosh.yml          dev          external-ip-not-recommended.yml      jumpbox-
user.yml     misc        runtime-configs  turbulence.yml  vsphere
bosh-lite-docker.yml  ci               docker       external-ip-with-registry-not-
recommended.yml  LICENSE          NOTICE       softlayer        uaa.yml      warden
bosh-lite-runc.yml    config-server.yml  docs         gcp                          local-bosh-
release.yml  openstack    syslog.yml       vcloud
```

- Deploy Bosh Director:

Create the following file named ‘deploy-bosh.sh’:

deploy-bosh.sh:	Description:
**/usr/local/bin/bosh create-env bosh.yml **	
**--state=mystate.json **	=> mystate.json contains the state of the deployment
**--vars-store=mycreds.yml **	=> mycreds.yml contains the credentials
**-o vsphere/cpi.yml **	=> CPI used (vSphere here)
**-o uaa.yml **	
**-o misc/powerdns.yml **	
**-o credhub.yml **	
**-v director_name=kubobosh **	=> Name of Bosh Director
**-v internal_cidr=10.40.206.128/25 **	=> Subnet CIDR
**-v internal_gw=10.40.206.253 **	=> Default GW
**-v internal_ip=10.40.206.147 **	=> IP of the VM
**-v network_name=PG-MGMT **	=> Name of the DVS (or VSS) port-group
**-v vcenter_dc=DC-KUBO **	=> DC name on vCenter
**-v vcenter_ds=NFS-DATASTORE **	=> datastore where VM will be instantiated
**-v vcenter_ip=10.40.206.23 **	=> vCenter IP address
**-v vcenter_user='administrator@vsphere.local' **	=> vCenter user name
**-v vcenter_password='VMware1!' **	=> vCenter password
**-v vcenter_templates=kubobosh-templates **	=> vCenter folder name (will be created by the script)
**-v vcenter_vms=kubobosh-vms **	=> vCenter folder name (will be created by the script)
**-v vcenter_disks=kubobosh-disks **	=> Datastore repository where K8s disks will be created
**-v vcenter_cluster=Cluster-MGMT **	=> ESXi cluster
-v dns_recursor_ip=10.20.20.1	=> DNS server IP address

Caution: when copy/paste the above script, make sure there is no space after the character '\' otherwise the shell would not interpret correctly the next command after '\'.

Run the script:

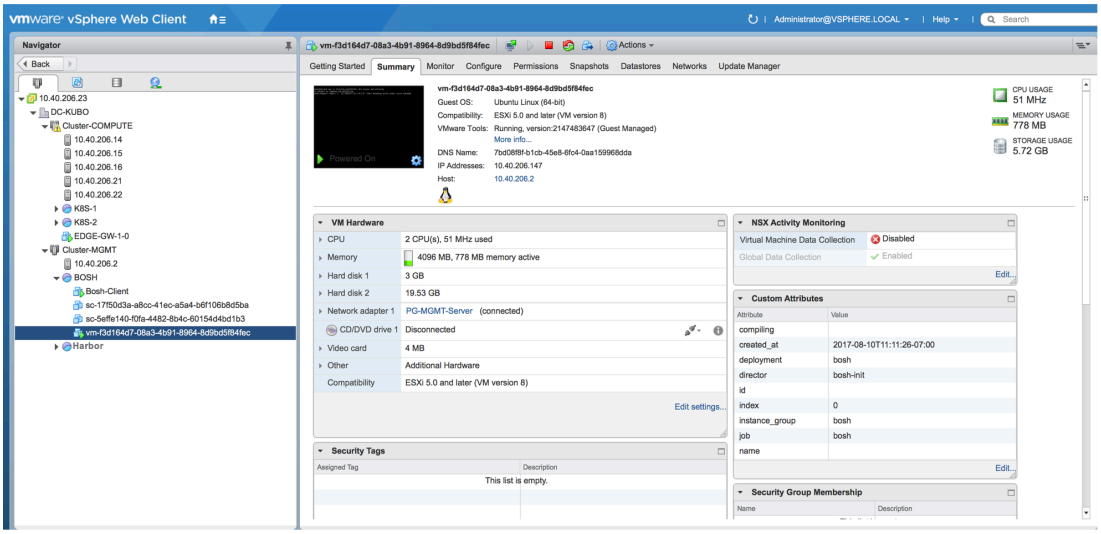
```
root@bosh-client:~/bosh-deployment# chmod + x ./deploy-bosh.sh

root@bosh-client:~/bosh-deployment# ./deploy-bosh.sh

Deployment manifest: '/root/bosh-deployment/bosh.yml'
Deployment state: 'mystate.json'

Started validating
  Downloading release 'bosh'... Skipped [Found in local cache] (00:00:00)
  Validating release 'bosh'... Finished (00:00:03)
  Downloading release 'bosh-vsphere-cpi'... Skipped [Found in local cache] (00:00:00)
  Validating release 'bosh-vsphere-cpi'... Finished (00:00:00)
  Downloading release 'uaa'... Skipped [Found in local cache] (00:00:00)
  Validating release 'uaa'... Finished (00:00:02)
  Downloading release 'credhub'... Skipped [Found in local cache] (00:00:00)
  Validating release 'credhub'... Finished (00:00:05)
  Validating cpi release... Finished (00:00:00)
  Validating deployment manifest... Finished (00:00:00)
  Downloading stemcell... Skipped [Found in local cache] (00:00:00)
  Validating stemcell... Finished (00:00:15)
Finished validating (00:00:28)

Started installing CPI
  Compiling package 'vsphere_cpi_ruby/14067294a0cd16a61646eedc3de4e9ed22d46076'... Finished (00:02:46)
  Compiling package 'vsphere_cpi/63a1a7a11086a7dbacedafe55636d958fa3ff64a'... Finished (00:01:07)
```

```
  Compiling package 'vsphere_cpi_mkisofs/72aac8fb0c0089065a00ef38a4e30d7d0e5a16ea'... Finished (00:03:21)
  Installing packages... Finished (00:00:00)
  Rendering job templates... Finished (00:00:00)
  Installing job 'vsphere_cpi'... Finished (00:00:00)
Finished installing CPI (00:07:16)

Starting registry... Finished (00:00:00)
Uploading stemcell 'bosh-vsphere-esxi-ubuntu-trusty-go_agent/3421.9'... Finished (00:00:57)



Started deploying

<SNIP>

Updating instance 'bosh/0'... Finished (00:01:35)

Waiting for instance 'bosh/0' to be running... Finished (00:01:17)

Running the post-start scripts 'bosh/0'... Finished (00:00:10)

Finished deploying (00:13:32)

Stopping registry... Finished (00:00:00)

Cleaning up rendered CPI jobs... Finished (00:00:00)

Succeeded
```

Bosh Director is now successfully deployed.

Check:

You should be able to see Bosh Director VM now created in the MGMT Cluster:



(You can manually move the VM under the BOSH resource-pool as shown in this diagram).

Note: the file named ‘mystate.json’ now contains all information related to this deployment. If you need to redeploy Bosh Director VM from a clean state, do not forget to delete first this file.

- Connect to Bosh Director from Bosh Client:

```
root@bosh-client:~/bosh-deployment# /usr/local/bin/bosh alias-env kubobosh -e 10.40.206.147 --ca-cert
<(/usr/local/bin/bosh int ./mycreds.yml --path /director_ssl/ca)

Using environment '10.40.206.147' as anonymous user
 Name     kubobosh
UUID     68d7ef59-3b9d-47ba-8bb2-ca93408aa01c
Version   262.3.0 (00000000)
CPI      vsphere_cpi
Features  compiled_package_cache: disabled
         config_server: enabled
         dns: enabled
         snapshots: disabled
User     (not logged in)
Succeeded
```

```
root@bosh-client:~/bosh-deployment# export BOSH_CLIENT=admin

root@bosh-client:~/bosh-deployment# export BOSH_CLIENT_SECRET=$(/usr/local/bin/bosh int ./mycreds.yml --path
/admin_password)
```

- Check:

```
root@bosh-client:~/bosh-deployment# /usr/local/bin/bosh -e kubobosh env


Using environment '10.40.206.147' as client 'admin'

Name     kubobosh
UUID     68d7ef59-3b9d-47ba-8bb2-ca93408aa01c
Version  262.3.0 (00000000)
CPI      vsphere_cpi
Features  compiled_package_cache: disabled
          config_server: enabled
          dns: enabled
          snapshots: disabled
User     admin

Succeeded
```

3.3 Deploy Kubernetes Cluster

- Git clone Kubo deployment:

```
root@bosh-client:~/bosh-deployment# cd ..

root@bosh-client:~# git clone https://github.com/cloudfoundry-incubator/kubo-deployment
Cloning into 'kubo-deployment'...
remote: Counting objects: 6649, done.
remote: Compressing objects: 100% (8/8), done.
remote: Total 6649 (delta 5), reused 7 (delta 5), pack-reused 6635
Receiving objects: 100% (6649/6649), 5.69 MiB | 1.76 MiB/s, done.
Resolving deltas: 100% (3602/3602), done.
Checking connectivity... done.
```

```
root@bosh-client:~# ls
bosh-deployment  kubo-deployment
```

```
root@bosh-client:~# cd kubo-deployment

root@bosh-client:~/kubo-deployment# ls
bin  bosh-deployment  configurations  CONTRIBUTING.md  docs  LICENSE  manifests  NOTICE  README.md  src
```

- Create ‘create-cloud-config.sh’ script file:

create-cloud-config.sh:	Description:
**/usr/local/bin/bosh int configurations/vsphere/cloud-config.yml **	
**-o manifests/ops-files/k8s-haproxy-static-ips-vsphere.yml **	=> Name of the output file
**-v director_name=kubobosh **	
**-v internal_cidr=10.40.207.0/24 **	
**-v internal_gw=10.40.207.253 **	=> Name of Bosh Director
**-v internal_ip=10.20.20.1 **	=> Network CIDR for K8s Cluster
**-v kubernetes_master_host=10.40.207.42 **	=> Default GW
**-v worker_haproxy_ip_addresses=10.40.207.43 **	=> DNS Server IP address
**-v reserved_ips=[10.40.207.1-10.40.207.41,10.40.207.44-10.40.207.68,10.40.207.103-10.40.207.254] **	=> IP address of master node1 (=VIP of HAproxy) => Reserved IP (will not be used)
**-v network_name=PG-COMP-VM-1 **	
**-v deployments_network=PG-COMP-VM-1 **	
**-v vcenter_cluster=Cluster-COMPUTE **	
-v vcenter_rp="K8S-1" > mycloudconfig-1.yml	=> Network Port-Group => Network Port-Group => ESxi Cluster where K8s Cluster will be hosted => Resource-Pool for K8s Cluster

Caution: when copy/paste the above script, make sure there is no space after the character '\' otherwise the shell would not interpret correctly the next command after '\'.

Note: Kubo release 0.7.0 has removed the manifest file:
manifests/ops-files/k8s_master_static_ip_vsphere.yml (the file defines only the variable kubernetes_master_host).
We need to use instead:
manifests/ops-files/k8s-haproxy-static-ips-vsphere.yml (the file defines 2 variables now: kubernetes_master_host and worker_haproxy_ip_addresses).

Note: syntax to exclude different ranges in reserved IP pool: "-v reserved_ips=[10.40.207.1-10.40.207.41,10.40.207.43-10.40.207.68,10.40.207.103-10.40.207.254]"

Note: K8s master IP (with HA proxy) and worker-haproxy IP should not be in the reserved IP pool (that's why reserved_ips=[10.40.207.1-10.40.207.41,10.40.207.44-10.40.207.68,10.40.207.103-10.40.207.254] => excludes the IP 10.40.207.42 and 10.40.207.43)

Run the script file:

