Mt. Oxford
Mon 4:50 pm - 5:30 pm

Introducing Montecristo: The Cassandra Cluster Health Check Tool

Dave Herrington

Chief Engineer
RhinoSource, Inc.

COMMUNITY
THE ASF CONFERENCE
CODE

Denver, Colorado
October 7-10, 2024

RhinoSource
THE APACHE SOFTWARE FOUNDATION

Presentation Overview

Introducing Montecristo: The Cassandra Cluster Health Check Tool

Dave Herrington

Chief Engineer
RhinoSource, Inc.

COMMUNITY OVER CODE
THE ASF CONFERENCE
Denver, Colorado
October 7-10, 2024

DataStax has recently published Montecristo, their Cassandra/DSE cluster health check tool, to datastax-labs on GitHub.

This session provides an overview of its features, the steps to install and run it, and helpful tips and tricks to get it working.

Meet Dave Herrington

- University of Illinois at Urbana-Champaign - Electrical Engineering
- Oracle Corp. – Oracle Financials FINCON Team
- Founder DARC Corp. – Oracle E-Business Suite Platinum Partner
- CIO Parelli – Global training provider, Netflix streaming model
- Founder & Chief Engineer at RhinoSource (since 2012)
  - Located in San Francisco Bay Area, CA
  - DataStax Enterprise and Cassandra specialization
  - Have helped many large & small DataStax customers
  - Cluster architecture, data model validation, load & stress testing, health checkups, upgrades, troubleshooting
  - Key Partners: DataStax, Innominds, AxonOps and AWS

www.rhinosource.com

Presentation Outline

- Presentation Overview
- History and Contributor Credits
- Health Check Process Overview
- Building, Deploying and Running ds-collector
- Installing and Running Montecristo
- Q&A

Products Covered Today

- DataStax Diagnostic Collector (ds-collector)
  – Apache-2.0 License
  – On GitHub

- DataStax Montecristo
  – Apache-2.0 License
  – On GitHub

History of Montecristo

- Montecristo was originally created as an internal project by The Last Pickle, who were acquired by Datastax in 2020.

- Datastax Professional Services adopted and further enhanced the project into the codebase that is published in GitHub.

Author & Contributor Credits

- DataStax Diagnostic Collector (ds-collector)
  - Andrew Hogg
  - Alex Ott
  - Michael Semb Wever
  - Madhavan Sridharan
  - Phil Miesle
  - Radovan Zevoncek
- Montecristo
  - Andrew Hogg
  - Michael Semb Wever
  - Jeremy Artero
  - Pierre-Yves de Britto
  - Edward Capriolo
  - Joaquin Casares
  - Alexander Dejanovski
  - Rebecca Downes
  - Marvin Froeder
  - Miles Garnsey
  - Anthony Grasso
  - Jon Haddad
  - Alexander Dejanovski
  - Brendan Cicchi
  - Anthony Grasso
  - Joel Serrano
  - Jon Moses
  - Romain Anselin
  - Jeremy Hanna
  - Nate McCall
  - Phil Miesle
  - Kareem Missoumi
  - Yuki Morishita
  - Aaron Morton
  - Alex Ott
  - Alain Rodriguez
  - John Sanda
  - Radovan Zvoncek

Cassandra Health Check Process Overview

Health Checks???

- **What is a cluster health check?**
  - A periodic analysis of cluster configuration and performance to highlight problems that need to be addressed.
- **How does it work?**
  - We collect diagnostic and configuration files from the nodes of a DSE or Cassandra cluster (“diagnostic tarballs” aka “diagnostic snapshots” aka “support bundles”)
  - Then, we analyze this data set with automated scripts to identify issues and areas of improvement.
- **What is the output/deliverable?**
  - A detailed report with a summary of recommendations:
    - Immediate, Near-Term and Long-Term

