Streamline Hadoop DevOps with Apache Ambari

Alejandro Fernandez

May 18, 2017
Speaker

Alejandro Fernandez

Staff Software Engineer @ Hortonworks

Apache Ambari PMC

alejandro@apache.org
WHY ARE WE HERE?

“WORKING FROM MIAMI”
What is Apache Ambari?

Apache Ambari is the open-source platform to deploy, manage and monitor Hadoop clusters.
Single Pane of Glass for Hadoop
20.5k commits over 4.5 years by 80 committers/contributors

AND GROWING

# of Jiras
Ambari Releases

April ‘15
Jul-Sep ‘15
Dec ’15-Feb ’16
Aug-Nov ’16
Mar’17

Ambari 2.0
Ambari 2.1
Ambari 2.2
Ambari 2.4
Ambari 2.5

Release GA
Release #.#.1
Release #.#.2

NEW
Exciting Enterprise Features in Ambari 2.5

Service Features
- AMBARI-2330: Service Auto-Restart
- AMBARI-19275: Download All Client Configs
- AMBARI-7748: Manage JournalNode HA

Security
- AMBARI-18650: Password Credential Store
- AMBARI-18365: API Authentication Using SPNEGO

Core
- AMBARI-18731: Scale Testing on 2500 Agents
- AMBARI-18990: Self-Heal DB Inconsistencies

Ambari Metrics System
- AMBARI-17859: New Grafana dashboards
- AMBARI-15901: AMS High Availability
- AMBARI-19320: HDFS TopN User and Operation Visualization

Alerts & Log Search
- AMBARI-19257: Built-in SNMP Alert
- AMBARI-16880: Simplified Log Rotation Configs
Simply Operations - Lifecycle

- Deploy
- Secure/LDAP
- Monitor
- Scale, Extend, Analyze
- Upgrade
- Smart Configs
Deploy On Premise

Mix-and-Match

- Red Hat
- Linux
- Ubuntu
- openSUSE
- Java 7
- Java 8
- PostgreSQL
- Microsoft SQL Server
- Oracle Database
- MySQL
Deploy On The Cloud

Certified environments
Sysprepped VMs
Hundreds of similar clusters
Ephemeral workloads

Amazon Web Services
Microsoft Azure
Google Cloud Platform
OpenStack
Deploy with Blueprints

• Systematic way of defining a cluster

[Diagram: Configs + Topology + Hosts = Cluster]

• Export existing cluster into blueprint
  ➜ /api/v1/clusters/:clusterName?format=blueprint
# Create a cluster with Blueprints

<table>
<thead>
<tr>
<th>1. POST /api/v1/blueprints/my-blueprint</th>
</tr>
</thead>
<tbody>
<tr>
<td>```json</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>```</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>2. POST /api/v1/clusters/my-cluster</th>
</tr>
</thead>
<tbody>
<tr>
<td>```json</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>```</td>
</tr>
</tbody>
</table>
Create a cluster with Blueprints

1. POST /api/v1/blueprints/my-blueprint

```
{
  "configurations" : [
    {
      "hdfs-site" : {
        "dfs.datanode.data.dir" : "/hadoop/1,
                    "/hadoop/2,/hadoop/3"
      }
    },
    "host_groups" : [
      {
        "name" : "master-host",
        "components" : [ {
          "name" : "NAMENODE" }, {
          "name" : "RESOUCEDMANAGER" },
        ...
        ],
        "cardinality" : "1"
      },
      {
        "name" : "worker-host",
        "components" : [ {"name" : "DATANODE" }, {
          "name" : "NODMANAGER" },
        ...
        ],
        "cardinality" : "1+
      }
    ],
    "Blueprints" : {
      "stack_name" : "HDP",
      "stack_version" : "2.5"
    }
  }
}
```