```
root@bosh-client:~/kubo-deployment# chmod +x ./create-cloud-config.sh

root@bosh-client:~/kubo-deployment# ./create-cloud-config.sh
```

```
root@bosh-client:~/kubo-deployment# ls

bin  bosh-deployment  configurations  CONTRIBUTING.md  create-cloud-
config.sh  docs  LICENSE  manifests  mycloudconfig-1.yml  NOTICE  README.md  src
```

The content of the file 'mycoudconfig-1.yml' should look like this:

```
azs:
- cloud_properties:
    datacenters:
    - clusters:
      - Cluster-COMPUTE:
          resource_pool: K8S-1
  name: z1
compilation:
  az: z1
  network: PG-COMP-VM-1
  reuse_compilation_vms: true
  vm_type: worker
  workers: 4
disk_types:
- disk_size: 10240
  name: 10240
- disk_size: 5120
  name: 5120
networks:
- name: PG-COMP-VM-1
  subnets:
  - azs:
    - z1
    cloud_properties:
      name: PG-COMP-VM-1
    dns:
    - 10.20.20.1
    gateway: 10.40.207.253
    range: 10.40.207.0/24
    reserved:
    - 10.40.207.1-10.40.207.41
    - 10.40.207.43-10.40.207.68
    - 10.40.207.103-10.40.207.254
    static:
    - 10.40.207.42
    - 10.40.207.43
  type: manual
vm_types:
- cloud_properties:
    cpu: 1
    disk: 20480
    ram: 4096
  name: common
- cloud_properties:
    cpu: 1
    disk: 20480
    ram: 4096
  name: master
- cloud_properties:
    cpu: 1
    disk: 102400
    ram: 8192
  name: worker
```

The file named ‘mycoudconfig-1.yml’ contains characteristics for K8s cluster deployment like hardware specifications for the VM. You can modify the content of the file to match with your environment.

- Update cloud config:

```
root@bosh-client: ~/kubo-deployment# bosh -e kubobosh update-cloud-config mycloudconfig-1.yml

Using environment '10.40.206.147' as client 'admin'
<SNIP>

Continue? [yN]: y

Succeeded
```

- Create 'create-kubo-deployment.sh' script file:

create-kubo-deployment.sh:	Description:
/usr/local/bin/bosh int manifests/kubo.yml ** **-o manifests/ops-files/master-haproxy-vsphere.yml ** **-o manifests/ops-files/worker-haproxy.yml ** **-v deployments_network= PG-COMP-VM-1 ** **-v kubo-admin-password="VMware1!" ** **-v kubelet-password="VMware1!" ** **-v kubernetes_master_port=443 ** **-v kubernetes_master_host=10.40.207.42 ** **-v deployment_name=mykubocluster-1 ** **-v worker_haproxy_tcp_frontend_port=1234 ** **-v worker_haproxy_tcp_backend_port=4231 > mykubo-1.yml	=> Network Port-Group => Password for kubectl admin password => Password for kubelet => K8s api-server listening on HTTPS => IP of K8s master node (=VIP for HAproxy) => Name of the deployment

Caution: when copy/paste the above script, make sure there is no space after the character '\' otherwise the shell would not interpret correctly the next command after '\'.

Run the script file:

```
root@bosh-client:~/kubo-deployment# chmod +x ./create-kubo-deployment.sh

root@bosh-client:~/kubo-deployment# ./create-kubo-deployment.sh
```

```
root@bosh-client:~/kubo-deployment# ls
bin  bosh-deployment  configurations  CONTRIBUTING.md  create-cloud-config.sh  create-kubo-
deployment.sh  docs  LICENSE  manifests  mycloudconfig-1.yml  mykubo-1.yml  NOTICE  README.md  src
```

The file named 'mykubo-1.yml' contains a detailed declaration of the K8s cluster deployment. It should look like this:

```
instance_groups:
- azs:
  - z1
```

```
  instances: 3
  jobs:
  - name: etcd
    properties:
      etcd:
        peer_require_ssl: false
        require_ssl: false
    release: kubo-etcd
  name: etcd
  networks:
  - name: PG-COMP-VM-1
  persistent_disk_type: 5120
  stemcell: trusty
  vm_type: common
- azs:
  - z1
  instances: 2
  jobs:
  - name: cloud-provider
    properties: {}
    release: kubo
  - name: flanneld
    release: kubo
  - name: kubernetes-api
    properties:
      admin-password: VMware1!
      admin-username: admin
      backend_port: 8443
      kubelet-password: VMware1!
      port: 443
      tls:
        kubernetes: ((tls-kubernetes))
    release: kubo
  - name: kubeconfig
    properties:
      kubelet-password: VMware1!
      kubernetes-api-url: https://10.40.207.42:443
      tls:
        kubernetes: ((tls-kubernetes))
    release: kubo
  - name: kubernetes-controller-manager
    properties: {}
    release: kubo
  - name: kubernetes-scheduler
    release: kubo
  - name: kubernetes-system-specs
    properties:
      kubernetes-api-url: https://10.40.207.42:443
    release: kubo
  name: master
  networks:
  - name: PG-COMP-VM-1
  stemcell: trusty
  vm_type: master
- azs:
  - z1
  instances: 1
  jobs:
```

```
  - consumes:
      tcp_backend:
        from: master_haproxy
    name: haproxy
    properties:
      ha_proxy:
        disable_http: true
        tcp_link_port: 9999
    release: haproxy
  name: master-haproxy
  networks:
  - default:
    - dns
    - gateway
    name: PG-COMP-VM-1
    static_ips:
    - 10.40.207.42
  stemcell: trusty
  vm_type: common
- azs:
  - z1
  instances: 3
  jobs:
  - name: flanneld
    release: kubo
  - name: docker
    properties:
      docker:
        flannel: true
        ip_masq: false
        iptables: false
        log_level: error
        storage_driver: overlay
      env: {}
    release: docker
  - name: kubeconfig
    properties:
      kubelet-password: VMware1!
      kubernetes-api-url: https://10.40.207.42:443
      tls:
        kubernetes: ((tls-kubernetes))
    release: kubo
  - name: cloud-provider
    properties: {}
    release: kubo
  - name: kubelet
    properties:
      backend_port: 4231
      kubernetes-api-url: https://10.40.207.42:443
      port: 1234
      tls:
        kubelet: ((tls-kubelet))
    release: kubo
  - name: kubernetes-proxy
    properties:
      kubernetes-api-url: https://10.40.207.42:443
    release: kubo
  name: worker
```

```
  networks:
  - name: PG-COMP-VM-1
  persistent_disk_type: 10240
  stemcell: trusty
  vm_type: worker
- azs:
  - z1
  instances: 1
  jobs:
  - consumes:
      tcp_backend:
        from: worker_haproxy
    name: haproxy
    properties:
      ha_proxy:
        disable_http: true
        tcp_link_port: 9999
    release: haproxy
  name: worker-haproxy
  networks:
  - default:
    - dns
    - gateway
    name: PG-COMP-VM-1
  stemcell: trusty
  vm_type: common
name: mykubocluster-1
releases:
- name: kubo-etcd
  sha1: 91f7a6592ee6c5242854f3654fe786574e816ffc
  url: https://storage.googleapis.com/kubo-public/kubo-etcd-2-ubuntu-trusty-3421.11-20170721-091603-591124789-
20170721091609.tgz
  version: 2
- name: kubo
  version: latest
- name: docker
  sha1: 0ac80f013cc686047cdd7ccc428a8784c5e691bc
  url: https://storage.googleapis.com/kubo-public/docker-28.0.1-ubuntu-trusty-3421.11-20170720-164316-303456764-
20170720164324.tgz
  version: 28.0.1
- name: haproxy
  sha1: 19f705d4958b24a4c49e9ec8770b5bee4ba454be
  url: https://storage.googleapis.com/kubo-public/haproxy-8.3.0-ubuntu-trusty-3421.11-20170721-091831-348952426-
20170721091831.tgz
  version: latest
stemcells:
- alias: trusty
  os: ubuntu-trusty
  version: "3421.11"
update:
  canaries: 1
  canary_watch_time: 10000-300000
  max_in_flight: 1
  serial: true
  update_watch_time: 10000-300000
variables:
- name: kubo_ca
  options:
```

```
    common_name: ca
    is_ca: true
  type: certificate
- name: tls-kubelet
  options:
    alternative_names:
    - 10.40.207.42
    ca: kubo_ca
    common_name: 10.40.207.42
  type: certificate
- name: tls-kubernetes
  options:
    alternative_names:
    - 10.40.207.42
    - 10.100.200.1
    - kubernetes
    - kubernetes.default
    - kubernetes.default.svc
    - kubernetes.default.svc.cluster.local
    ca: kubo_ca
    common_name: 10.40.207.42
  type: certificate
```

- Upload the stemcell:

```
root@bosh-client:~/kubo-deployment# /usr/local/bin/bosh -e kubobosh upload-stemcell
https://s3.amazonaws.com/bosh-core-stemcells/vsphere/bosh-stemcell-3421.11-vsphere-esxi-ubuntu-trusty-go_agent.tgz
Using environment '10.40.206.147' as client 'admin'

Task 44

23:36:09 | Update stemcell: Downloading remote stemcell (00:29:26)
00:05:35 | Update stemcell: Extracting stemcell archive (00:00:05)
00:05:40 | Update stemcell: Verifying stemcell manifest (00:00:00)
00:05:43 | Update stemcell: Checking if this stemcell already exists (00:00:00)
00:05:43 | Update stemcell: Uploading stemcell bosh-vsphere-esxi-ubuntu-trusty-go_agent/3421.11 to the cloud (already exists, skipped) (00:00:00)
00:05:43 | Update stemcell: Save stemcell bosh-vsphere-esxi-ubuntu-trusty-go_agent/3421.11 (sc-5effe140-f0fa-4482-8b4c-60154d4bd1b3) (already exists, skipped) (00:00:00)

Started  Tue Aug  8 23:36:09 UTC 2017
Finished Wed Aug  9 00:05:43 UTC 2017
Duration 00:29:34

Task 44 done

Succeeded
```

Check:

```
root@bosh-client:~/kubo-deployment#  /usr/local/bin/bosh -e kubobosh stemcells

Using environment '10.40.206.147' as client 'admin'

Name                                        Version   OS          CPI  CID
bosh-vsphere-esxi-ubuntu-trusty-go_agent  3421.11*  ubuntu-trusty  -   sc-5effe140-f0fa-4482-8b4c-60154d4bd1b3
(*) Currently deployed

1 stemcells

Succeeded
```

Note: KuBo release 0.7.0 uses Ubuntu 3421.11. So, we really need this particular version number of the Ubuntu OS.

- Upload Bosh Kubo release:

```
root@bosh-client:~/kubo-deployment# wget https://github.com/cloudfoundry-incubator/kubo-release/releases/download/v0.7.0/kubo-release-0.7.0.tgz

