Cloud Foundry and ODPi

Working Towards Enabling Cloud-Native, Data-Driven Applications

Roman Shaposhnik, Director of Open Source, Pivotal, @rhatr
Richard Pelavin, Reactor8 Inc.
Agenda

- ODPi & CFF – what’s the deal?
- Data App Case Study
- Prototype Architecture and Functionality
- Case Study Lessons Learned
What is ODPi?

A shared industry effort focused on promoting and advancing the state of Apache Hadoop® and Big Data Technologies for the Enterprise.
Platform governed in a Hybrid way
Platform governed in a Hybrid way
Nearly doubled membership since 2015
ODPi 1.0 release

1. Runtime specification
   https://github.com/odpi/specs

2. Reference implementation
   • RPM and DEB packages
   • Maven repo
     http://repo.odpi.org/ODPi/1.0/

3. Validation testsuite
   …/acceptance-tests
Cloud-native ODPi

BOSH
Case Study Objectives

• Provide “data developer” with self-service Cloud deployment
  – Prototype: Spark/HDFS cluster + Zeppelin on AWS
  – Remove/minimize manual steps from empty state to full deployment (i.e., getting close to ‘single click’ deployment)

• Use Apache Bigtop packaging and config logic

• Eye towards ODPi implementation after ODPi v1 and beyond

• Use BOSH for Cloud/deployment abstraction

• Implement quickly, leverage off-shelf tooling/practices to fill gaps
Leveraging Apache Bigtop

• Packaging and configuration for data service components
  – Standard packaging (e.g., debs, rpms)
  – Vetted Puppet modules for config tightly coupled to packaging

• “Turnkey” with ODPi, which is based on Bigtop
Leveraging BOSH

• Cloud portability
  – IaaS/Cloud-independent abstraction for images, node deployment, and connection to network and storage
  – Vetted providers exist for AWS, Azure, etc

• Robust node deployment

• Simple and intuitive command line user experience
BOSH Gaps for Case Study

- BOSH is “node-centric” while ‘service-centric’ is needed
  - No explicit control of node ordering except for horizontally scaled nodes (canary pattern)
  - Node restore does not directly provide for updates to other impacted nodes in service
  - Limited way to specify configuration dependencies between nodes in a service

- Lack of support for standard Linux packaging
Prototype Architecture and Approach
Orchestrator + BOSH Approach

- Use orchestrator ‘above’ BOSH that provides one coordinated mechanism to
  - Provision/discover IaaS network
  - Deploy BOSH director on IaaS network
  - Address gaps that BOSH has for service orchestration
Prototype Architecture
name: spark-zeppelin-cluster
nodes:
  master:
    components:
    - spark::master
    - zeppelin::server
  workers:
    components:
    - spark::worker
    links:
      - spark_master: master/spark::master

workflows:
create:
stages:
  - name: spark master
    components:
      - spark::master
  - name: spark workers
    components:
      - spark::worker
Bigtop Packages in BOSH Env.

- **NAT instance**
  - To enable nodes to connect to Bigtop package repo
  - To port map to Zeppelin service

- **DTK node agent pulls and executes Bigtop configuration logic to install and configure Bigtop services**
Connectivity through NAT instance

- BOSH AWS Director
- NAT instance
- Spark master Zeppelin
- Spark worker
- Spark worker
- AWS VPC
DTK - BOSH - AWS Interaction

DTK

- Discovers VPC subnets
- Spins up NAT gateway and BOSH director
- BOSH manifest: bare-bone nodes w/DTK agents
- Node state
- Executes a workflow to provision a Spark cluster

BOSH

- Spins up NAT gateway and BOSH director
- BOSH manifest: bare-bone nodes w/DTK agents
- Node state
- Executes a workflow to provision a Spark cluster

AWS Service

- Spin up EC2 nodes

EC2 Nodes w/DTK Agents
DTK – BOSH - AWS Interaction

• DTK discovers VPC public and private subnets and spins up NAT gateway

• DTK spins up node with a BOSH AWS director
  – Spins up proxy node and installs bosh-cli and bosh-init,
  – Generates a manifest for bosh-init to deploy BOSH director

• DTK creates and executes a workflow to deploy nodes and provision a Spark cluster
  – Generates and sends manifest to BOSH director to spin up bare-bone nodes with DTK agents on them
  – DTK interacts with its agents to bring up the data services
Lessons Learned and Suggested Directions

• It was straightforward to integrate BOSH with the DTK orchestrator

• More time than expected porting scripts to BOSH
  – BOSH can benefit from adding support to use/import standard packages

• Can view the BOSH compile/deploy workflow as providing immutable infrastructure (‘nailed bits’, predictability)
  – Can view BOSH packaging/layout as one way to achieve this
  – We think BOSH model is amenable to treat standard packages