2. POST /api/v1/clusters/my-cluster

```
{
  "blueprint" : "my-blueprint",
  "host_groups" : [
    {
      "name" : "master-host",
      "hosts" : [
        { "fgdn" : "master001.ambari.apache.org" }
      }
    },
    {
      "name" : "worker-host",
      "hosts" : [
        { "fgdn" : "worker001.ambari.apache.org" },
        { "fgdn" : "worker002.ambari.apache.org" },
        ...
        {
          "fgdn" : "worker099.ambari.apache.org"
        }
      }
    }
  ]
}
```
Create a cluster with Blueprints

1. POST /api/v1/blueprints/my-blueprint

```
{
  "configurations" : [
    {
      "hdfs-site" : {
        "dfs.datanode.data.dir" : "/hadoop/1, /hadoop/2, /hadoop/3"
      }
    },
    "host_groups" : [
      {
        "name" : "master-host",
        "components" : [
          { "name" : "NAMENODE" },
          { "name" : "RESOURCEMANAGER" },
          ...
        ],
        "cardinality" : "1"
      },
      {
        "name" : "worker-host",
        "components" : [
          { "name" : "DATANODE" },
          { "name" : "NODEMANAGER" },
          ...
        ],
        "cardinality" : "1+"
      }
    ],
    "Blueprints" : {
      "stack_name" : "HDP",
      "stack_version" : "2.5"
    }
  }
}
```

2. POST /api/v1/clusters/my-cluster

```
{
  "blueprint" : "my-blueprint",
  "host_groups" : [
    {
      "name" : "master-host",
      "hosts" : [
        { "fqdn" : "master001.ambari.apache.org"
      }
    },
    {
      "name" : "worker-host",
      "hosts" : [
        { "fqdn" : "worker001.ambari.apache.org" },
        { "fqdn" : "worker002.ambari.apache.org" },
        ...
      ]
    }
  ]
}
```
Create a cluster with Blueprints

1. POST /api/v1/blueprints/my-blueprint

```json
{
    "configurations" : [ 
        
    
    
    
    
    ],
    "host_groups" : [ 
        
    
    
    
    ],
    "Blueprints" : { 
        "stack_name" : "HDP",
        "stack_version" : "2.5"
    }
}
```

2. POST /api/v1/clusters/my-cluster

```json
{
    "blueprint" : "my-blueprint",
    "host_groups" : [ 
        
    
    
    
    ],
    "Blueprints" : { 
        "stack_name" : "HDP",
        "stack_version" : "2.5"
    }
}
```
Blueprints for Large Scale

- **Kerberos**, secure out-of-the-box

- **High Availability** is setup initially for NameNode, YARN, Hive, Oozie, etc

- **Host Discovery** allows Ambari to automatically install services for a Host when it comes online

- **Stack Advisor** for config recommendations
POST /api/v1/clusters/MyCluster/hosts

[
  {
    "blueprint": "single-node-hdfs-test2",
    "host_groups": [
      {
        "host_group": "worker",
        "host_count": 3,
        "host_predicate": "Hosts/cpu_count>1"
      },
      {
        "host_group": "super-worker",
        "host_count": 5,
        "host_predicate": "Hosts/cpu_count>2 & Hosts/total_mem>3000000"
      }
    ]
  }
]
Service Layout

Common Services
- HDFS
  - 2.1.0.2.0
    - configuration
    - package
      - alerts.json
      - kerberos.json
      - metainfo.xml
      - metrics.json
      - widgets.json

Stack Override
- 2.3
  - repos
  - services
    - HDFS
      - configuration
        - hadoop-env.xml
        - hdfs-site.xml
        - ranger-hdfs-audit.xml
        - ranger-hdfs-plugin-properties.xml
        - ranger-hdfs-policymgr-ssl.xml
        - ranger-hdfs-security.xml
    - quicklinks
      - quicklinks.json
      - metainfo.xml
      - widgets.json
Custom Service

Starter Pack:
- metainfo.xml
- Python scripts: lifecycle management
- Configs: key, value, description, allow empty, password, etc.