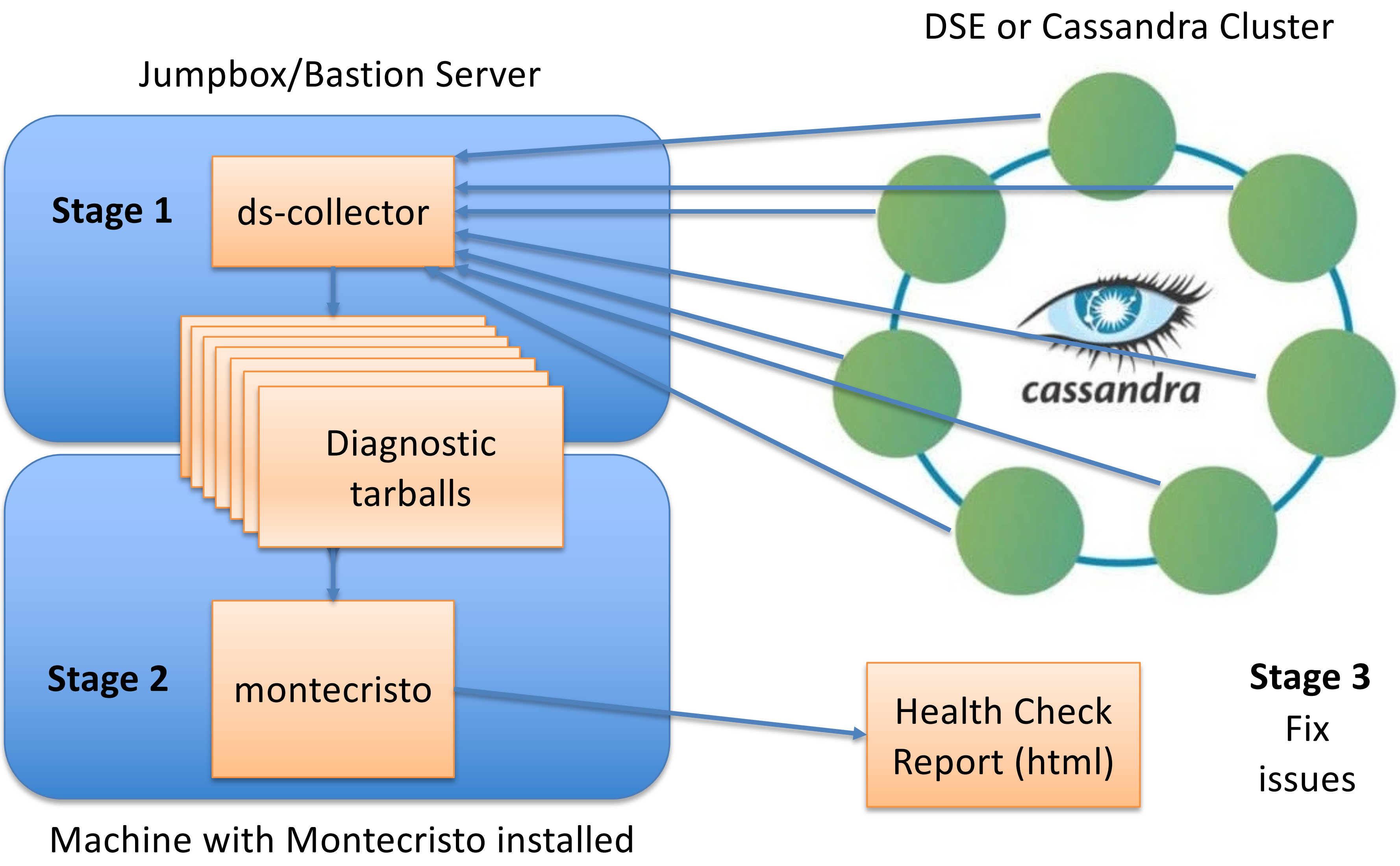
Health Check Process

- **Stage 1**
  - Collect Diagnostic Tarballs from DSE or Cassandra cluster using the open source DataStax Diagnostic Collector: ***ds-collector***
- **Stage 2**
  - Run open source ***Montecristo*** on the collected tarballs to Produce Health Check Report
- **Stage 3**
  - Fix your cluster issues

- *(repeat every 6-12 months)*

- *DataStax formerly would do this service for customers, but now companies must do it themselves or have a DataStax partner like RhinoSource perform these health checks for them.*

