(../..)

In this lab you will clone a release repository, compile that release, and upload it to your BOSH-lite director so that you can deploy it.

You will also gain experience with tools for troubleshooting a BOSH deployment.

Setup

In this lab you will use BOSH to deploy a simple Java app called `articulate`. This app Jar file that exposes a web application.

You will be deploying that Jar using the `articulate-release`.

First, check out the v1 tag (https://github.com/platform-acceleration-lab/apps-platform-acceleration-bosh-code/tree/v1) in the `platform-acceleration-bosh-code` repository.

```
cd platform-acceleration-bosh-code
git checkout v1
```

Next, `cd` into the `articulate-release` directory.

```
cd articulate-release
```

This is your release directory. Explore its contents.

A Note on YAML

If you are not familiar with YAML, take several minutes to go over this overview of YAML syntax (http://docs.ansible.com/ansible/YAMLSyntax.html).

It is required for both BOSH and Concourse.

Target your BOSH-lite Director

Make sure you are targeting your BOSH-lite director by running the following command:

```
bosh target 192.168.50.4
```

Create the Release

Tell BOSH to create the release from the current directory:

```
bosh create release
```

When you run the `bosh create release` command, BOSH fetches any required blobs and tars everything into a single, compressed file -- the release.

Normally, BOSH then uploads blobs to the blobstore configured in `final.yml` (usually the blobstore is some kind of cloud storage like s3). In this case, though, we've created a `dev` release, which means the release and blobs are stored locally.

Upload the Release

Upload the release to the BOSH director.

```
bosh upload release
```

Upload Required Stemcells

Look at your deployment manifest, `articulate.yml` (in the root of the `platform-acceleration-bosh-code` repository)

In the `resource_pools` block, note the required stemcell.

```
resource_pools:
  - name: articulate
    network: articulate
    cloud_properties: {}
    stemcell:
      name: bosh-warden-boshlite-ubuntu-trusty-go_agent
      version: latest
```

Upload this stemcell (http://bosh.io/stemcells/bosh-warden-boshlite-ubuntu-trusty-go_agent) to the director.

```
bosh upload stemcell https://s3.amazonaws.com/bosh-core-stemcells/ward
en/bosh-stemcell-3312.6-warden-boshlite-ubuntu-trusty-go_agent.tgz
```

Deploy

Change into the root of the `platform-acceleration-bosh-code` and point the BOSH cli to the deployment manifest.

```
bosh deployment articulate.yml
```

Deploy.

```
bosh deploy
```

What happens?

Troubleshooting the Deployment

The deployment should fail with an error:

```
Deploying
---------
Are you sure you want to deploy? (type 'yes' to continue):
Director task 53
Deprecation: Ignoring cloud config. Manifest contains 'networks' section.

  Started preparing deployment > Preparing deployment. Done (00:00:00)

  Started preparing package compilation > Finding packages to compile. Done (00:00:00)

  Started updating job articulate > articulate/0 (53e4bbb8-630d-4144-9009-10a059f8cd91) (canary).
Failed: 'articulate/0 (53e4bbb8-630d-4144-9009-10a059f8cd91)' is not running after update. Review
logs for failed jobs: articulate (00:01:38)

Error 400007: 'articulate/0 (53e4bbb8-630d-4144-9009-10a059f8cd91)' is not running after update. R
eview logs for failed jobs: articulate

Task 53 error

For a more detailed error report, run: bosh task 53 --debug
```

What went wrong?

Download the deployment logs from your BOSH director.

```
bosh logs articulate 0
```

Now extract, and examine the logs:

NOTE: replace `ARTICULATE_LOG_TARBALL` with the name of the tarball downloaded by the BOSH cli.

```
tar -xzvf $ARTICULATE_LOG_TARBALL
cat articulate/articulate.stderr.log
```

What is the error?

Modify the `articulate.yml` deployment manifest by adding a `port` property to the `articulate` job. Use port `9000`.

```
jobs:
 - release: articulate
   name: articulate
   properties: {port: 9000}
```

Redeploy with `bosh -n deploy`. (The `-n` flag tells BOSH to deploy without the interactive prompt.)

Verifying the Deployment

Verify the deployment was successful.

```
bosh vms
```

The articulate VM should be *running*.

`curl` the application to confirm that it is running. In order to use `curl`, configure a route in the local routing table to use the VirtualBox VM's IP as the gateway for the `10.244.0.0/16` subnet.

```
sudo route add –net 10.244.0.0/16 192.168.50.4
curl –H 'Accept: application/json' http://10.244.9.5:9000/info
```

NOTE: The `route add` command will only work on macs.

The response should look like the following:

```
{
  "build": {
    "artifact": "articulate",
    "name": "articulate",
    "description": "Articulate the value of Cloud Foundry",
    "version": "0.0.1–SNAPSHOT"
  }
}
```

Summary

In this lab you learned how to create a release, how to upload releases and stemcells to a BOSH director, how to deploy using BOSH and BOSH manifests, and how to download logs from the BOSH director to troubleshoot failing deployments.

Beyond the lab

The deployment in this lab initially failed because the articulate job (in the release) defined a default port value of `0`. It is better to *provide no value at all* than to provide a problematic default. In the first case, BOSH will prevent you from even starting a deployment.

You can see this in practice by doing the following:

1. Edit the `spec` file for the articulate job (`articulate-release/jobs/articulate/spec`) to provide no default value for the `port` property.
2. Recreate and re-upload the release.
3. Restore the `articulate.yml` manifest back to its original state with no port value provided
4. Run `bosh deploy` . What happens? Notice how BOSH prevents you from starting the deployment.

(https://pivotal.io)