--2017-09-11 21:07:26--  https://github.com/cloudfoundry-incubator/kubo-release/releases/download/v0.7.0/kubo-release-0.7.0.tgz
Resolving github.com (github.com)... 192.30.255.112, 192.30.255.113
Connecting to github.com (github.com)|192.30.255.112|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://github-production-release-asset-2e65be.s3.amazonaws.com/68842993/14fa62a6-93d4-11e7-8af0-d9257b637ad9?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIWNJYAX4CSVEH53A%2F20170912%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20170912T040727Z&X-Amz-Expires=300&X-Amz-Signature=84fd7d3ac6ecef9a6b182a184728dd1b36345ac53dea09d181f550c484a55eb1&X-Amz-SignedHeaders=host&actor_id=0&response-content-disposition=attachment%3B%20filename%3Dkubo-release-0.7.0.tgz&response-content-type=application%2Foctet-stream [following]
--2017-09-11 21:07:27--  https://github-production-release-asset-2e65be.s3.amazonaws.com/68842993/14fa62a6-93d4-11e7-8af0-d9257b637ad9?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIWNJYAX4CSVEH53A%2F20170912%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20170912T040727Z&X-Amz-Expires=300&X-Amz-Signature=84fd7d3ac6ecef9a6b182a184728dd1b36345ac53dea09d181f550c484a55eb1&X-Amz-SignedHeaders=host&actor_id=0&response-content-disposition=attachment%3B%20filename%3Dkubo-release-0.7.0.tgz&response-content-type=application%2Foctet-stream
Resolving github-production-release-asset-2e65be.s3.amazonaws.com (github-production-release-asset-2e65be.s3.amazonaws.com)... 52.216.1.192
Connecting to github-production-release-asset-2e65be.s3.amazonaws.com (github-production-release-asset-2e65be.s3.amazonaws.com)|52.216.1.192|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 483862757 (461M) [application/octet-stream]
Saving to: 'kubo-release-0.7.0.tgz'

kubo-release-0.7.0.tgz          100%[=====================================================================>]
461.45M   540KB/s    in 21m 7s

2017-09-11 21:28:35 (373 KB/s) - 'kubo-release-0.7.0.tgz' saved [483862757/483862757]
```

```
root@bosh-client:~/kubo-deployment# # /usr/local/bin/bosh -e kubobosh upload-release kubo-release-0.7.0.tgz

Using environment '10.40.206.147' as client 'admin'

############################################################# 100.00% 53.51 MB/s 8s
Task 1

04:34:13 | Extracting release: Extracting release (00:00:06)
04:34:19 | Verifying manifest: Verifying manifest (00:00:00)
04:34:19 | Resolving package dependencies: Resolving package dependencies (00:00:00)
04:34:19 | Creating new packages: cni/fb66deef2826ccd6c6c135dbc915094e6cef2ab6 (00:00:01)
04:34:20 | Creating new packages: etcdctl/35165b48a3100f6f0e4af03c211f913dcf0055b2 (00:00:00)
04:34:20 | Creating new packages: flanneld/69e5913473152bb3a97fee5ad5f237cb6b3becba (00:00:01)
04:34:21 | Creating new packages: golang/dd608878e7f3335773a316e718b07a7e5c3cd32b (00:00:02)
04:34:23 | Creating new packages: govc/02be57c077b9ed2a47481deb5f5dfa0d295ad242 (00:00:00)
04:34:23 | Creating new packages: jq/a8a92d1eb93b806ff9e4f9e8daab4d0dec04b962 (00:00:00)
04:34:23 | Creating new packages: kubernetes/85d418a7debb01fc4825c28cef3da558757af57c (00:00:12)
04:34:35 | Creating new packages: pid_utils/96db60d4d683939fd187297035544c340e75d9a4 (00:00:00)
04:34:35 | Creating new packages: route-sync/4d89a033084648e6143f4d94cf4a4c210f0bedea (00:00:01)
04:34:36 | Creating new packages: socat/44be0e2da76a8c3db0409993b168aa26d4bc3cd4 (00:00:00)
04:34:36 | Creating new jobs: cloud-provider/763a705f0ba0e860f531acc2c9048cb61406f6d2 (00:00:00)
04:34:36 | Creating new jobs: flanneld/957cfb48b203e4c2de73fa4042d63be0597ed3b6 (00:00:00)
04:34:36 | Creating new jobs: kubeconfig/a231cdac5d0b3d6767b3183658841ffc2615b6ea (00:00:00)
04:34:36 | Creating new jobs: kubelet/1cd437ae9cee2e70b06d29972eb32de325fd9b4f (00:00:00)
04:34:36 | Creating new jobs: kubernetes-api/c4675b7a89462d8f5a2362a83bfc4006d264f265 (00:00:00)
04:34:36 | Creating new jobs: kubernetes-api-route-registrar/d752a6dcd94d8ed90ce5a955f8f2664100b1ad04 (00:00:00)
04:34:36 | Creating new jobs: kubernetes-controller-manager/1bdf4f211ee421dda9e9c9234959993154759559
(00:00:00)
04:34:36 | Creating new jobs: kubernetes-proxy/5868e32de3928cf21907bc80dedbe2a942c02d46 (00:00:00)
04:34:36 | Creating new jobs: kubernetes-scheduler/466bc37ad7132f3131e3ce36994b9d06616f31ba (00:00:00)
04:34:36 | Creating new jobs: kubernetes-system-specs/a1f68df09ab9a6b9dd7f6583425024a208ca17f1 (00:00:00)
04:34:36 | Creating new jobs: route-sync/d70357cfdf5207f60d8be522a8d6caf33f41f944 (00:00:00)
04:34:36 | Creating new jobs: syslog-forwarding-setup/844a26588f86525a3a677cd664e4875619781245 (00:00:00)
04:34:36 | Release has been created: kubo/0.7.0 (00:00:00)

Started  Tue Sep 12 04:34:13 UTC 2017
Finished Tue Sep 12 04:34:36 UTC 2017
Duration 00:00:23

Task 1 done

Succeeded
```

Check:

```
root@bosh-client:~/kubo-deployment# # /usr/local/bin/bosh -e kubobosh releases

Using environment '10.40.206.147' as client 'admin'

Name  Version  Commit Hash
kubo  0.7.0    1017224

(*) Currently deployed
(+) Uncommitted changes
```

```
1 releases

Succeeded
```

- Deploy K8s Cluster:

```
root@bosh-client: ~/kubo-deployment#  /usr/local/bin/bosh -e kubobosh -d mykubocluster-1 deploy mykubo-1.yml

Using environment '10.40.206.147' as client 'admin'

Using deployment 'mykubocluster-1'

Release 'kubo-etcd/2' already exists.

Release 'docker/28.0.1' already exists.

Task 327

17:34:18 | Downloading remote release: Downloading remote release (00:00:01)
17:34:19 | Verifying remote release: Verifying remote release (00:00:00)
17:34:19 | Extracting release: Extracting release (00:00:00)
17:34:19 | Verifying manifest: Verifying manifest (00:00:00)
17:34:19 | Resolving package dependencies: Resolving package dependencies (00:00:00)
17:34:20 | Processing 2 existing jobs: Processing 2 existing jobs (00:00:00)
17:34:20 | Compiled Release has been created: haproxy/8.3.0 (00:00:00)

Started  Tue Aug 15 17:34:18 UTC 2017
Finished Tue Aug 15 17:34:20 UTC 2017
Duration 00:00:02

Task 327 done
<SNIP>

Continue? [yN]: y

Continue? [yN]: y

Task 328
17:34:22 | Preparing deployment: Preparing deployment (00:00:03)
17:34:28 | Preparing package compilation: Finding packages to compile (00:00:00)
17:34:28 | Creating missing vms: etcd/a1e0f121-ec1e-47fd-926b-8699fd87fdf5 (0)
17:34:28 | Creating missing vms: etcd/2e97d38a-76f1-448d-8fb3-d2423623fef5 (1)
17:34:28 | Creating missing vms: etcd/68e20a9f-8cad-42ca-8ffa-5e1ebe6552d5 (2)
17:34:28 | Creating missing vms: master/5ea5bcf9-e647-45ad-9e63-2286f517b14f (1)
17:34:28 | Creating missing vms: worker/67a4670a-667e-4bfd-9f94-8476f1952296 (2)
17:34:28 | Creating missing vms: master-haproxy/7b51590a-73aa-43ba-b9e0-14e54a22cee8 (0)
17:34:28 | Creating missing vms: worker/1c0ceb4b-9cf2-4772-8db9-89a9e64e7a66 (0)
17:34:28 | Creating missing vms: worker/9556df08-cd39-4c0b-80cd-e9e6e44a93cf (1)
17:34:28 | Creating missing vms: worker-haproxy/573f4cde-ec43-424b-b92f-b13dd3b25dbb (0)
17:34:28 | Creating missing vms: master/acf2a254-cc96-407a-a708-6c9bdf1fd608 (0)
17:37:40 | Creating missing vms: worker/1c0ceb4b-9cf2-4772-8db9-89a9e64e7a66 (0) (00:03:12)
17:37:41 | Creating missing vms: etcd/68e20a9f-8cad-42ca-8ffa-5e1ebe6552d5 (2) (00:03:13)
17:37:41 | Creating missing vms: worker/67a4670a-667e-4bfd-9f94-8476f1952296 (2) (00:03:13)
```

```
17:37:41 | Creating missing vms: worker/9556df08-cd39-4c0b-80cd-e9e6e44a93cf (1) (00:03:13)
17:37:41 | Creating missing vms: master/5ea5bcf9-e647-45ad-9e63-2286f517b14f (1) (00:03:13)
17:37:42 | Creating missing vms: master/acf2a254-cc96-407a-a708-6c9bdf1fd608 (0) (00:03:14)
17:37:42 | Creating missing vms: etcd/2e97d38a-76f1-448d-8fb3-d2423623fef5 (1) (00:03:14)
17:37:42 | Creating missing vms: etcd/a1e0f121-ec1e-47fd-926b-8699fd87fdf5 (0) (00:03:14)
17:37:52 | Creating missing vms: worker-haproxy/573f4cde-ec43-424b-b92f-b13dd3b25dbb (0) (00:03:24)
17:37:52 | Creating missing vms: master-haproxy/7b51590a-73aa-43ba-b9e0-14e54a22cee8 (0) (00:03:24)
17:37:52 | Updating instance etcd: etcd/a1e0f121-ec1e-47fd-926b-8699fd87fdf5 (0) (canary) (00:01:06)
17:38:58 | Updating instance etcd: etcd/2e97d38a-76f1-448d-8fb3-d2423623fef5 (1) (00:00:59)
17:39:57 | Updating instance etcd: etcd/68e20a9f-8cad-42ca-8ffa-5e1ebe6552d5 (2) (00:00:59)
17:40:56 | Updating instance master: master/acf2a254-cc96-407a-a708-6c9bdf1fd608 (0) (canary) (00:01:32)
17:42:28 | Updating instance master: master/5ea5bcf9-e647-45ad-9e63-2286f517b14f (1) (00:01:17)
17:43:45 | Updating instance master-haproxy: master-haproxy/7b51590a-73aa-43ba-b9e0-14e54a22cee8 (0) (canary)
17:44:18 | Updating instance worker: worker/1c0ceb4b-9cf2-4772-8db9-89a9e64e7a66 (0) (canary) (00:05:25)
17:49:43 | Updating instance worker: worker/67a4670a-667e-4bfd-9f94-8476f1952296 (2) (00:03:13)
17:52:56 | Updating instance worker: worker/9556df08-cd39-4c0b-80cd-e9e6e44a93cf (1) (00:02:05)
17:55:01 | Updating instance worker-haproxy: worker-haproxy/573f4cde-ec43-424b-b92f-b13dd3b25dbb (0) (canary)
(00:00:31)

Started  Tue Aug 15 17:34:18 UTC 2017
Finished Tue Aug 15 17:55:32 UTC 2017
Duration 00:21:14