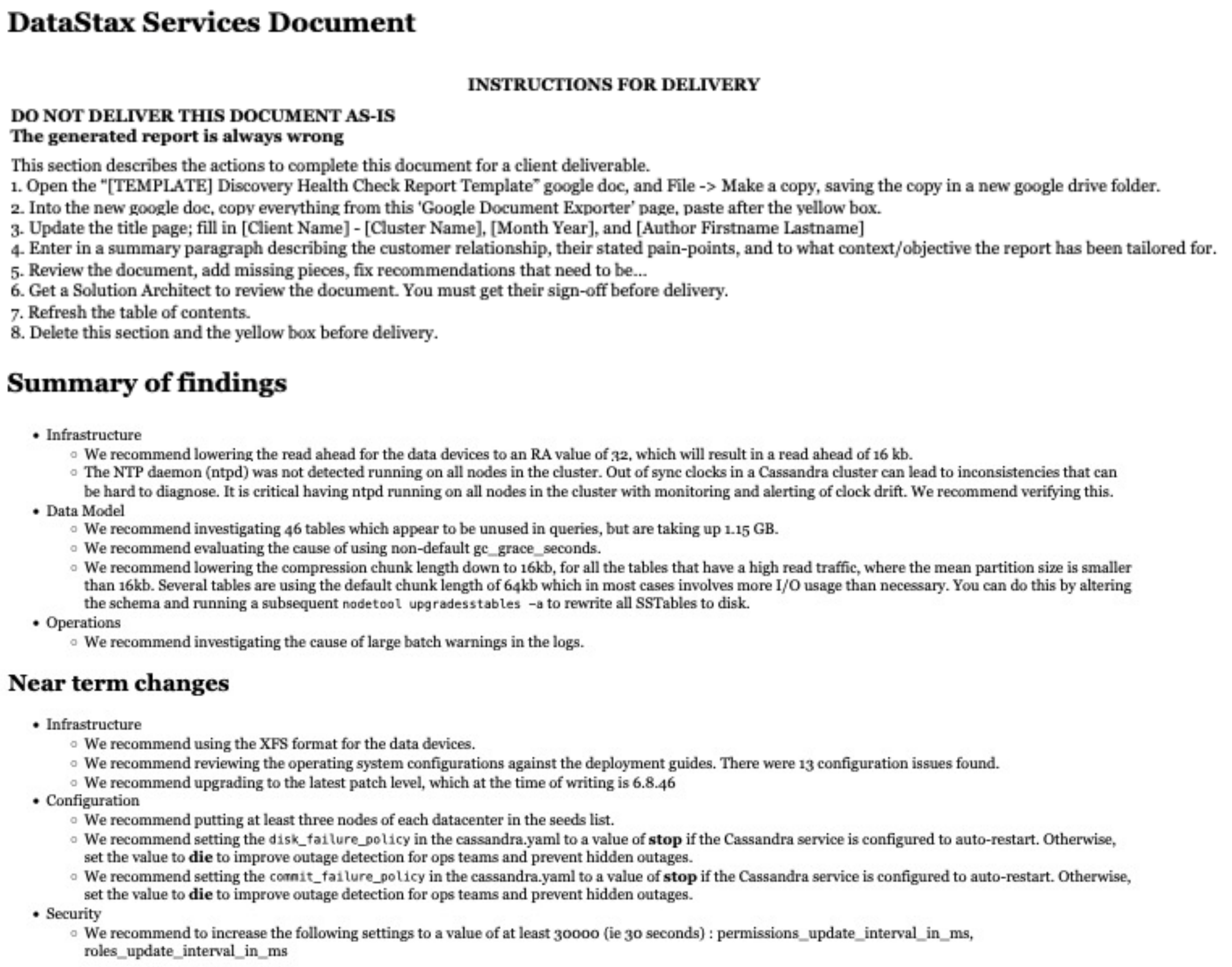
Health Check Process



RhinoSource

THE APACHE SOFTWARE FOUNDATION

Output: Health Check Report



DataStax Services Document

INSTRUCTIONS FOR DELIVERY

DO NOT DELIVER THIS DOCUMENT AS-IS
The generated report is always wrong

This section describes the actions to complete this document for a client deliverable.
1. Open the "[TEMPLATE] Discovery Health Check Report Template" google doc, and File -> Make a copy, saving the copy in a new google drive folder.
2. Into the new google doc, copy everything from this 'Google Document Exporter' page, paste after the yellow box.
3. Update the title page; fill in [Client Name] - [Cluster Name], [Month Year], and [Author Firstname Lastname]
4. Enter in a summary paragraph describing the customer relationship, their stated pain-points, and to what context/objective the report has been tailored for.
5. Review the document, add missing pieces, fix recommendations that need to be...
6. Get a Solution Architect to review the document. You must get their sign-off before delivery.
7. Refresh the table of contents.
8. Delete this section and the yellow box before delivery.

Summary of findings

- Infrastructure
  - We recommend lowering the read ahead for the data devices to an RA value of 32, which will result in a read ahead of 16 kb.
  - The NTP daemon (ntpd) was not detected running on all nodes in the cluster. Out of sync clocks in a Cassandra cluster can lead to inconsistencies that can be hard to diagnose. It is critical having ntpd running on all nodes in the cluster with monitoring and alerting of clock drift. We recommend verifying this.
- Data Model
  - We recommend investigating 46 tables which appear to be unused in queries, but are taking up 1.15 GB.
  - We recommend evaluating the cause of using non-default gc_grace_seconds.
  - We recommend lowering the compression chunk length down to 16kb, for all the tables that have a high read traffic, where the mean partition size is smaller than 16kb. Several tables are using the default chunk length of 64kb which in most cases involves more I/O usage than necessary. You can do this by altering the schema and running a subsequent `nodetool upgradesstables -a` to rewrite all SSTables to disk.
- Operations
  - We recommend investigating the cause of large batch warnings in the logs.

Near term changes

- Infrastructure
  - We recommend using the XFS format for the data devices.
  - We recommend reviewing the operating system configurations against the deployment guides. There were 13 configuration issues found.
  - We recommend upgrading to the latest patch level, which at the time of writing is 6.8.46
- Configuration
  - We recommend putting at least three nodes of each datacenter in the seeds list.
  - We recommend setting the `disk_failure_policy` in the cassandra.yaml to a value of **stop** if the Cassandra service is configured to auto-restart. Otherwise, set the value to **die** to improve outage detection for ops teams and prevent hidden outages.
  - We recommend setting the `commit_failure_policy` in the cassandra.yaml to a value of **stop** if the Cassandra service is configured to auto-restart. Otherwise, set the value to **die** to improve outage detection for ops teams and prevent hidden outages.
- Security
  - We recommend to increase the following settings to a value of at least 30000 (ie 30 seconds) : permissions_update_interval_in_ms, roles_update_interval_in_ms

> 60 pages

Created in HTML, can save a PDF.

Can convert the PDF to Word doc.

Health Check Report Contents

- Summary of findings
  - Immediate, Near-term and Long-term changes
- Infrastructure Overview and Recommendations
  - Cluster Node Listing
  - Storage Configuration
  - O/S Configuration
    - Swap, NTP, Java version, Limits
  - DSE or Cassandra Configuration
    - Customized settings
    - Mismatches and missing configs
    - Best practice recommendations
    - Java Heap and GC settings
  - Security Configuration
- Data Model Overview and Recommendations
  - Replication strategy
  - Indexes
  - Unused tables
  - gc_grace_seconds
  - Custom, UDT and collection data types
  - Bloom filters
  - Materialized views
  - Compaction settings
  - Table compression settings
  - Read-heavy tables, Top-10 Tables by size, by reads, by writes
- Operations Recommendations
  - Cache Usage
  - High SSTable Counts
  - Cassandra Logs Analysis
    - Blocked flush writers
    - Dropped Hints
    - Hinted Handoff Tasks
    - Dropped Messages
    - JVM Garbage Collection (GC) Pauses
    - Large batch warnings
    - Failed Repair Sessions
    - Commit Log Sync Warnings
    - Long Pause Warnings
    - Compacted Large Partitions
    - Aborted Hints
    - Tombstone Warnings
    - Aggregation query warnings
    - Prepared statement discard warnings
  - Astra Guardrail Checks
  - Table Operations
    - Read Ops/Hr by Consistency Level
    - Write Ops/Hr by Consistency Level