- Templates: Jinja template with config replacement
- Role Command Order: dependency of start, stop commands
- Service Advisor: recommend/validate configs on changes
- Kerberos: principals and keytabs, configs to change when Kerberized
- Widgets: UI config knobs, sections
- Alerts: definition, type: [port, web, python script], interval
- Metrics: for Ambari Metrics System
<service>
  <name>SAMPLESRV</name>
  <displayName>New Sample Service</displayName>
  <comment>A New Sample Service</comment>
  <version>1.0.0</version>
  <components>
    <component>
      <name>SAMPLESRV_MASTER</name>
      <displayName>Sample Srv Master</displayName>
      <category>MASTER</category>
      <cardinality>1</cardinality>
      <commandScript>
        <script>scripts/master.py</script>
        <scriptType>PYTHON</scriptType>
        <timeout>600</timeout>
      </commandScript>
    </component>
    <component>
      <name>SAMPLESRV_SLAVE_OR_CLIENT</name>
      <displayName>Sample Slave or Client</displayName>
      <category>SLAVE | CLIENT</category>
      <cardinality>0+ | 0-1 | 1 | 1+</cardinality>
      <commandScript>
        <script>scripts/slave_or_client.py</script>
        <scriptType>PYTHON</scriptType>
        <timeout>600</timeout>
      </commandScript>
    </component>
  </components>
</service>
...<customCommand>
  <name>DECOMMISSION</name>
  <commandScript>
    <script>scripts/decommission.py</script>
    <scriptType>PYTHON</scriptType>
    <timeout>1200</timeout>
  </commandScript>
</customCommand>

<dependency>
  <name>HDFS/NAMENODE</name>
  <scope>cluster | host</scope>
  <auto-deploy>
    <enabled>true | false</enabled>
  </auto-deploy>
</dependency>

...<requiredServices>
  <service>HDFS</service>
</requiredServices>
Custom Service – metainfo.xml

```xml
...<configuration-dependencies>
  <config-type>service-env</config-type>
  <config-type>service-site</config-type>
  <config-type>hdfs-site</config-type>
</configuration-dependencies>

<osSpecifics>
  <osSpecific>
    <osFamily>any</osFamily>
    <packages>
      <package>
        <name>rpm_apt_pkg_name</name>
      </package>
    </packages>
  </osSpecific>
</osSpecifics>
```
import sys
from resource_management import Script

class Master(Script):
    
def install(self, env):
        print 'Install the Sample Srv Master'
def stop(self, env):
        print 'Stop the Sample Srv Master'
def start(self, env):
        print 'Start the Sample Srv Master'
def status(self, env):
        print 'Status of the Sample Srv Master'
def configure(self, env):
        print 'Configure the Sample Srv Master'

if __name__ == "__main__":
    Master().execute()
# Configurations

- Kerberos
- HTTPS
- Zookeeper Servers
- Memory Settings
- High Availability

## Example

```bash
# Atlas Servers
atlas.enableTLS = true|false
atlas.server.http.port = 21000
atlas.server.https.port = 21443
```

```plaintext
atlas.rest.address = http(s)://host:port
```
Service Advisors in Ambari 3.0

- Break up single Stack Advisor into 22 Service Advisors
- Rewrite in Java for stronger checking and faster speed
- Use Drools
Comprehensive Security

Kerberos
- MIT KDC
- Keytab management

LDAP/AD
- User auth
- Sync

Ranger
- Security policies
- Audit
- Authorization

Atlas
- Governance
- Compliance
- Linage & history
- Data classification

Knox
- Perimeter security
- Supports LDAP/AD
- Sec. for REST/HTTP
- SSL
Ambari manages Kerberos principals and keytabs

Works with existing MIT KDC or Active Directory

Once Kerberized, handles

- Adding hosts
- Adding components to existing hosts
- Adding services
- Moving components to different hosts
Testing at Scale: 3000 Agents

- Each Agent has own hostname, home dir, log dir, PID, ambari-agent.ini file
- Agent Multiplier can bootstrap 50 Agents per VM
- Tried Docker + Weave before and not very stable for networking
Testing at Scale: 3000 Agents

Ambari Server

Testing
- Scale (server cannot tell the difference)
- Kerberos
- Stack Advisor
- Alerts
- Rolling & Express Upgrade
- UI