Task 328 done

Succeeded
```

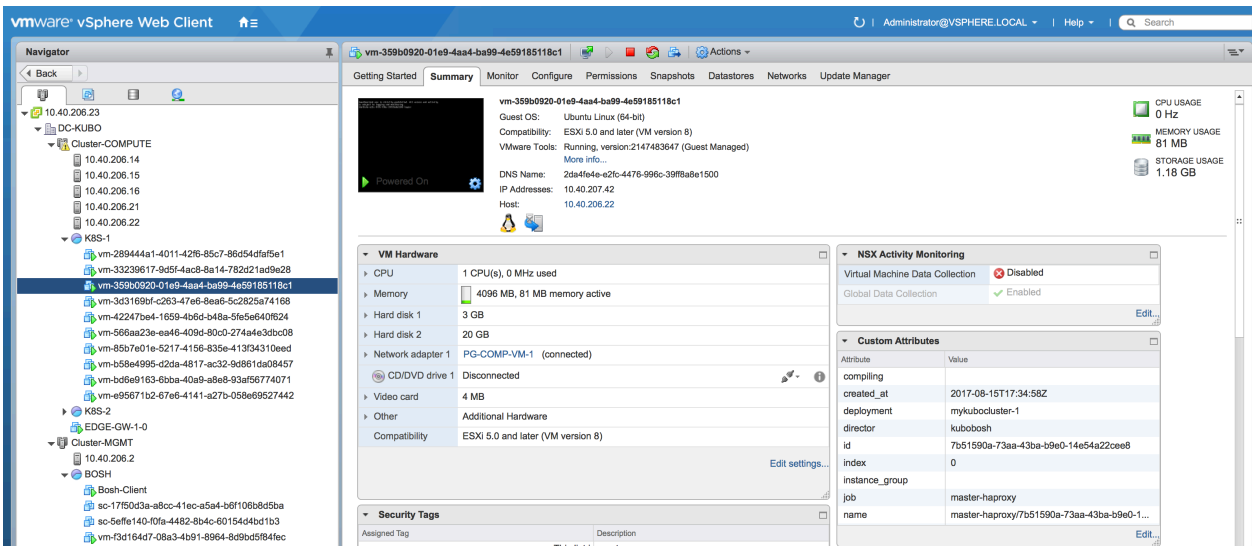
Check:

```
root@bosh-client:~/kubo-deployment# /usr/local/bin/bosh -e kubobosh deployments

Using environment '10.40.206.147' as client 'admin'


Name          Release(s)     Stemcell(s)                                  Team(s)  Cloud Config

mykubocluster-1  kubo-etcd/2    bosh-vsphere-esxi-ubuntu-trusty-go_agent/3421.11  -        latest
             kubo/0.7.0
             docker/28.0.1
             haproxy/8.4.0


1 deployments


Succeeded
```

```
root@bosh-client:~/kubo-deployment# /usr/local/bin/bosh -e kubobosh instances

Using environment '10.40.206.147' as client 'admin'
```

```
Task 333. Done

Deployment 'mykubocluster-1'

Instance                                        Process State  AZ  IPs
etcd/2e97d38a-76f1-448d-8fb3-d2423623fef5       running        z1  10.40.207.70
etcd/68e20a9f-8cad-42ca-8ffa-5e1ebe6552d5       running        z1  10.40.207.71
etcd/a1e0f121-ec1e-47fd-926b-8699fd87fdf5       running        z1  10.40.207.69
master-haproxy/7b51590a-73aa-43ba-b9e0-14e54a22cee8  running   z1  10.40.207.42
master/5ea5bcf9-e647-45ad-9e63-2286f517b14f     running        z1  10.40.207.73
master/acf2a254-cc96-407a-a708-6c9bdf1fd608     running        z1  10.40.207.72
worker-haproxy/573f4cde-ec43-424b-b92f-b13dd3b25dbb  running   z1  10.40.207.77
worker/1c0ceb4b-9cf2-4772-8db9-89a9e64e7a66     running        z1  10.40.207.74
worker/67a4670a-667e-4bfd-9f94-8476f1952296     running        z1  10.40.207.75
worker/9556df08-cd39-4c0b-80cd-e9e6e44a93cf     running        z1  10.40.207.76

10 instances

Succeeded
```

On vCenter, you should be able to see the K8s Cluster deployed in COMPUTE cluster, under the specified resource-pool:



3.4 Connect to K8s Cluster

- Install credhub CLI on Bosh Client VM:

```
root@bosh-client:~# ls
bosh-deployment  kubo-deployment
```

```
root@bosh-client:~# mkdir credhub-cli

root@bosh-client:~# cd credhub-cli/
```

```
root@bosh-client:~/credhub-cli# wget https://github.com/cloudfoundry-incubator/credhub-cli/releases/download/1.4.0/credhub-linux-1.4.0.tgz

--2017-08-08 22:16:32--  https://github.com/cloudfoundry-incubator/credhub-cli/releases/download/1.4.0/credhub-linux-1.4.0.tgz
Resolving github.com (github.com)... 192.30.253.112, 192.30.253.113
Connecting to github.com (github.com)|192.30.253.112|:443... connected.
HTTP request sent, awaiting response... 302 Found
[following])... 52.216.22.43

<SNIP>

Connecting to github-production-release-asset-2e65be.s3.amazonaws.com (HTTP request sent, awaiting response... 200 OK
Length: 2701606 (2.6M) [application/octet-stream]
Saving to: 'credhub-linux-1.4.0.tgz'

credhub-linux-
1.4.0.tgz                   100%[=========================================================================================>]   2.58M  1.33MB/s    in 1.9s

2017-08-08 22:16:35 (1.33 MB/s) - 'credhub-linux-1.4.0.tgz' saved [2701606/2701606]
```

```
root@bosh-client:~/credhub-cli# ls
credhub-linux-1.4.0.tgz
```

```
root@bosh-client:~/credhub-cli# tar -xvf credhub-linux-1.4.0.tgz
./
./credhub
```

```
root@bosh-client:~/credhub-cli# ls
credhub  credhub-linux-1.4.0.tgz


root@bosh-client:~/credhub-cli# ls -l
total 10480
-rwxr-xr-x 1 root root 8025076 Jul 26 15:08 credhub
-rw-r--r-- 1 root root 2701606 Jul 26 15:08 credhub-linux-1.4.0.tgz
```

```
root@bosh-client:~/credhub-cli# cp credhub /usr/local/bin/
```

Check:

```
root@bosh-client:~/credhub-cli# credhub
Usage:
  credhub [OPTIONS] [command]

Application Options:
      --version  Version of CLI and targeted CredHub API
      --token    Return your current CredHub authorization token

Help Options:
  -h, --help     Show this help message

Available commands:
  api        Set the CredHub API target to be used for subsequent commands (aliases: a)
  delete     Delete a credential value (aliases: d)
  find       Find stored credentials based on query parameters (aliases: f)
  generate   Set a credential with a generated value (aliases: n)
  get        Get a credential value (aliases: g)
  import     Set multiple credential values (aliases: i)
  login      Authenticate user with CredHub (aliases: l)
  logout     Discard authenticated user session (aliases: o)
  regenerate Set a credential with a generated value using the same attributes as the stored value (aliases: r)
  set        Set a credential with a provided value (aliases: s)
```

- Install kubectl CLI on Bosh Client VM:

```
root@bosh-client:~/credhub-cli# cd ..
```

```
root@bosh-client:~# ls
 bosh-deployment  credhub-cli   kubo-deployment
```

```
root@bosh-client:~# mkdir kubectl-cli
root@bosh-client:~# cd kubectl-cli/
```

```
root@bosh-client:~/kubectl-cli#  curl -LO https://storage.googleapis.com/kubernetes-release/release/$(curl -s
https://storage.googleapis.com/kubernetes-release/release/stable.txt)/bin/linux/amd64/kubectl


  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 68.9M  100 68.9M    0     0  24.7M      0  0:00:02  0:00:02 --:--:-- 24.7M
```

```
root@bosh-client:~/kubectl-cli# ls -l
total 70644
-rw-r--r-- 1 root root 72337322 Aug  8 22:22 kubectl
```

```
root@bosh-client:~/kubectl-cli# chmod +x ./kubectl
```

```
root@bosh-client:~/kubectl-cli# cp ./kubectl /usr/local/bin/kubectl
```

Check:

```
root@bosh-client:~/kubectl-cli# kubectl
kubectl controls the Kubernetes cluster manager.

Find more information at https://github.com/kubernetes/kubernetes.

Basic Commands (Beginner):
  create        Create a resource by filename or stdin
  expose        Take a replication controller, service, deployment or pod and expose it as a new Kubernetes Service
  run           Run a particular image on the cluster
  run-container  Run a particular image on the cluster
  set           Set specific features on objects

Basic Commands (Intermediate):
  get           Display one or many resources
  explain       Documentation of resources
  edit          Edit a resource on the server
  delete        Delete resources by filenames, stdin, resources and names, or by resources and label selector
```

```
Deploy Commands:
  rollout        Manage the rollout of a resource
  rolling-update Perform a rolling update of the given ReplicationController
  rollingupdate  Perform a rolling update of the given ReplicationController
  scale          Set a new size for a Deployment, ReplicaSet, Replication Controller, or Job
  resize         Set a new size for a Deployment, ReplicaSet, Replication Controller, or Job
  autoscale      Auto-scale a Deployment, ReplicaSet, or ReplicationController

<SNIP>
```

- Login to Bosh 's credhub instance:

```
root@bosh-client:~/ kubectl-cli # cd ..
```

```
root@bosh-client:~# ls

 bosh-deployment  credhub-cli   kubectl-cli   kubo-deployment
```

```
root@bosh-client:~# cd bosh-deployment/
```

```
root@bosh-client:~/bosh-deployment# credhub_user_password=$(bosh -e kubobosh int "../bosh-deployment/mycreds.yml" --path "/credhub_cli_password")
```

```
root@bosh-client:~/bosh-deployment# credhub_api_url="https://10.40.206.147:8844"
```

```
root@bosh-client:~/bosh-deployment# bosh -e kubobosh int "../bosh-deployment/mycreds.yml" --path="/uaa_ssl/ca" > credhubca.crt
```

```
root@bosh-client:~/bosh-deployment# bosh -e kubobosh int "../bosh-deployment/mycreds.yml" --path="/credhub_tls/ca" > credhub.crt
```

```
root@bosh-client:~/bosh-deployment# credhub login -u credhub-cli -p "${credhub_user_password}" -s "${credhub_api_url}" --ca-cert credhubca.crt --ca-cert credhub.crt

Setting the target url: https://10.40.206.147:8844
Login Successful
```

- Get K8s deployment certificate:

```
root@bosh-client:~/bosh-deployment# bosh int <(credhub get -n "/kubobosh/mykubocluster-1/tls-kubernetes" --output-json) --path=/value/ca > mykubecert.crt
```

```
root@bosh-client:~/bosh-deployment# endpoint="10.40.207.42"
root@bosh-client:~/bosh-deployment# port="443"
root@bosh-client:~/bosh-deployment# address="https://${endpoint}:${port}"
```

```
root@bosh-client:~/bosh-deployment# kubectl config set-cluster "mykubocluster-1" --server="$address" --certificate-authority=mykubecert.crt --embed-certs=true

Cluster "mykubocluster-1" set.
```

```
root@bosh-client:~/bosh-deployment# admin_password="VMware1!"
root@bosh-client:~/bosh-deployment# context_name="mykubocluster-1"
```

```
root@bosh-client:~/bosh-deployment# kubectl config set-credentials "mykubocluster-admin" --token="${admin_password}"

User "mykubocluster-admin" set.
```

```
root@bosh-client:~/bosh-deployment# kubectl config set-context "mykubocluster-1" --cluster="mykubocluster-1" --user="mykubocluster-admin"

Context "mykubocluster-1" created.
```

```
root@bosh-client:~/bosh-deployment# kubectl config use-context "mykubocluster-1"

Switched to context "mykubocluster-1".
```

Check:

```
root@bosh-client:~/bosh-deployment# kubectl get pods --namespace=kube-system
NAME                          READY    STATUS    RESTARTS   AGE
heapster-3662623559-h34fx          1/1      Running   0         38m
kube-dns-571194822-vfgv0           3/3      Running   0         38m
kubernetes-dashboard-3374579233-sgnws   1/1      Running   0         38m
monitoring-influxdb-1308349597-rzr5x    1/1      Running   0         38m
```

3.5 Deploy a Second Kubernetes Cluster

So far, we have already deployed 1 K8s cluster instance. Let's see how to deploy a second K8s cluster instance using the same Bosh Director.