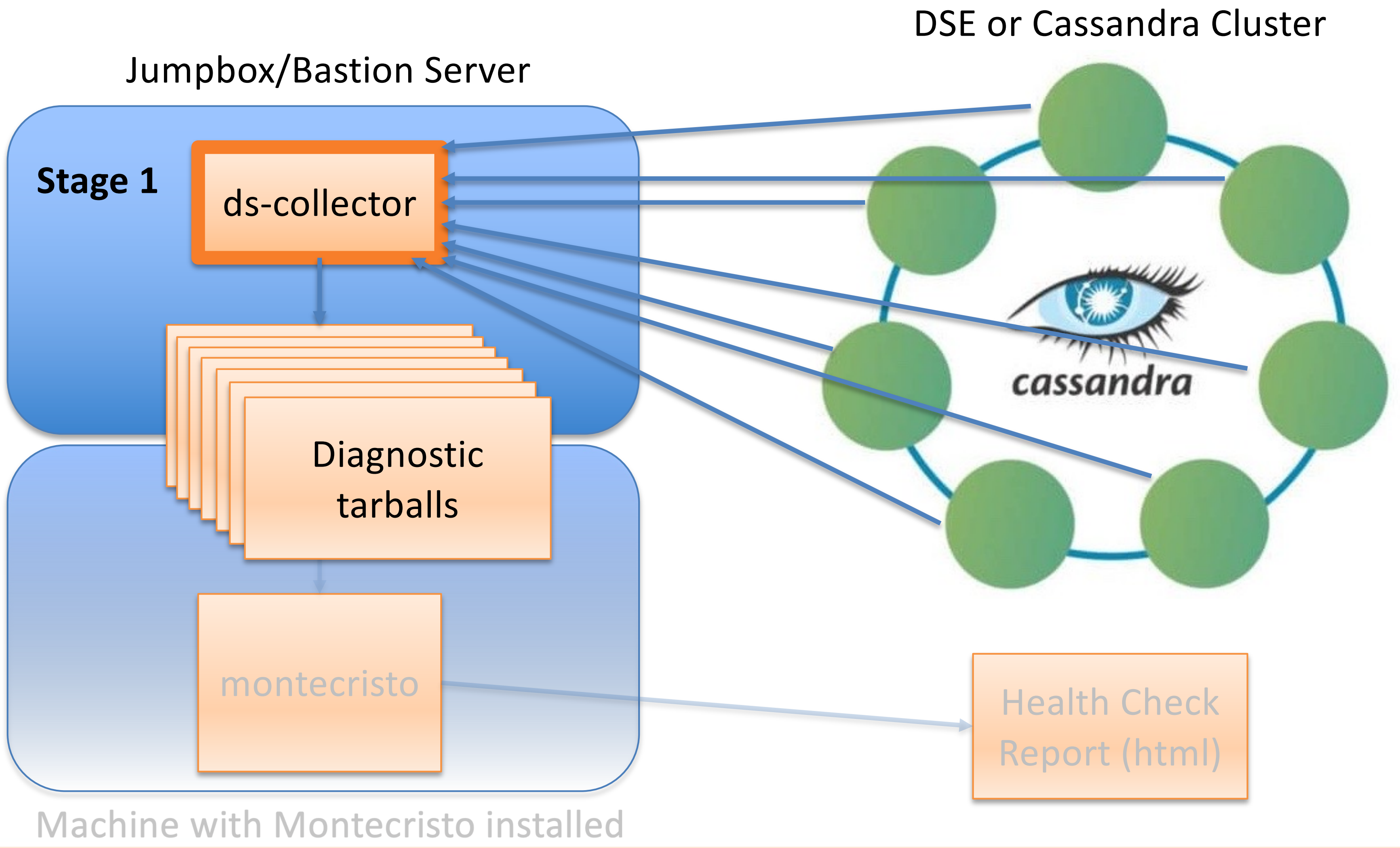
Deep Dive:
Health Check Process

Stage 1:

Diagnostic Tarball Collection

with ds-collector

Health Check Process: Stage 1: Tarball Collection



Deploying ds-collector

- Requires multiple steps:
  - Download latest source .tar.gz from
    - https://github.com/datastax/diagnostic-collection/releases
  - Build from source ← *Cannot run the source*
    - Recommend doing the build on **Mac OS X**
    - YMMV building with Linux
    - Docker must be running on the build machine
  - Deploy the built ds-collector tar.gz onto the collection machine
    - Can be Linux or Mac OS X
    - Works equally well on both
    - Yes, you can build on Mac OS X and deploy to Linux.

Building the ds-collector

1. Download source (tar.gz) from Github
   - ➢ https://github.com/datastax/diagnostic-collection/releases
2. Uncompress the source tar.gz:

```
tar xvf diag*.tar.gz
```