PERF Stack

Dummy Services
- Happy: always passes
- Sleepy: always timesout
- Grumpy: always fails
- Zookeeper
- HDFS
- YARN
- HBASE
<table>
<thead>
<tr>
<th>Name</th>
<th>IP Address</th>
<th>Rack</th>
<th>Cores</th>
<th>RAM</th>
<th>Disk Usage</th>
<th>Load Avg</th>
<th>Versions</th>
<th>Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001.ambari.apache.org</td>
<td>192.168.64.101</td>
<td>/default-rack</td>
<td>1 (1)</td>
<td>2.79GB</td>
<td></td>
<td></td>
<td>PERF-1.0</td>
<td>2 Components</td>
</tr>
<tr>
<td>0002.ambari.apache.org</td>
<td>192.168.64.101</td>
<td>/default-rack</td>
<td>1 (1)</td>
<td>2.79GB</td>
<td></td>
<td></td>
<td>PERF-1.0</td>
<td>2 Components</td>
</tr>
<tr>
<td>0003.ambari.apache.org</td>
<td>192.168.64.101</td>
<td>/default-rack</td>
<td>1 (1)</td>
<td>2.79GB</td>
<td></td>
<td></td>
<td>PERF-1.0</td>
<td>2 Components</td>
</tr>
<tr>
<td>0004.ambari.apache.org</td>
<td>192.168.64.101</td>
<td>/default-rack</td>
<td>1 (1)</td>
<td>2.79GB</td>
<td></td>
<td></td>
<td>PERF-1.0</td>
<td>2 Components</td>
</tr>
<tr>
<td>0005.ambari.apache.org</td>
<td>192.168.64.101</td>
<td>/default-rack</td>
<td>1 (1)</td>
<td>2.79GB</td>
<td></td>
<td></td>
<td>PERF-1.0</td>
<td>2 Components</td>
</tr>
<tr>
<td>0006.ambari.apache.org</td>
<td>192.168.64.101</td>
<td>/default-rack</td>
<td>1 (1)</td>
<td>2.79GB</td>
<td></td>
<td></td>
<td>PERF-1.0</td>
<td>2 Components</td>
</tr>
</tbody>
</table>

**Summary**

- Group: Default (1)
- Service Actions
- Manage Config Groups

**Config**

- **V1**
- admin authored on Fri, Oct 21, 2016 16:36
- Discard
- Save

**General**

- Features
  - Success percentage: 100%

**Settings**

- Advanced
Optimize for Large Scale

- Dedicated database server with SSD
- MySQL 5.7 and DB tuning
- Purge old Ambari history: commands, alerts, BP topology, upgrades.

```
export AMBARI_JVM_ARGS=$AMBARI_JVM_ARGS' -Xms2048m -Xmx8192m
```

<table>
<thead>
<tr>
<th></th>
<th>10 Hosts</th>
<th>50 Hosts</th>
<th>100 Hosts</th>
<th>&gt; 500 Hosts</th>
</tr>
</thead>
<tbody>
<tr>
<td>agent.threadpool.size.max</td>
<td>25</td>
<td>35</td>
<td>75</td>
<td>100</td>
</tr>
<tr>
<td>alerts.cache.enabled</td>
<td></td>
<td></td>
<td>true</td>
<td></td>
</tr>
<tr>
<td>alerts.cache.size</td>
<td></td>
<td>50000</td>
<td></td>
<td>100000</td>
</tr>
<tr>
<td>alerts.execution.scheduler.maxThreads</td>
<td></td>
<td></td>
<td>2</td>
<td>4</td>
</tr>
</tbody>
</table>

https://community.hortonworks.com/articles/80635/optimize-ambari-performance-for-large-clusters.html
Background: Upgrade Terminology

Manual Upgrade

- The user follows instructions to upgrade the stack
- Incurs downtime
### Background: Upgrade Terminology

#### Rolling Upgrade
- **Automated**
- **Upgrades one component per host at a time**
- **Preserves cluster operation and minimizes service impact**

