BOSH Tutorial

VMware 2012 - Cloud Foundry

April 10, 2012

1 Introduction

This tutorial will guide you through the process of deploying a multi-tier WordPress installation using BOSH. Due to its simplicity, WordPress is a good way to learn BOSH, but it is not meant to represent a realistic use case.

2 Prerequisites

A working BOSH director

3 Installing BOSH on an Ubuntu VM

3.1 Install Ruby via rbenv

1. BOSH is written in Ruby. Let's install Ruby's dependencies

   ```
   sudo apt-get install git-core build-essential libsqlite3-dev curl libmysqlclient-dev libxml2
   ```

2. Get the latest version of rbenv

   ```
   cd
   git clone git://github.com/sstephenson/rbenv.git .rbenv
   ```

3. Add ~/.rbenv/bin to your $PATH for access to the rbenv command-line utility

   ```
   echo 'export PATH="$HOME/.rbenv/bin:$PATH"' >> ~/.bash_profile
   ```

4. Add rbenv init to your shell to enable shims and autocompletion

   ```
   echo 'eval "$(rbenv init -)"' >> ~/.bash_profile
   ```

5. Download Ruby 1.9.2

   ```
   wget http://ftp.ruby-lang.org/pub/ruby/1.9/ruby-1.9.2-p290.tar.gz
   ```

6. Unpack and install Ruby

```
tar xvfz ruby-1.9.2-p290.tar.gz
cd ruby-1.9.2-p290
./configure --prefix=$HOME/.rbenv/versions/1.9.2-p290
make
make install
```

7. Restart your shell so the path changes take effect

```
source ~/.bash_profile
```

8. Set your default Ruby to be version 1.9.2

```
rbenv global 1.9.2-p290
```

3.2 Install Local BOSH and BOSH Releases

1. Sign up for an account on the Cloud Foundry public Gerrit server at http://reviews.cloudfoundry.org/ [1]

2. Generate an ssh public key (accept all defaults)

```
ssh-keygen -t rsa
```

3. Copy your key from ∼/.ssh/id_rsa.pub into your Gerrit account

4. Set your name and email

```
git config --global user.name "Firstname Lastname"
git config --global user.email "your_email@youremail.com"
```

5. Install our gerrit-cli gem:

```
gem install gerrit-cli
rbenv rehash
```

6. Clone BOSH repositories from Gerrit

```
gerrit clone ssh://reviews.cloudfoundry.org:29418/bosh-sample-release.git
gerrit clone ssh://reviews.cloudfoundry.org:29418/release.git
gerrit clone ssh://reviews.cloudfoundry.org:29418/bosh.git
```

7. Run some rake tasks to install the BOSH CLI

```
cd ~/bosh
rake bundle_install (Note: if this fails run 'gem pristine rake' and retry)
cd cli
bundle exec rake build
gem install pkg/bosh_cli-x.x.x.gem
```

[1] http://reviews.cloudfoundry.org/

3.3 Deploy to your BOSH Environment

With a fully configured environment, we can begin deploying the sample application to our environment. As listed in the prerequisites, you should already have an environment running, as well as the IP address of the BOSH Director. Ask your BOSH technical contact for help if you need it.

Point BOSH at a Target and Clean your Environment

1. Target your director (this IP is an example - replace with yours!)

   ```
   bosh target 192.168.1.99:25555
   ```

2. Check the state of your BOSH settings.

   ```
   bosh status
   ```

3. The result of your status will be akin to:

   ```
   Target         targetname (http://198.1621.99:25555) Ver: 0.3.12 (01169817)
   UUID           #####hex-uuid-goes-here-#############
   User           admin
   Deployment     not set
   ```

4. If you have previously used this BOSH director you may have existing deployments and releases. List previous deployments (we will remove them in a moment)

   ```
   bosh deployments
   ```

5. If you have any existing deployments `bosh deployments` will show something like:

   ```
   +---------+
   | Name    |
   +---------+
   | example |
   +---------+
   ```

6. Delete any existing deployments (ex: example)

   ```
   bosh delete deployment example
   ```

7. Answer `yes` to the prompt and wait for the deletion to complete

8. List previous releases (we will remove them in a moment)

   ‘bosh releases‘

9. If you have result of `bosh releases` should be akin to:

```
+--------+----------+
| Name   | Versions |
+--------+----------+
| myapp  | 1, 2, 3  |
+--------+----------+
```

10. Delete the existing releases (ex: myapps)

```
bosh delete release myapp
```

11. Answer `yes` to the prompt and wait for the deletion to complete

Create a Release

1. Change directories into bosh-sample-release:

```
cd ~/bosh-sample-release
```

   This directory contains the sample WordPress deployment and release files.

2. Reset your environment

```
bosh reset release
```

3. Answer `yes` to the prompt and wait for the environment to be reset
4. Create a release

```
bosh create release --force --with-tarball
```

5. Answer `wordpress` to the `release name` prompt
6. Your terminal will display information about the release including the Release Manifest, Packages, Jobs, and tarball location.
7. Open `bosh-sample-release/wordpress.yml` in your favorite text editor and confirm that `name` is `wordpress` and `version` matches the version that was displayed in your terminal (if this is your first release, this will be version 1).

Deploy the Release

1. Upload the WordPress example Release to your Environment

   ```
   bosh upload release dev_releases/wordpress-1.tgz
   ```

2. Your terminal will display information about the upload, and an upload progress bar will reach 100% after a few minutes.

3. Open `bosh-sample-release/wordpress.yml`. You will need to edit several things to match your environment. You should update director_uuid, any VLANs, the IPs under networks, dns servers and any static ips listed for specific jobs. An example manifest is in the Appendix.