Link to doc

3. Set environment variables:

```
export ISSUE=DSE-001
export is_dse=true
```

←*just make up a JIRA ticket number*

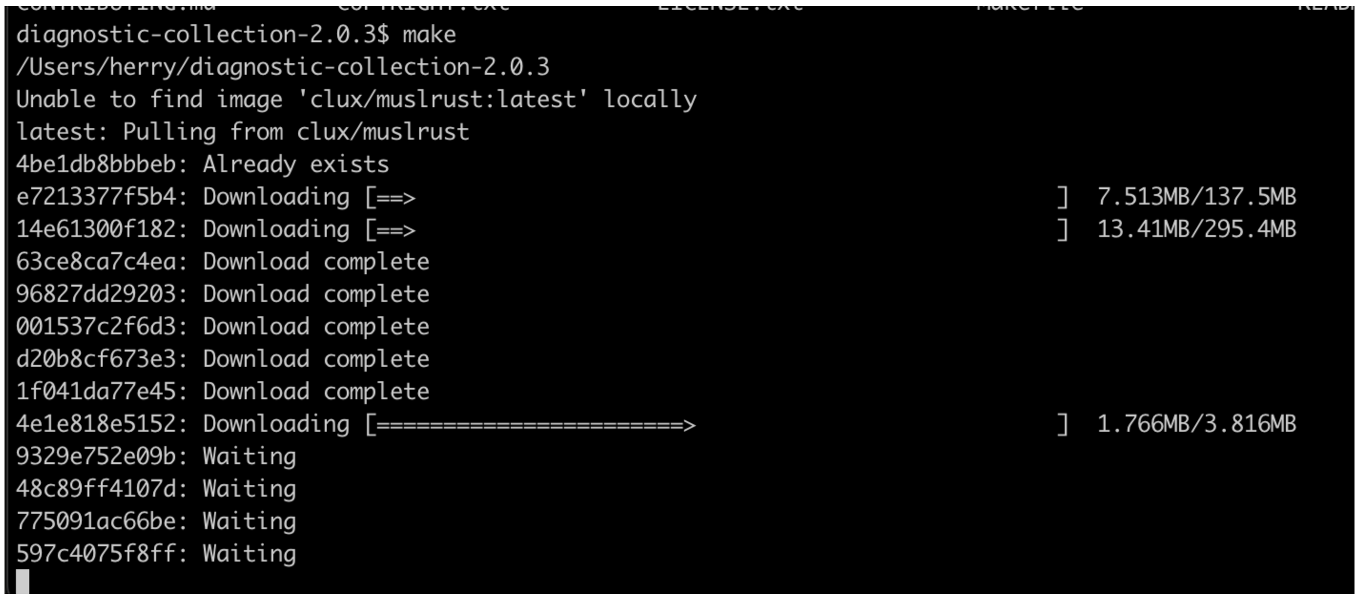
←*only if DSE*

4. Make sure Docker is installed and running on your build machine.

Building the ds-collector

5. From the uncompressed diagnostic-collection-* directory, build it using the make command:

```
make
```



```
diagnostic-collection-2.0.3$ make
/Users/herry/diagnostic-collection-2.0.3
Unable to find image 'clux/muslrust:latest' locally
latest: Pulling from clux/muslrust
4be1db8bbbeb: Already exists
e7213377f5b4: Downloading [==>                                                ]  7.513MB/137.5MB
14e61300f182: Downloading [==>                                                ]  13.41MB/295.4MB
63ce8ca7c4ea: Download complete
96827dd29203: Download complete
001537c2f6d3: Download complete
d20b8cf673e3: Download complete
1f041da77e45: Download complete
4e1e818e5152: Downloading [=======================>                           ]  1.766MB/3.816MB
9329e752e09b: Waiting
48c89ff4107d: Waiting
775091ac66be: Waiting
597c4075f8ff: Waiting
```

6. Deploy the built ds-collector.*.tar.gz that is created to the machine to be used to do the tarball collection.
   - Or, if using the same machine to both build and run the tarball collection, move it (up) outside of the build directory.
   - Again, you can build on a Mac and deploy the built collector to Linux.

Running ds-collector: req’s

System Requirements for tarball collection machine:

- Linux or macOS X machine
  - NOTE: macOS X requires coreutils and sshpass to be installed using ”brew install”.
- Stand-alone machine: Do not install ds-collector on a cluster node
- Can ssh to all cluster nodes
- ssh user on nodes should have sudo privilege
- Has docker or k8s access, if nodes run inside docker or k8s

Running ds-collector: Steps

1. Transfer the built ds-collector.*.tar.gz from the build machine to the collection machine (if not the same machine)
2. Uncompress it (tar xvf ds-collector*.tar.gz)
3. cd ./collector
4. Edit the config file collector.conf, uncommenting configs, as needed:
   - ➢ Customize for your cluster's directory locations, as needed (logHome, configHome)
   - ➢ Set your ssh parameters (userName, sshPassword)
   - ➢ Uncomment skipS3="true" (if not already uncommented), to disable the S3 transfer feature
   - ➢ Add cqlsh credentials if required (cqlshUsername and cqlshPassword)
   - ➢ If for DSE, uncomment and set is_dse="true" (if not already set correctly)
   - ➢ Browse other parameters and set if needed for your cluster deployment (e.g. cqlshSSL, dt_opts if you have extra security in place)

Running ds-collector

1. Run ds-collector in test (-T) and verbose (-v) modes to ssh & resolve any issues reported:

```
./ds-collector -T –v -f collector.conf -n <cassandra_contact_node>
```

➢ Review the log created under /tmp/datastax/ds-collector*

2. Run ds-collector to extract (-X) from 1 node only (-d) to troubleshoot any issues (looko for any NOTOK messages):

```
./ds-collector -X -d -v -f collector.conf -n <cassandra_contact_node>
```

3. See the TROUBLESHOOTING.md page and fix configurations to solve issues.
4. Run ds-collector to extract from all nodes:

```
./ds-collector -X -f collector.conf -n <cassandra_contact_node>
```