#### Manual Upgrade
- **The user follows instructions to upgrade the stack**
- **Incurs downtime**
## Background: Upgrade Terminology

<table>
<thead>
<tr>
<th>Upgrade Type</th>
<th>Automation</th>
<th>Parallel Upgrade</th>
<th>Downtime</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Express Upgrade</strong></td>
<td>✔️ Automated</td>
<td>✔️ Runs in parallel</td>
<td>✗ Incurs downtime</td>
</tr>
<tr>
<td><strong>Rolling Upgrade</strong></td>
<td>✔️ Automated</td>
<td>✗ Upgrades one component per host</td>
<td>✔️ Preserves cluster operation and minimizes service impact</td>
</tr>
<tr>
<td><strong>Manual Upgrade</strong></td>
<td>✗ The user follows instructions</td>
<td>✗ Incurs downtime</td>
<td></td>
</tr>
</tbody>
</table>
Automated Upgrade: Rolling or Express

1. **Check Prerequisites**
   - Review the prereqs to confirm your cluster configs are ready

2. **Prepare**
   - Take backups of critical cluster metadata

3. **Register + Install**
   - Register the HDP repository and install the target HDP version on the cluster

4. **Perform Upgrade**
   - Perform the HDP upgrade. The steps depend on upgrade method: Rolling or Express

5. **Finalize**
   - Finalize the upgrade, making the target version the current version
Process: Rolling Upgrade

- ZooKeeper
- Ranger/KMS
- Kafka
- Core Masters
- Core Slaves
- Hive
- Spark
- Oozie
- Falcon
- Clients
- Knox
- Storm
- Slider
- Flume
- Accumulo
- HDFS, YARN, MR, Tez, HBase, Pig, Hive, etc.

On Failure:
- Retry
- Ignore
- Downgrade

Finalize or Downgrade
Process: Express Upgrade

- Stop High-Level: Spark, Storm, etc
- Back up HDFS, HBase, Hive
- Stop Low-Level: YARN, MR, HDFS, ZK
- Change Stack + Configs
- Zookeeper
- Ranger/KMS
- HDFS
- YARN
- MapReduce2
- HBase
- Hive
- Oozie
- Falcon
- Knox
- Storm
- Slider
- Flume
- Accumulo
- Finalize or Downgrade

On Failure,
- Retry
- Ignore
- Downgrade

Hosts in Parallel
- 1
- 100
- 1
- 100
Total Time: 2:53 13:16 26:26

Scales linearly with # of hosts
Scales linearly with # batches (defaults to 100 hosts at a time)
Upgrade Endpoint

Status:
http://server:8080|8443/api/v1/clusters/$name/upgrades

Navigate
Groups: Core Master, Core Slaves, …, Post Cluster, etc.
Items: Upgrading Zookeeper Server on host namenode.apache.org

Upgrade: {
  cluster_name: "c1",
  create_time: 1495002598280,
  direction: "UPGRADE",
  downgrade_allowed: true,
  end_time: 1495003244035,
  exclusive: false,
  from_version: "2.5.3.0-37",
  pack: "nonrolling-upgrade-2.5",
  progress_percent: 100,
  request_context: "Upgrading to 2.5.5.0-157",
  request_id: 6,
  request_status: "COMPLETED",
  skip_failures: false,
  skip_service_check_failures: false,
  start_time: 1495002598412,
  suspended: false,
  to_version: "2.5.5.0-157",
  type: "INTERNAL_REQUEST",
  upgrade_type: "NON_ROLLING"}
Upgrade – Debugging with SQL

```sql
SELECT u.upgrade_id, u.direction, u.from_version, u.to_version, hrc.request_id, hrc.task_id, substr(g.group_title, 0, 30), substr(i.item_text, 0, 30), hrc.status
FROM upgrade u JOIN upgrade_group g ON g.upgrade_id = u.upgrade_id
JOIN upgrade_item i ON i.upgrade_group_id = g.upgrade_group_id
JOIN host_role_command hrc ON hrc.stage_id = i.stage_id AND hrc.request_id = u.request_id
ORDER BY hrc.task_id;
```
## Alerting Framework