```
root@bosh-client:~/ # cd kubo-deployment/
```

- Create 'create-cloud-config-2.sh' script file:

create-cloud-config-2.sh:	Description:
**/usr/local/bin/bosh int configurations/vsphere/cloud-config.yml **	
-o manifests/ops-files/ k8s-haproxy-static-ips-vsphere.yml	=> Name of the output file
**-v director_name=kubobosh **	
**-v internal_cidr=10.40.207.0/24 **	
**-v internal_gw=10.40.207.253 **	=> Name of Bosh Director
**-v internal_ip=10.20.20.1 **	=> Network CIDR for K8s Cluster
**-v kubernetes_master_host=10.40.207.80 **	=> Default GW
**-v worker_haproxy_ip_addresses=10.40.207.81 **	=> DNS Server IP address
**-v reserved_ips=[10.40.207.1-10.40.207.79,10.40.207.94-10.40.207.254] **	=> IP address of master node1 (=VIP of HAproxy)
**-v network_name=PG-COMP-VM-2 **	=> Reserved IP (will not be used)
**-v deployments_network=PG-COMP-VM-2 **	
**-v vcenter_cluster=Cluster-COMPUTE **	
-v vcenter_rp="K8S-2" > mycloudconfig-2.yml	=> Network Port-Group
	=> Network Port-Group

	=> ESXi Cluster where K8s Cluster will be hosted => Resource-Pool for K8s Cluster

Run the script file:

```
root@bosh-client:~/kubo-deployment# chmod +x ./create-cloud-config-2.sh
root@bosh-client:~/kubo-deployment# ./create-cloud-config-2.sh
```

```
root@bosh-client:~/kubo-deployment# ls

bin bosh-deployment configurations CONTRIBUTING.md create-cloud-
config.sh docs LICENSE manifests mycloudconfig-1.yml mycloudconfig-2.yml NOTICE README.md src
```

- Update cloud config:

```
root@bosh-client: ~/kubo-deployment# bosh -e kubobosh update-cloud-config mycloudconfig-2.yml

Using environment '10.40.206.147' as client 'admin'
<SNIP>

Continue? [yN]: y

Succeeded
```

- Create 'create-kubo-deployment-2.sh' script file:

create-kubo-deployment-2.sh:	Description:
/usr/local/bin/bosh int manifests/kubo.yml ** **-o manifests/ops-files/master-haproxy-vsphere.yml ** **-o manifests/ops-files/worker-haproxy.yml ** **-v deployments_network=PG-COMP-VM-2 ** **-v kubo-admin-password="VMware1!" ** **-v kubelet-password="VMware1!" ** **-v kubernetes_master_port=443 ** **-v kubernetes_master_host=10.40.207.80 ** **-v deployment_name=mykubocluster-2 ** **-v worker_haproxy_tcp_frontend_port=1234 ** **-v worker_haproxy_tcp_backend_port=4231 > mykubo-2.yml	=> Network Port-Group => Password for kubectl admin password => Password for kubelet => K8s api-server listening on HTTPS => IP of K8s master node (=VIP for HAproxy) => Name of the deployment

Run the script file:

```
root@bosh-client:~/kubo-deployment# chmod +x ./create-kubo-deployment-2.sh
root@bosh-client:~/kubo-deployment# ./create-kubo-deployment-2.sh
```

```
root@bosh-client:~/kubo-deployment# ls
bin  bosh-deployment  configurations  CONTRIBUTING.md  create-cloud-config.sh  create-kubo-deployment-
2.sh  docs  LICENSE  manifests  mycloudconfig-1.yml  mycloudconfig-2.yml  mykubo-1.yml mykubo-2.yml
NOTICE  README.md  src
```

- Deploy K8s Cluster:

```
root@bosh-client: ~/kubo-deployment#  /usr/local/bin/bosh -e kubobosh -d mykubocluster-2 deploy mykubo-2.yml

Using environment '10.40.206.147' as client 'admin'

Using deployment 'mykubocluster-2'

Release 'kubo-etcd/2' already exists.

Release 'docker/28.0.1' already exists.

Task 346

20:07:56 | Downloading remote release: Downloading remote release (00:00:01)
20:07:57 | Verifying remote release: Verifying remote release (00:00:00)
20:07:57 | Extracting release: Extracting release (00:00:00)
20:07:57 | Verifying manifest: Verifying manifest (00:00:00)
20:07:57 | Resolving package dependencies: Resolving package dependencies (00:00:00)
20:07:57 | Processing 2 existing jobs: Processing 2 existing jobs (00:00:00)
20:07:57 | Compiled Release has been created: haproxy/8.3.0 (00:00:00)

Started  Tue Aug 15 20:07:56 UTC 2017
Finished Tue Aug 15 20:07:57 UTC 2017
Duration 00:00:01

Task 346 done

<SNIP>
Continue? [yN]: y

Task 347
20:07:59 | Preparing deployment: Preparing deployment (00:00:02)
20:08:04 | Preparing package compilation: Finding packages to compile (00:00:00)
20:08:04 | Creating missing vms: etcd/d1b9d989-0e7d-4c99-98c2-38a371c1b8cf (0)
<SNIP>

Started  Tue Aug 15 20:07:56 UTC 2017
Finished Tue Aug 15 20:23:58 UTC 2017
Duration 00:16:02

Task 347 done
```

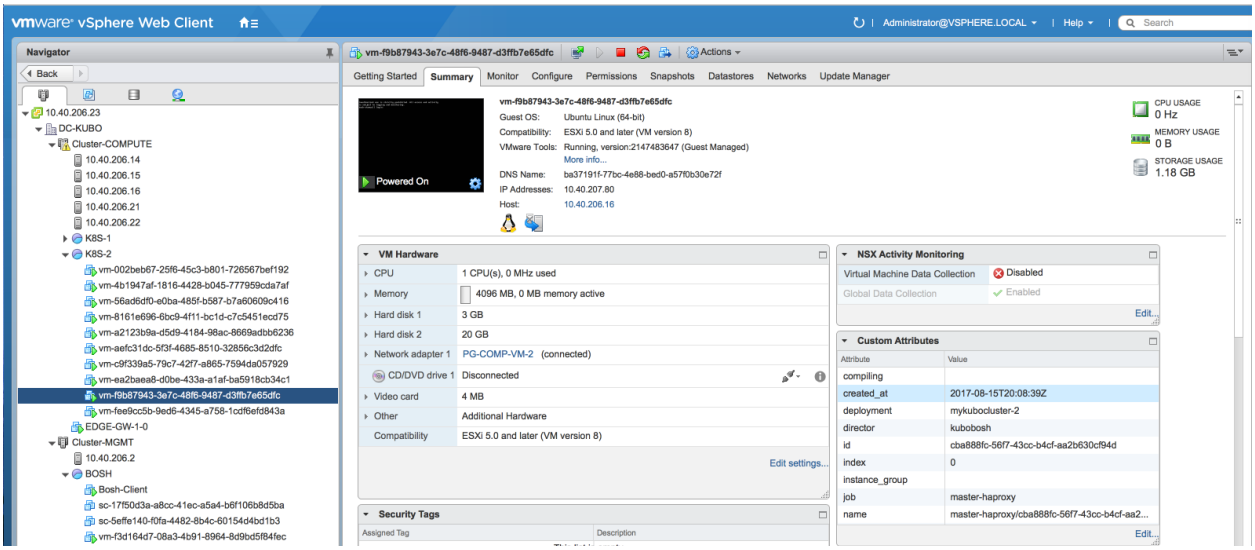
```
Succeeded
```

Check:

```
root@bosh-client:~/kubo-deployment# /usr/local/bin/bosh -e kubobosh deployments
Using environment '10.40.206.147' as client 'admin'

Name          Release(s)     Stemcell(s)                              Team(s)  Cloud Config
mykubocluster-1  kubo-etcd/2    bosh-vsphere-esxi-ubuntu-trusty-go_agent/3421.11  -       outdated
              kubo/0.7.0
              docker/28.0.1
              haproxy/8.4.0
mykubocluster-2  kubo-etcd/2    bosh-vsphere-esxi-ubuntu-trusty-go_agent/3421.11  -       latest
              kubo/0.7.0
              docker/28.0.1
              haproxy/8.4.0

2 deployments

Succeeded
```

```
root@bosh-client:~/kubo-deployment# /usr/local/bin/bosh -e kubobosh instances -d mykubocluster-2
Using environment '10.40.206.147' as client 'admin'

Task 350. Done

Deployment 'mykubocluster-2'

Instance                                     Process State  AZ  IPs
etcd/52f8fdc8-763f-4e22-b2bc-da39e26c3933          running       z1  10.40.207.82
etcd/9ed5ff80-c885-4207-8a40-e94ddf11b91a          running       z1  10.40.207.83
etcd/d1b9d989-0e7d-4c99-98c2-38a371c1b8cf          running       z1  10.40.207.90
master-haproxy/cba888fc-56f7-43cc-b4cf-aa2b630cf94d  running       z1  10.40.207.80
master/290ae301-a88d-49d3-8e5c-d056901c8e66        running       z1  10.40.207.84
master/2a573ec0-b417-475f-b24c-1649d90aaa67        running       z1  10.40.207.85
worker-haproxy/7f9ef734-323f-4aea-b5b5-aaf0bbf5cef6  running       z1  10.40.207.89
worker/84531868-aa8b-43af-880f-2806a6034ab1        running       z1  10.40.207.86
worker/870bba84-5cb0-48fd-ba2d-a99bf8639fe7        running       z1  10.40.207.87
worker/a20a650b-6ac6-42fe-9774-aa3a1282ad28        running       z1  10.40.207.88

10 instances

Succeeded
```

On vCenter, you should be able to see the new K8s cluster deployed in the COMPUTE Cluster, under resource-pool K8S-2:



- Login to Bosh ‘s credhub instance:

```
root@bosh-client:~/kubo-deployment # cd ..
```

```
root@bosh-client:~# ls
 bosh-deployment  credhub-cli   kubectl-cli   kubo-deployment
```

```
root@bosh-client:~# cd bosh-deployment/
```

```
root@bosh-client:~/bosh-deployment# credhub_user_password=$(bosh -e kubobosh int "../bosh-deployment/mycreds.yml" --path "/credhub_cli_password")
```

```
root@bosh-client:~/bosh-deployment# credhub_api_url="https://10.40.206.147:8844"
```

```
root@bosh-client:~/bosh-deployment# bosh -e kubobosh int "../bosh-deployment/mycreds.yml" --path="/uaa_ssl/ca" > credhubca.crt
```

```
root@bosh-client:~/bosh-deployment# bosh -e kubobosh int "../bosh-deployment/mycreds.yml" --
path="/credhub_tls/ca" > credhub.crt
```

```
root@bosh-client:~/bosh-deployment# credhub login -u credhub-cli -p "${credhub_user_password}" -s
"${credhub_api_url}" --ca-cert credhubca.crt --ca-cert credhub.crt

Setting the target url: https://10.40.206.147:8844
Login Successful
```

- Get K8s deployment certificate:

```
root@bosh-client:~/bosh-deployment# bosh int <(credhub get -n "/kubobosh/mykubocluster-2/tls-kubernetes" --
output-json) --path=/value/ca > mykubecert.crt
```

```
root@bosh-client:~/bosh-deployment# endpoint="10.40.207.80"

root@bosh-client:~/bosh-deployment# port="443"

root@bosh-client:~/bosh-deployment# address="https://${endpoint}:${port}"
```

```
root@bosh-client:~/bosh-deployment# kubectl config set-cluster "mykubocluster-2" --server="$address" --certificate-
authority=mykubecert.crt --embed-certs=true

Cluster "mykubocluster-2" set.
```

```
root@bosh-client:~/bosh-deployment# admin_password="VMware1!"

root@bosh-client:~/bosh-deployment# context_name="mykubocluster-2"
```

```
root@bosh-client:~/bosh-deployment# kubectl config set-credentials "mykubocluster-admin" --
token="${admin_password}"

User "mykubocluster-admin" set.
```

```
root@bosh-client:~/bosh-deployment# kubectl config set-context "mykubocluster-2" --cluster="mykubocluster-2" --user="mykubocluster-admin"

Context "mykubocluster-2" created.
```

```
root@bosh-client:~/bosh-deployment# kubectl config use-context "mykubocluster-2"

Switched to context "mykubocluster-2".
```

Check:

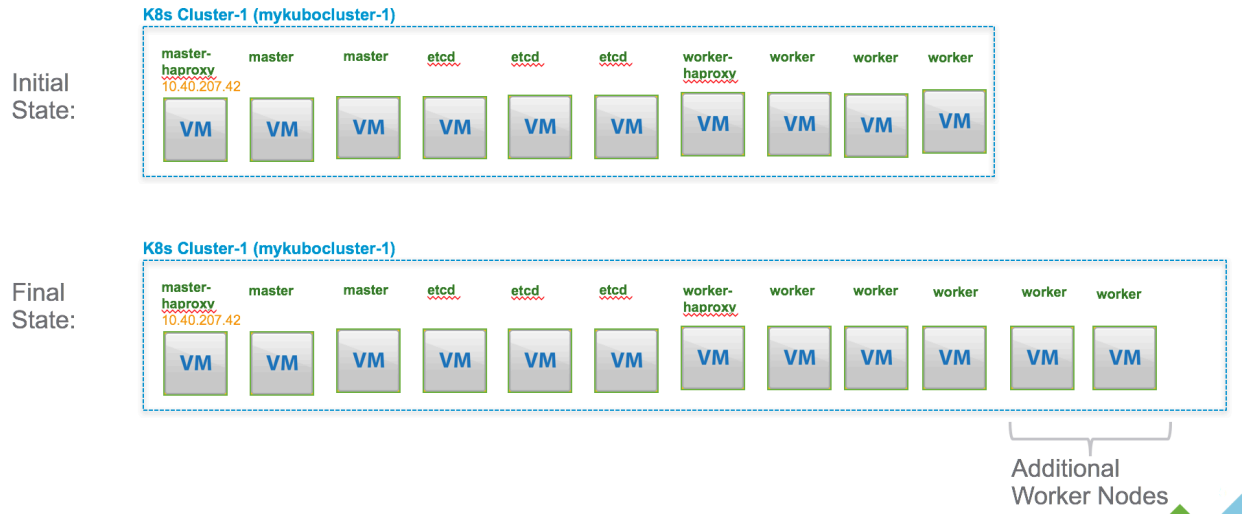
```
root@bosh-client:~/bosh-deployment# kubectl get pods --namespace=kube-system
NAME                          READY    STATUS    RESTARTS   AGE
heapster-3855037257-z3xgt           1/1      Running   0         4m
kube-dns-571194822-r5zrt            3/3      Running   0         4m
kubernetes-dashboard-3374579233-nw9dx   1/1      Running   0         4m
monitoring-influxdb-1308349597-vztqx    1/1      Running   0         4m
```

4. Scale in/out K8s Cluster

4.1 Scale out K8s Cluster

This section contains procedure to increase number of worker nodes in an existing K8s cluster deployment.

For mykubocluster-1, we want to scale the number of worker nodes from 4 to 6 as shown below:



step1: edit 'mykubo-1.yml' and modify number of instances for worker nodes (modify from 3 to 5):

```
<SNIP>
- azs:
  - z1
  instances: 5
  jobs:
  - name: flanneld
    release: kubo
  - name: docker
    properties:
      docker:
        default_ulimits:
        - nofile=65536
        flannel: true
        ip_masq: false
        iptables: false
        log_level: error
        storage_driver: overlay
        insecure_registries: ["10.40.207.9"]
      env: {}
    release: docker
  - name: kubeconfig
    properties:
      kubelet-password: VMware1!
      kubernetes-api-url: https://10.40.207.40:443
```

```
    tls:
      kubernetes: ((tls-kubernetes))
  release: kubo
 - name: cloud-provider
  properties: {}
  release: kubo
 - name: kubelet
  properties:
    backend_port: 4231
    kubernetes-api-url: https://10.40.207.40:443
    port: 1234
    tls:
      kubelet: ((tls-kubelet))
  release: kubo
 - name: kubernetes-proxy
  properties:
    kubernetes-api-url: https://10.40.207.40:443
  release: kubo
 name: worker
 networks:
 - name: DPortGroup-VM-1
 persistent_disk_type: 10240
 stemcell: trusty
 vm_type: worker
<SNIP>
```

step2: update K8s Cluster

```
root@bosh-client:~/kubo-deployment# / usr/local/bin/bosh -e kubobosh -d mykubocluster-1 deploy mykubo-1.yml

Using environment '10.40.206.147' as client 'admin'

Using deployment 'mykubocluster-1'

Release 'kubo-etcd/2' already exists.

Release 'docker/28.0.1' already exists.

Release 'haproxy/8.4.0' already exists.

 instance_groups:
 - name: worker
-  instances: 3
+  instances: 5

Continue? [yN]: y

Task 270

15:54:50 | Preparing deployment: Preparing deployment (00:00:01)
15:54:53 | Preparing package compilation: Finding packages to compile (00:00:00)
15:54:53 | Creating missing vms: worker/326e9ded-46c5-4291-be31-15485ca373ea (3)
15:54:53 | Creating missing vms: worker/7aa09699-6a8d-41a5-9872-3b2c02897cdf (4) (00:00:57)
<SNIP>
Task 270 done
```

```
Succeeded
```

We have successfully added 2 worker nodes in the K8s cluster deployment!

Check:

```
root@bosh-client:~/kubo-deployment# bosh -e kubobosh instances
Using environment '10.40.206.147' as client 'admin'

Task 690. Done

Deployment 'mykubocluster-1'

Instance                                     Process State  AZ  IPs
etcd/82615904-7d6c-4d67-a67a-411157660d63    running        z1  10.40.207.69
etcd/8a3708c7-e247-46cd-a13f-8da027cf9436    running        z1  10.40.207.70
etcd/fd53a53a-fe22-4d38-b021-ac8b37c8abcc    running        z1  10.40.207.71
master-haproxy/8fdf7c98-36d4-4da2-bb4e-68bec7484d82  running  z1  10.40.207.42
master/85260ce9-4c31-4010-aef6-e07d64097ab6  running        z1  10.40.207.73
master/e69ae876-4ede-4a8f-8ba9-1cfcf9a7675d  running        z1  10.40.207.72
worker-haproxy/039af73b-8cb8-4c2e-8fc1-590661805d94  running  z1  10.40.207.77
worker/11995ec7-2712-40f8-86ce-5821b4d5c399  running        z1  10.40.207.75
worker/326e9ded-46c5-4291-be31-15485ca373ea  running        z1  10.40.207.78
worker/6c78486e-0faf-4c0d-b7c5-7ba2f16ca781  running        z1  10.40.207.76
worker/7aa09699-6a8d-41a5-9872-3b2c02897cdf  running        z1  10.40.207.79
worker/88b7d382-7d3d-4ca7-b846-6b1616bc189a  running        z1  10.40.207.74

12 instances

Succeeded
```

```
root@bosh-client:~/kubo-deployment# kubectl get nodes -o wide
NAME          STATUS   AGE   VERSION   EXTERNAL-IP   OS-IMAGE           KERNEL-VERSION
10.40.207.74  Ready    5d    v1.6.6    <none>        Ubuntu 14.04.5 LTS  4.4.0-83-generic
10.40.207.75  Ready    5d    v1.6.6    <none>        Ubuntu 14.04.5 LTS  4.4.0-83-generic
10.40.207.76  Ready    5d    v1.6.6    <none>        Ubuntu 14.04.5 LTS  4.4.0-83-generic
10.40.207.78  Ready    9m    v1.6.6    <none>        Ubuntu 14.04.5 LTS  4.4.0-83-generic
10.40.207.79  Ready    7m    v1.6.6    <none>        Ubuntu 14.04.5 LTS  4.4.0-83-generic
```

4.2 Scale in K8s Cluster

In case you need to reduce the number of worker node (“scale in” in this case), perform the same operation as above by modifying the instances field in ‘mykubo-1.yml’ file and then by updating the K8s

cluster (same command than previously, i.e **/ usr/local/bin/bosh -e kubobosh -d mykubocluster-1 deploy mykubo-1.yml**).

5. Harbor Integration

5.1 Installing Harbor

In this lab, we are going to install Harbor in a VM (Ubuntu 16.04). Harbor will be running as a set of Docker containers so the following pre-requisites must be met on the VM:

- Docker engine installed
- Docker-compose utility available

Harbor will be using a self-signed certificate and will be configured with secure access mode (HTTPS).

5.1.1 Install Pre-Reqs

Install Docker Engine:

```
root@harbor:~/ # curl -sSL https://get.docker.com/ | sh
```

Install docker-compose:

```
root@harbor:~/ # apt-get install docker-compose
```

5.1.2 Create Certificates

Create certificates - CA certificate:

```
root@harbor:~/DATA  # mkdir CERTIFICATES
root@harbor:~/DATA # cd CERTIFICATES
```

```
root@harbor:~/DATA/CERTIFICATES  # openssl req \
>    -newkey rsa:4096 -nodes -sha256 -keyout ca.key \
>    -x509 -days 365 -out ca.crt
Generating a 4096 bit RSA private key
.....................................................................................................................................++
.....................++
writing new private key to 'ca.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:CA
Locality Name (eg, city) []:PALO ALTO
Organization Name (eg, company) [Internet Widgits Pty Ltd]:VMware
```

```
Organizational Unit Name (eg, section) []:CNABU
Common Name (e.g. server FQDN or YOUR name) []:Francis
Email Address []:guillierf@vmware.com
```

Check:

```
root@harbor:/DATA/CERTIFICATES # ls
ca.crt  ca.key
```

Files ca.crt and ca.key must have been created.

Create certificates – Certificate Signing Request (CSR):

```
root@harbor:~/DATA/CERTIFICATES  # openssl req \
>    -newkey rsa:4096 -nodes -sha256 -keyout harbor.com.key \
>    -out harbor.com.csr
Generating a 4096 bit RSA private key
.....................................................................++
.....................................................................................................................++
writing new private key to 'harbor.com.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:CA
Locality Name (eg, city) []:PALO ALTO
Organization Name (eg, company) [Internet Widgits Pty Ltd]:VMware
Organizational Unit Name (eg, section) []:CNABU
Common Name (e.g. server FQDN or YOUR name) []:Francis
Email Address []:guillierf@vmware.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

Check:

```
root@harbor:/DATA/CERTIFICATES # ls
ca.crt  ca.key  harbor.com.csr  harbor.com.key
```

Files harbor.com.csr and harbor.com.key must have been created.