Accessing ds-collector Tarballs

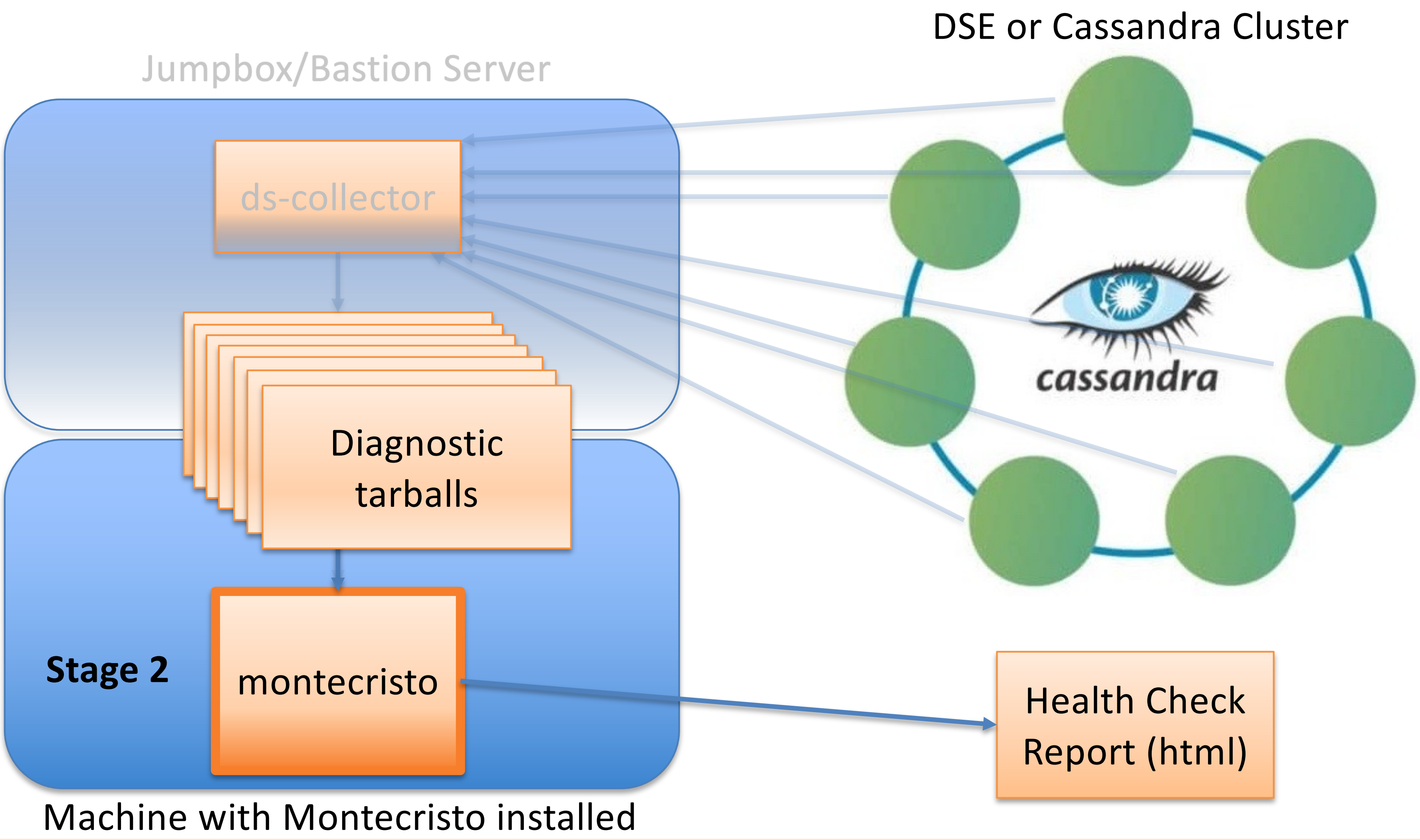
- A diagnostic tarball file is generated for each node under /tmp/datastax
  - `<host1name>_<timestamp>.tar.gz`
  - `<host2name>_<timestamp>.tar.gz`
  - `<host3name>_<timestamp>.tar.gz`
- Transfer these .tar.gz files to the machine with Montecristo installed.
- Or these files can be uploaded to DataStax or a DataStax partner to run a health check.

Deep Dive: Health Check Process

Stage 2:

Running Montecristo to Produce Health Check Report

Health Check Process: Stage 2: Running Montecristo



Montecristo

- System Requirements:
  - Linux or mac OS X machine (not a cluster node)
  - Java 8 installed
  - Docker Desktop installed
  - Hugo, zip and jq installed
  - Has access to diagnostic tarballs
- Download site:
  - https://github.com/datastax-labs/Montecristo

(Let's Cover the Install Later)

- Let's assume Montecristo is already installed.
- We'll go over the installation steps at then end.
- Let's run it!

Staging the ds-collector Tarballs

- Create a ./ds-discovery directory on the machine that will run Montecristo. (Make sure there are no spaces in the path above this directory).
- Under that create a directory for your issue #, e.g. DSE-001.
- Place the tarballs collected by ds-collector from each node in that directory:

```
./ds-discovery/
    ./DSE-001/
        ./<node1name>_<timestamp>.tar.gz
        ./<node2name>_<timestamp>.tar.gz
        ./<node3name>_<timestamp>.tar.gz
```

- Under the Issue directory, create a directory named “extracted”.
- Uncompress each tarball and move the uncompressed directory under the “extracted” directory, shortening the directory name to just the node’s name (remove the timestamp), so it looks like this:

```
./ds-discovery/
    ./DSE-001/
        ./extracted/
            ./<node1name>/
            ./<node2name>/
            ./<node3name>/
```

Running Montecristo

- Run it:

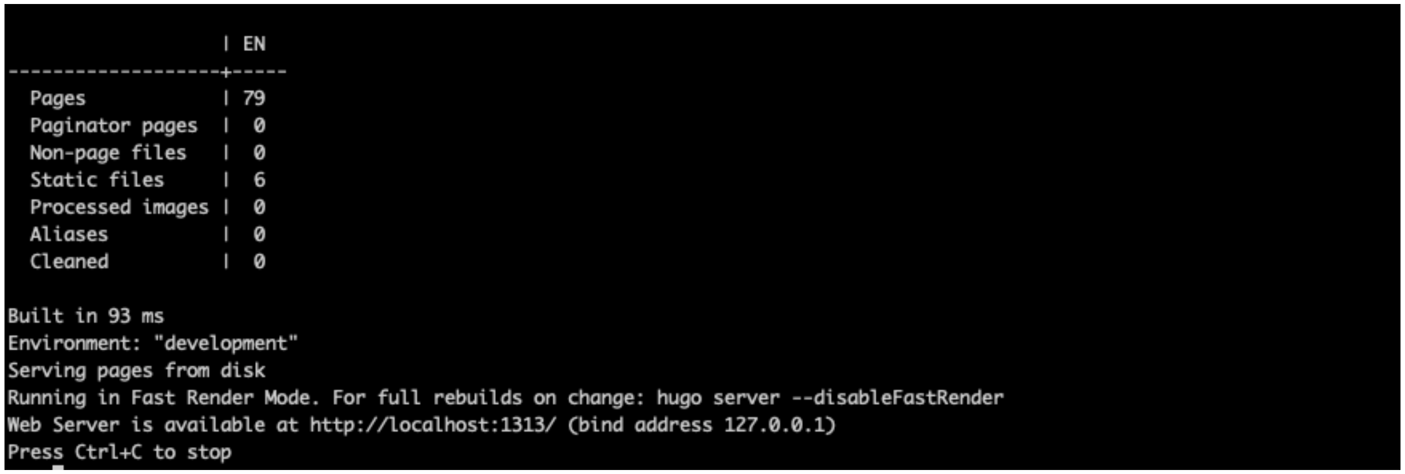
```
cd ./Montecristo
./run.sh -d -c ~/ds-discovery/DSE-001/extracted/ DSE-001
```