4. Deploy the Release

   ```
   bosh deploy
   ```

5. Your deployment will take a few minutes.

6. Copy the URL for your WordPress installation from the `properties.wordpress.servername` value in `wordpress.yml`

7. Browse your WordPress blog at this URL.

8. Complete the form to install your WordPress blog

3.4 Appendix

```
---
name: wordpress
director_uuid: #####hex-uuid-goes-here-############

release:
    name: wordpress
    version: 1

compilation:
    workers: 4
    network: default
    cloud_properties:
        ram: 2048
        disk: 8096
        cpu: 2

# this section describes how updates are handled

update:
```

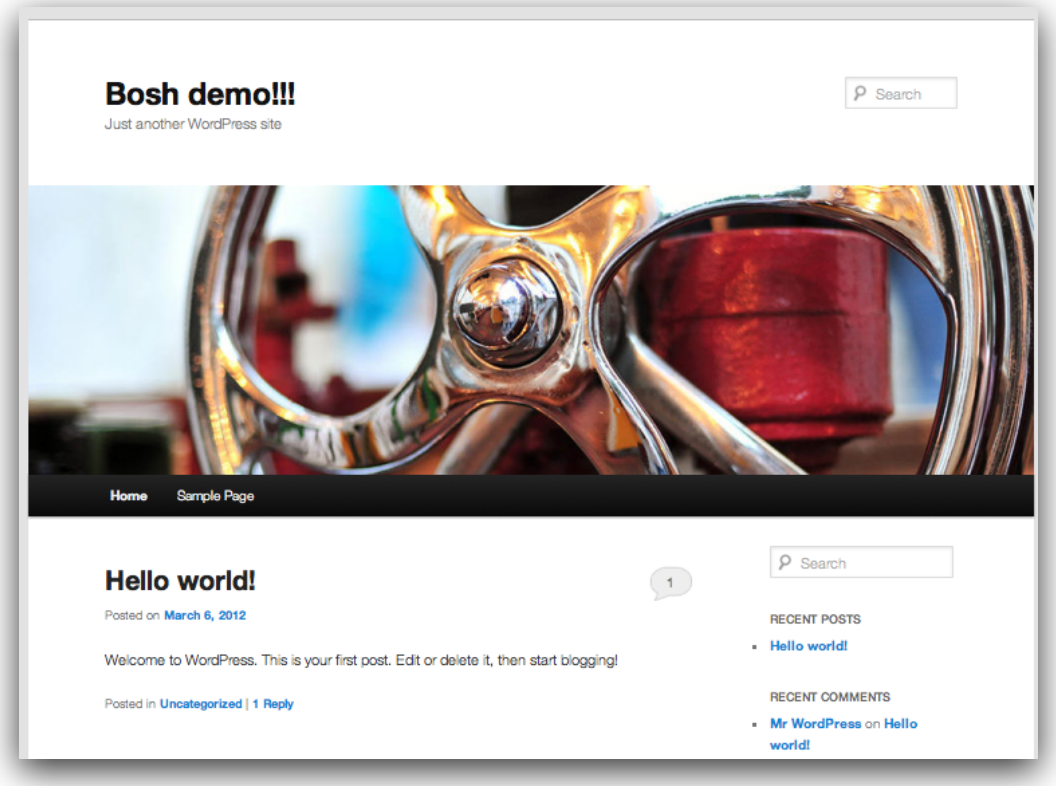

Figure 1: Deployed Wordpress

```
    canaries: 1
    canary_watch_time: 30000
    update_watch_time: 30000
    max_in_flight: 4
    max_errors: 1

networks:
- name: default
    subnets:
    - reserved:
        - 192.168.224.2 - 192.168.224.10
        - 192.168.224.200 - 192.168.224.254
    static:
        - 192.168.224.11 - 192.168.224.100
        range: 192.168.224.0/23
        gateway: 192.168.224.1
    dns:
        - 192.168.1.4
        - 192.168.1.5
    cloud_properties:
```

```
        name: VLAN1234

- name: dmz
  subnets:
  - static:
    - 192.168.4.241 - 192.168.4.242
  range: 192.168.4.241/28
  dns:
        - 192.168.1.4
        - 192.168.1.5
    cloud_properties:
  cloud_properties:
    name: VLAN4321

resource_pools:

    - name: infrastructure
    network: default
    size: 6
    stemcell:
        name: bosh-stemcell
        version: 0.4.7
    cloud_properties:
        cpu: 1
        disk: 8192
        ram: 4096

jobs:
    - name: mysql
    template: mysql
    instances: 1
    resource_pool: infrastructure
    persistent_disk: 16384
    networks:
        - name: default
        static_ips:
            - 192.168.224.11

    - name: wordpress
    template: wordpress
    instances: 4
    resource_pool: infrastructure
    networks:
        - name: default
        static_ips:
            - 192.168.224.12 - 192.168.224.15

    - name: nginx
    template: nginx
```

```
    instances: 1
    resource_pool: infrastructure
    networks:
        - name: default
            default: [dns, gateway]
            static_ips:
                - 192.168.224.1
        - name: dmz
            static_ips:
                - 192.168.4.241

properties:
    wordpress:
        admin: foo@bar.com
        port: 8008
        servers:
                - 192.168.224.12
                - 192.168.224.13
                - 192.168.224.14
                - 192.168.224.15
        servername: wp.myurl.mycompany.com
        db:
            name: wp
            user: wordpress
            pass: w0rdpr3ss
        auth_key: random key
        secure_auth_key: random key
        logged_in_key: random key
        nonce_key: random key
        auth_salt: random key
        secure_auth_salt: random key
        logged_in_salt: random key
        nonce_salt: random key
    mysql:
        address: 192.168.224.11
        port: 3306
        password: verysecretpasswordforroot
    nginx:
        workers: 1
```