Generate the certificate for Harbor host:

```
root@harbor:/DATA/CERTIFICATES # echo subjectAltName = IP:10.40.206.145 > extfile.cnf
```

```
root@harbor:/DATA/CERTIFICATES# openssl x509 -req -days 365 -in harbor.com.csr -CA ca.crt -CAkey ca.key -
CAcreateserial -CAcreateserial -extfile extfile.cnf -out harbor.com.crt
Signature ok
subject=/C=US/ST=CA/L=PALO ALTO/O=VMware/OU=CNABU/CN=Francis/emailAddress=guillierf@vmware.com
Getting CA Private Key
```

Check:

```
root@harbor:/DATA/CERTIFICATES# ls
ca.crt  ca.key  ca.srl  extfile.cnf  harbor.com.crt  harbor.com.csr  harbor.com.key
```

File harbor.com.crt must have been created.

Copy certificates to specific directory:

```
root@harbor:/DATA/CERTIFICATES# mkdir -p /root/cert/

root@harbor:/DATA/CERTIFICATES# cp harbor.com.crt /root/cert

root@harbor:/DATA/CERTIFICATES# cp harbor.com.key /root/cert
```

5.1.3 Install and Configure Harbor

Download Harbor offline installation binary:

```
root@harbor:~/ # mkdir /DATA
root@harbor:~/ # cd  /DATA
root@harbor:~/DATA # wget https://github.com/vmware/harbor/releases/download/v1.1.2/harbor-offline-installer-
v1.1.2.tgz
root@harbor:~/DATA # tar xvf harbor-offline-installer-v1.1.2.tgz
```

Edit harbor.cfg:

```
root@harbor:~/DATA # cd /DATA/harbor

root@harbor:~/DATA/harbor #  vi harbor.cfg
<SNIP>
hostname = 10.40.206.145
```

```
ui_url_protocol = https
ssl_cert = /root/cert/harbor.com.crt
ssl_cert_key = /root/cert/harbor.com.key
harbor_admin_password = VMware1!
<SNIP>
```

Note: for simplicity, we are not using DNS here. This is the reason why the field hostname is populated with Harbor IP address (10.40.206.145).

Prepare Harbor install:

```
root@harbor:~/DATA/harbor #  ./prepare
Generated and saved secret to file: /data/secretkey
Generated configuration file: ./common/config/nginx/nginx.conf
Generated configuration file: ./common/config/adminserver/env
Generated configuration file: ./common/config/ui/env
Generated configuration file: ./common/config/registry/config.yml
Generated configuration file: ./common/config/db/env
Generated configuration file: ./common/config/jobservice/env
Generated configuration file: ./common/config/jobservice/app.conf
Generated configuration file: ./common/config/ui/app.conf
Generated certificate, key file: ./common/config/ui/private_key.pem, cert file: ./common/config/registry/root.crt
The configuration files are ready, please use docker-compose to start the service.
```

Start harbor:

```
root@harbor:~/DATA/harbor #  docker-compose up -d

<SNIP>
Digest: sha256:07cd4b73ec64e12581399c4ab7c523553955946a02bba2be715c4f02b97bdf86
Status: Downloaded newer image for vmware/nginx:1.11.5-patched
Creating harbor-log
Creating harbor-adminserver
Creating registry
Creating harbor-db
Creating harbor-ui
Creating nginx
Creating harbor-jobservice
```

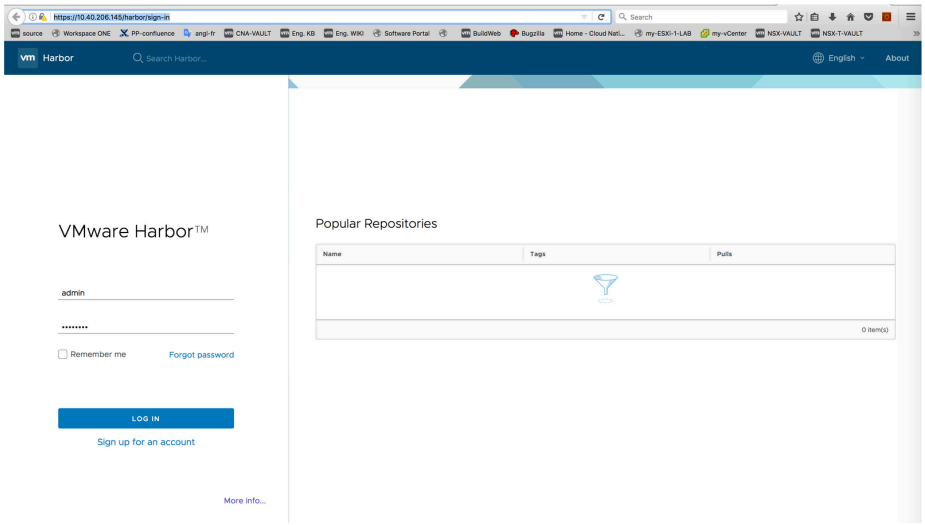
Check:

```
root@harbor:~/DATA/harbor #  docker-compose ps

     Name                   Command               State                         Ports
-------------------------------------------------------------------------------------------------------------------------------
harbor-adminserver   /harbor/harbor_adminserver      Up
harbor-db            docker-entrypoint.sh mysqld     Up      3306/tcp
harbor-jobservice    /harbor/harbor_jobservice       Up
harbor-log           /bin/sh -c crond && rm -f  ...   Up      127.0.0.1:1514->514/tcp
harbor-ui            /harbor/harbor_ui               Up
nginx                nginx -g daemon off;            Up      0.0.0.0:443->443/tcp, 0.0.0.0:4443->4443/tcp, 0.0.0.0:80->80/tcp
```

```
registry          /entrypoint.sh serve /etc/ ...   Up      5000/tcp
```

Open a web browser and use the following URL:
https://<Harbor IP Address>

You should be able to see this page:



5.1.4 Integrating Harbor with K8s Clusters

Now that Harbor is successfully installed and configured, we need to parameter K8s clusters with Harbor information like IP address and certificates.

Create a Kubernetes SECRET (command can be initiated from Bosh Client VM):

```
root@bosh-client:~/ # kubectl create secret docker-registry regsecret --docker-server=10.40.206.145 --docker-username=admin --docker-password='VMware1!' --docker-email='guillierf@vmware.com'

secret "regsecret" created
```

Check:

```
root@bosh-client:~/ # kubectl get secret

NAME                  TYPE                                  DATA      AGE
default-token-60f47   kubernetes.io/service-account-token   3         2h
regsecret             kubernetes.io/dockercfg               1         48m
```

Create client certificate for each K8s worker node:

SSH into worker node:

```
root@bosh-client:~/kubo-deployment/TEST-KUBECTL# bosh -e kubobosh -d mykubocluster-2 ssh worker/84531868-aa8b-43af-880f-2806a6034ab1
Using environment '10.40.206.147' as client 'admin'
<SNIP>
```

Create the directory /etc/docker/certs.d/<Harbor IP>/

```
worker/84531868-aa8b-43af-880f-2806a6034ab1:~$ sudo su

worker/84531868-aa8b-43af-880f-2806a6034ab1:/var/vcap/bosh_ssh/bosh_22f3c962f87d458#cd /etc

worker/84531868-aa8b-43af-880f-2806a6034ab1:/etc#cd docker/

worker/84531868-aa8b-43af-880f-2806a6034ab1:/etc/docker#ls
key.json

worker/84531868-aa8b-43af-880f-2806a6034ab1:/etc/docker#mkdir certs.d/

worker/84531868-aa8b-43af-880f-2806a6034ab1:/etc/docker#cd certs.d/

worker/84531868-aa8b-43af-880f-2806a6034ab1:/etc/docker/certs.d#mkdir 10.40.206.145

worker/84531868-aa8b-43af-880f-2806a6034ab1:/etc/docker/certs.d#cd 10.40.206.145/
```

Create the client certificate:

```
worker-haproxy/5694d3c7-1d16-4426-b085-b3bd4a8c23e5:/etc/docker/certs.d/10.40.206.145# openssl genrsa -out client.key 4096
Generating RSA private key, 4096 bit long modulus
..............++
.....................................................................................++
e is 65537 (0x10001)
worker-haproxy/5694d3c7-1d16-4426-b085-b3bd4a8c23e5:/etc/docker/certs.d/10.40.206.145# openssl req -new -x509 -text -key client.key -out client.cert
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:CA
Locality Name (eg, city) []:palo alto
Organization Name (eg, company) [Internet Widgits Pty Ltd]:VMware
Organizational Unit Name (eg, section) []:CNABU
Common Name (e.g. server FQDN or YOUR name) []:francis
Email Address []:guillierf@vmware.com
```

Check:

```
worker-haproxy/5694d3c7-1d16-4426-b085-b3bd4a8c23e5:/etc/docker/certs.d/10.40.206.145# ls

client.cert  client.key
```

Files client.cert and client.key should be created.

Import CA certificate from Harbor:

```
worker/84531868-aa8b-43af-880f-2806a6034ab1:/etc/docker/certs.d/10.40.206.145#scp
10.40.206.145:/DATA/CERTIFICATES/ca.crt .
The authenticity of host '10.40.206.145 (10.40.206.145)' can't be established.
ECDSA key fingerprint is 6c:ee:cf:bb:4e:5c:85:de:18:76:1b:63:55:d1:7d:9b.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.40.206.145' (ECDSA) to the list of known hosts.
root@10.40.206.145's password:
ca.crt                                        100% 2098     2.1KB/s   00:00
```

Everything is done. Worker node should be able to access Harbor registry using secure access mode (HTTPS).

5.1.5 Check that worker node is able to access Harbor registry

Bosh ssh into a worker node and issue the following commands:

```
worker/11995ec7-2712-40f8-86ce-5821b4d5c399:~$ sudo su

worker/11995ec7-2712-40f8-86ce-5821b4d5c399:/var/vcap/bosh_ssh/bosh_d34ef9a71fe0464# cd
/var/vcap/data/packages/docker/46a3b895e988e3879acec76f36c4728882503914/bin

worker/11995ec7-2712-40f8-86ce-
5821b4d5c399:/var/vcap/data/packages/docker/46a3b895e988e3879acec76f36c4728882503914/bin# ./docker -H
unix:///var/vcap/sys/run/docker/docker.sock login 10.40.206.145

Username (admin): admin
Password:
Login Succeeded
```

Note: the identifiant **46a3b895e988e3879acec76f36c4728882503914** will vary based on deployment.