  - The last parameter names the artifacts directory
- Answer questions when prompted:
  - Do you want to copy Artifacts? **Y**
  - Decompress again? **N**
  - Run conversion process? **Y**
  - Asks again, run the conversion process? **Y**

- It can take a while, especially with many nodes.

- When finished, Montecristo starts the hugo web server on the machine to make the report accessible.

Access Health Check Report



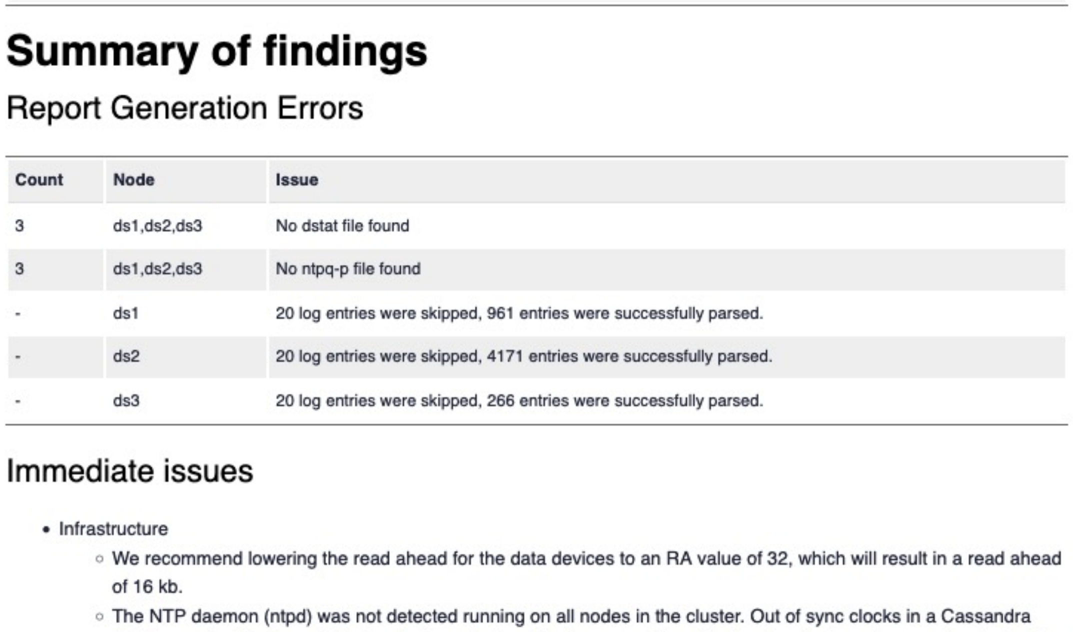
```
                   | EN
-------------------+-----
  Pages            | 79
  Paginator pages  |  0
  Non-page files   |  0
  Static files     |  6
  Processed images |  0
  Aliases          |  0
  Cleaned          |  0

Built in 93 ms
Environment: "development"
Serving pages from disk
Running in Fast Render Mode. For full rebuilds on change: hugo server --disableFastRender
Web Server is available at http://localhost:1313/ (bind address 127.0.0.1)
Press Ctrl+C to stop
```

Web Server is available at http://localhost:1313/
Press Ctrl+C to stop

Can print report as a PDF.

You can convert PDF to Word within MS Office.



Summary of findings

Report Generation Errors

Count	Node	Issue
3	ds1,ds2,ds3	No dstat file found
3	ds1,ds2,ds3	No ntpq-p file found
-	ds1	20 log entries were skipped, 961 entries were successfully parsed.
-	ds2	20 log entries were skipped, 4171 entries were successfully parsed.
-	ds3	20 log entries were skipped, 266 entries were successfully parsed.

Immediate issues

- Infrastructure
  - We recommend lowering the read ahead for the data devices to an RA value of 32, which will result in a read ahead of 16 kb.
  - The NTP daemon (ntpd) was not detected running on all nodes in the cluster. Out of sync clocks in a Cassandra

Installing Montecristo

- System Requirements:
  - Linux or Mac machine
  - Has access to diagnostic tarball directory
  - Cannot be run on a cluster node
- Download site: https://github.com/datastax/diagnostic-collection/releases

Installing Montecristo: Dependencies

Install Dependencies

- Install java 8 JDK with hotspot, hugo and jq (Mac OS X instructions)
- Ensure that zip and unzip are installed (they probably are already)

```
brew tap homebrew/cask-versions
brew install --cask temurin8
brew install hugo
brew install jq
brew install zip
brew install unzip
```

- Make sure Java 8 JDK is the default java:

```
java -version
```

- If not, set JAVA_HOME to point to the correct java, for example (set for your version):

```
/usr/libexec/java_home -V
export JAVA_HOME=`/usr/libexec/java_home -v 1.8.0_412`
```

Installing Montecristo: Gradle fix

- Even with JAVA_HOME set correctly, Gradle had an issue with the Java version on Mac OS X, when running Montecristo.
- To fix, I created the file `~/.gradle/gradle.properties`, containing the line:

```
org.gradle.java.home=/Library/Java/JavaVirtualMachines/temurin-8.jdk/Contents/Home
```

- I then manually built Gradle with debug and stacktrace turned on:

```
cd ~/montecristo/Montecristo/montecristo
./gradlew build install --stacktrace --debug -x test
```

- This fixed the gradle build problem.
- I don't think the setting of JAVA_HOME matters after the gradle.properties file is created.