<table>
<thead>
<tr>
<th>Alert Type</th>
<th>Description</th>
<th>Thresholds (units)</th>
</tr>
</thead>
<tbody>
<tr>
<td>WEB</td>
<td>Connects to a Web URL. Alert status is based on the HTTP response code</td>
<td>Response Code (n/a)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Connection Timeout (seconds)</td>
</tr>
<tr>
<td>PORT</td>
<td>Connects to a port. Alert status is based on response time</td>
<td>Response (seconds)</td>
</tr>
<tr>
<td>METRIC</td>
<td>Checks the value of a service metric. Units vary, based on the metric being checked</td>
<td>Metric Value (units vary)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Connection Timeout (seconds)</td>
</tr>
<tr>
<td>AGGREGATE</td>
<td>Aggregates the status for another alert</td>
<td>% Affected (percentage)</td>
</tr>
<tr>
<td>SCRIPT</td>
<td>Executes a script to handle the alert check</td>
<td>Varies</td>
</tr>
<tr>
<td>SERVER</td>
<td>Executes a server-side runnable class to handle the alert check</td>
<td>Varies</td>
</tr>
</tbody>
</table>
Motivation Behind Ambari Metrics System

- Limited Ganglia capabilities
- OpenTSDB – GPL license and needs a Hadoop cluster
- Aggregation at multiple levels: Service, Time
- Alerts based on metrics system
- Scale past a 1000 nodes
- Analytics based on use cases
- Fine grained control over retention, collection intervals, aggregation
- Pluggable and Extensible
AMS Architecture

- Custom Sinks – HDFS, YARN, HBase, Storm, Kafka, Flume, Accumulo
- Monitors – lightweight daemon for system metrics
- Collector – API daemon + HBase (embedded / distributed)
- Phoenix schema designed for fast reads
- Managed HBase
- Grafana support from version 2.2.2
AMS Features

- Simple POST API for sending metrics.
- Rich GET API to fetch metrics in specific granularity
  - Point in time & series
  - Top N support
  - Rate support
- Performs Host level aggregation as well as time based down sampling
- Highly tunable system
  - Adjust rate of collecting/sending metrics
  - Adjust granularity of data being stored
  - Skip Aggregation for certain metrics
  - Whitelist metrics
- Metadata API that provides information on what metrics are being collected and which component is sending these metrics
- Abstract Sink implementation to facilitate easy integration with metrics collector
- HTTPS Support
Grafana for Ambari Metrics

FEATURES

• Grafana as a “Native UI” for Ambari Metrics
• Pre-built Dashboards Host-level, Service-level
• Supports HTTPS

DASHBOARDS

• System Home, Servers
• HDFS Home, NameNodes, DataNodes
• YARN Home, Applications, Job History Server
• HBase Home, Performance
Log Search

Search and index HDP logs!

Capabilities

- Rapid Search of all HDP component logs
- Search across time ranges, log levels, and for keywords
Log Search

- **Log Feeder**
- **Worker Node**
- **Java Process**
- **Multi-output Support**
- **Grok filters**

- **Ambhari**

- **Solr Cloud**
- **Local Disk Storage**

- **Log Search UI**
Future of Apache Ambari 3.0

- Cloud features
- Service multi-instance (e.g., two ZK quorums)
- Service multi-versions (Spark 2.0 & Spark 2.2)
- YARN assemblies & services
- Patch Upgrades: upgrade individual components in the same stack version, e.g., just DN and RM in HDP 3.0.*.* with zero downtime
- Ambari High Availability
Resources

Contribute to Ambari:
https://cwiki.apache.org/confluence/display/AMBARI/Quick+Start+Guide

Referenced Articles:
https://community.hortonworks.com/articles/43816/how-to-createadd-the-service-stop-the-service.html

https://community.hortonworks.com/articles/80635/optimize-ambari-performance-for-large-clusters.html

Image Sources:

https://ak9.picdn.net/shutterstock/videos/2139614/thumb/1.jpg

Many thanks to the ASF, audience, and event organizers.