Once login to the registry is successful, try to pull an image from Harbor to validate everything is OK:

```
worker/11995ec7-2712-40f8-86ce-5821b4d5c399:/var/vcap/data/packages/docker/46a3b895e988e3879acec76f36c4728882503914/bin# ./docker -H unix:///var/vcap/sys/run/docker/docker.sock pull 10.40.206.145/library/redis


Using default tag: latest
latest: Pulling from library/redis
552d35eef8d3: Pull complete
1456a69aa136: Extracting [==================================================>] 2.059 kB/2.059 kB
1456a69aa136: Pull complete
1f49a48e1dbe: Extracting [=>                                 ] 32.77 kB/981.7 kB
1f49a48e1dbe: Pull complete
36d15e4aba9d: Extracting [==================>                ] 2.753 MB/8.063 MB
36d15e4aba9d: Extracting [============================>      ] 4.522 MB/8.063 MB
36d15e4aba9d: Pull complete
5091924c2fba: Pull complete
1f196e73b55c: Pull complete
Digest: sha256:1bce822ce3c234b07c4036ead05c464e7972a565b63779c37d0efd25b69d188a
Status: Downloaded newer image for 10.40.206.145/library/redis:latest
```

```
worker/11995ec7-2712-40f8-86ce-5821b4d5c399:/var/vcap/data/packages/docker/46a3b895e988e3879acec76f36c4728882503914/bin# ./docker -H unix:///var/vcap/sys/run/docker/docker.sock images | grep redis

10.40.206.145/library/redis                latest              d4f259423416        4 weeks ago         105.9 MB
```

5.1.5.1. Sample manifest file

To use the private registry, specify Harbor IP address in the image definition of the manifest file.

nginx.yml

```
apiVersion: v1
kind: Pod
metadata:
  name: private-reg
spec:
  containers:
    - name: private-reg-container
      image: 10.40.206.145/library/nginx
  imagePullSecrets:
    - name: regsecret
```

To create the pod from this yml file: **kubectl create -f nginx.yml**

6. Bosh Useful Commands

- Check KuBo releases:

```
root@bosh-client:~# /usr/local/bin/bosh -e kubobosh releases
Using environment '10.40.206.147' as client 'admin'

Name      Version  Commit Hash
docker    28.0.1*  8096ad43+
haproxy   8.4.0*   544916ce+
kubo      0.7.0*   1017224
kubo-etcd 2*       aa57fc9

(*) Currently deployed
(+) Uncommitted changes

4 releases

Succeeded
```

- Check stemcells release:

```
root@bosh-client:~# /usr/local/bin/bosh -e kubobosh stemcells
Using environment '10.40.206.147' as client 'admin'

Name                                          Version   OS            CPI  CID
bosh-vsphere-esxi-ubuntu-trusty-go_agent  3421.11*  ubuntu-trusty -    sc-5effe140-f0fa-4482-8b4c-60154d4bd1b3

(*) Currently deployed

1 stemcells

Succeeded
```

- Check current K8s deployments:

```
root@bosh-client:~# /usr/local/bin/bosh -e kubobosh deployments
Using environment '10.40.206.147' as client 'admin'

Name          Release(s)     Stemcell(s)                                  Team(s)  Cloud Config
mykubocluster-1  kubo-etcd/2    bosh-vsphere-esxi-ubuntu-trusty-go_agent/3421.11  -       latest
              kubo/0.7.0
              docker/28.0.1
              haproxy/8.4.0

1 deployments

Succeeded
```

- Check current VM deployments:

```
root@bosh-client:~# /usr/local/bin/bosh -e kubobosh instances
Using environment '10.40.206.147' as client 'admin'

Task 334. Done

Deployment 'mykubocluster-1'

Instance                                      Process State  AZ  IPs
etcd/2e97d38a-76f1-448d-8fb3-d2423623fef5          running       z1  10.40.207.70
etcd/68e20a9f-8cad-42ca-8ffa-5e1ebe6552d5          running       z1  10.40.207.71
etcd/a1e0f121-ec1e-47fd-926b-8699fd87fdf5          running       z1  10.40.207.69
master-haproxy/7b51590a-73aa-43ba-b9e0-14e54a22cee8  running       z1  10.40.207.42
master/5ea5bcf9-e647-45ad-9e63-2286f517b14f        running       z1  10.40.207.73
master/acf2a254-cc96-407a-a708-6c9bdf1fd608        running       z1  10.40.207.72
worker-haproxy/573f4cde-ec43-424b-b92f-b13dd3b25dbb  running       z1  10.40.207.77
worker/1c0ceb4b-9cf2-4772-8db9-89a9e64e7a66        running       z1  10.40.207.74
worker/67a4670a-667e-4bfd-9f94-8476f1952296        running       z1  10.40.207.75
worker/9556df08-cd39-4c0b-80cd-e9e6e44a93cf        running       z1  10.40.207.76

10 instances
```

```
Succeeded
```

- Check specific task output:

```
root@bosh-client:~# /usr/local/bin/bosh -e kubobosh task 328
Using environment '10.40.206.147' as client 'admin'

Task 328

17:34:22 | Preparing deployment: Preparing deployment (00:00:03)
17:34:28 | Preparing package compilation: Finding packages to compile (00:00:00)
17:34:28 | Creating missing vms: etcd/a1e0f121-ec1e-47fd-926b-8699fd87fdf5 (0)
17:34:28 | Creating missing vms: etcd/2e97d38a-76f1-448d-8fb3-d2423623fef5 (1)
17:34:28 | Creating missing vms: etcd/68e20a9f-8cad-42ca-8ffa-5e1ebe6552d5 (2)
17:34:28 | Creating missing vms: master/5ea5bcf9-e647-45ad-9e63-2286f517b14f (1)
17:34:28 | Creating missing vms: worker/67a4670a-667e-4bfd-9f94-8476f1952296 (2)
17:34:28 | Creating missing vms: master-haproxy/7b51590a-73aa-43ba-b9e0-14e54a22cee8 (0)
17:34:28 | Creating missing vms: worker/1c0ceb4b-9cf2-4772-8db9-89a9e64e7a66 (0)
17:34:28 | Creating missing vms: worker/9556df08-cd39-4c0b-80cd-e9e6e44a93cf (1)
17:34:28 | Creating missing vms: worker-haproxy/573f4cde-ec43-424b-b92f-b13dd3b25dbb (0)
17:34:28 | Creating missing vms: master/acf2a254-cc96-407a-a708-6c9bdf1fd608 (0)
17:37:40 | Creating missing vms: worker/1c0ceb4b-9cf2-4772-8db9-89a9e64e7a66 (0) (00:03:12)
17:37:41 | Creating missing vms: etcd/68e20a9f-8cad-42ca-8ffa-5e1ebe6552d5 (2) (00:03:13)
17:37:41 | Creating missing vms: worker/67a4670a-667e-4bfd-9f94-8476f1952296 (2) (00:03:13)
17:37:41 | Creating missing vms: worker/9556df08-cd39-4c0b-80cd-e9e6e44a93cf (1) (00:03:13)
17:37:41 | Creating missing vms: master/5ea5bcf9-e647-45ad-9e63-2286f517b14f (1) (00:03:13)
17:37:42 | Creating missing vms: master/acf2a254-cc96-407a-a708-6c9bdf1fd608 (0) (00:03:14)
17:37:42 | Creating missing vms: etcd/2e97d38a-76f1-448d-8fb3-d2423623fef5 (1) (00:03:14)
17:37:42 | Creating missing vms: etcd/a1e0f121-ec1e-47fd-926b-8699fd87fdf5 (0) (00:03:14)
17:37:52 | Creating missing vms: worker-haproxy/573f4cde-ec43-424b-b92f-b13dd3b25dbb (0) (00:03:24)
17:37:52 | Creating missing vms: master-haproxy/7b51590a-73aa-43ba-b9e0-14e54a22cee8 (0) (00:03:24)
17:37:52 | Updating instance etcd: etcd/a1e0f121-ec1e-47fd-926b-8699fd87fdf5 (0) (canary) (00:01:06)
17:38:58 | Updating instance etcd: etcd/2e97d38a-76f1-448d-8fb3-d2423623fef5 (1) (00:00:59)
17:39:57 | Updating instance etcd: etcd/68e20a9f-8cad-42ca-8ffa-5e1ebe6552d5 (2) (00:00:59)
17:40:56 | Updating instance master: master/acf2a254-cc96-407a-a708-6c9bdf1fd608 (0) (canary) (00:01:32)
17:42:28 | Updating instance master: master/5ea5bcf9-e647-45ad-9e63-2286f517b14f (1) (00:01:17)
17:43:45 | Updating instance master-haproxy: master-haproxy/7b51590a-73aa-43ba-b9e0-14e54a22cee8 (0) (canary)
(00:00:33)
```

```
17:44:18 | Updating instance worker: worker/1c0ceb4b-9cf2-4772-8db9-89a9e64e7a66 (0) (canary) (00:05:25)

17:49:43 | Updating instance worker: worker/67a4670a-667e-4bfd-9f94-8476f1952296 (2) (00:03:13)

17:52:56 | Updating instance worker: worker/9556df08-cd39-4c0b-80cd-e9e6e44a93cf (1) (00:02:05)

17:55:01 | Updating instance worker-haproxy: worker-haproxy/573f4cde-ec43-424b-b92f-b13dd3b25dbb (0) (canary)
(00:00:31)


Started  Tue Aug 15 17:34:22 UTC 2017

Finished Tue Aug 15 17:55:32 UTC 2017

Duration 00:21:10


Task 328 done


Succeeded
```

- Perform cloud-check operation on a K8s cluster deployment:

```
root@bosh-client:~# /usr/local/bin/bosh -e kubobosh cloud-check -d mykubocluster-1

Using environment '10.40.206.147' as client 'admin'


Using deployment 'mykubocluster-1'


Task 335


19:36:37 | Scanning 10 VMs: Checking VM states (00:00:06)

19:36:43 | Scanning 10 VMs: 10 OK, 0 unresponsive, 0 missing, 0 unbound (00:00:00)

19:36:43 | Scanning 6 persistent disks: Looking for inactive disks (00:00:09)

19:36:52 | Scanning 6 persistent disks: 6 OK, 0 missing, 0 inactive, 0 mount-info mismatch (00:00:00)


Started  Tue Aug 15 19:36:37 UTC 2017

Finished Tue Aug 15 19:36:52 UTC 2017

Duration 00:00:15


Task 335 done


# Type  Description


0 problems

Succeeded
```

- SSH to a specific VM instance:

```
root@bosh-client:~# /usr/local/bin/bosh -e kubobosh ssh master-haproxy/7b51590a-73aa-43ba-b9e0-14e54a22cee8 -d mykubocluster-1
Using environment '10.40.206.147' as client 'admin'

Using deployment 'mykubocluster-1'

Task 336. Done
Unauthorized use is strictly prohibited. All access and activity
is subject to logging and monitoring.
Welcome to Ubuntu 14.04.5 LTS (GNU/Linux 4.4.0-83-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

Last login: Tue Aug 15 19:37:39 2017 from 10.40.207.38
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

master-haproxy/7b51590a-73aa-43ba-b9e0-14e54a22cee8:~$
Succeeded
```

- Retrieve logs for a specific VM instance:

```
root@bosh-client:~# /usr/local/bin/bosh -e kubobosh logs master-haproxy/7b51590a-73aa-43ba-b9e0-14e54a22cee8 -d mykubocluster-1
Using environment '10.40.206.147' as client 'admin'

Using deployment 'mykubocluster-1'

Task 338
```

```
19:38:18 | Fetching logs for master-haproxy/7b51590a-73aa-43ba-b9e0-14e54a22cee8 (0): Finding and packing log files (00:00:01)

Started  Tue Aug 15 19:38:18 UTC 2017

Finished Tue Aug 15 19:38:19 UTC 2017

Duration 00:00:01

Task 338 done

Downloading resource '246884fb-cfba-4e5c-5a26-d7a635043f16' to '/root/mykubocluster-1.master-haproxy.7b51590a-73aa-43ba-b9e0-14e54a22cee8-20170815-123820-702151118.tgz'...

                                                    0.00%

Succeeded
```

- Delete a K8s cluster deployment :

```
root@bosh-client:~# /usr/local/bin/bosh -e kubobosh delete-deployment -d mykubocluster-1

Using environment '10.40.206.147' as client 'admin'

Using deployment 'mykubocluster-1'

Continue? [yN]: y
```

7. Conclusion

This guide has shown how to successfully deploy multiple Kubernetes clusters using a same and unique Bosh Director instance on vSphere. It also demonstrated how to scale out an existing Kubernetes deployment in order to increase number of worker nodes. Lastly, the guide covered integration with Harbor (VMware enterprise grade private registry) which allows the Kubernetes cluster to access images from a secure location using secure access mode (HTTPS).

This guide also includes useful Bosh commands to monitor and manage the Kubernetes clusters.