Installing Montecristo: Make Hugo

Make Hugo

- Run the first time after new install:

```
cd montecristo/src/main/resources/
./mkhugozip.sh
```

Installing Montecristo: DSE jars

To enable the conversion and reading of Datastax Enterprise sstablemetadata files, get the 6.8.17 version of DSE:

```
curl -L https://downloads.datastax.com/enterprise/dse-6.8.17-bin.tar.gz | tar xz
```

Create libs directory under Montecristo's ./dse-stats-converter:

```
mkdir ./montecristo/Montecristo/dse-stats-converter/libs
```

Copy these libs from DSE 6.8 Cassandra/lib to ./dse-stats-converter/libs:

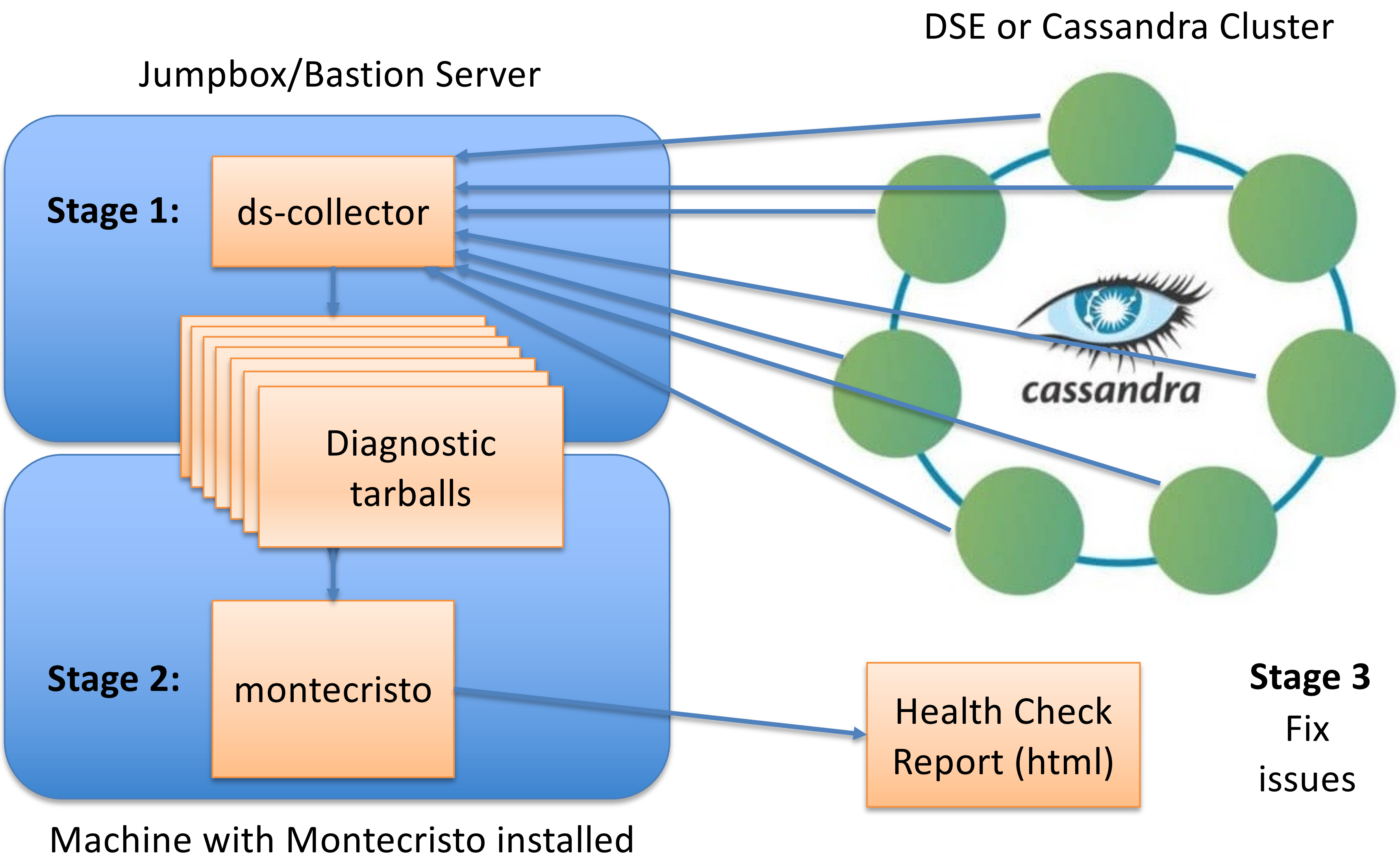
- agrona-0.9.26.jar
- dse-commons-6.8.17.jar
- dse-db-all-6.8.17.jar
- durian-3.4.0.jar
- jctools-core-2.1.2.jar
- netty-all-4.1.25.7.dse.jar
- rxjava-2.2.7.jar

If the dse-db-all.jar file exists in that folder, it will automatically enable the option within the run.sh script to offer the option of converting the files.

Helpful Links

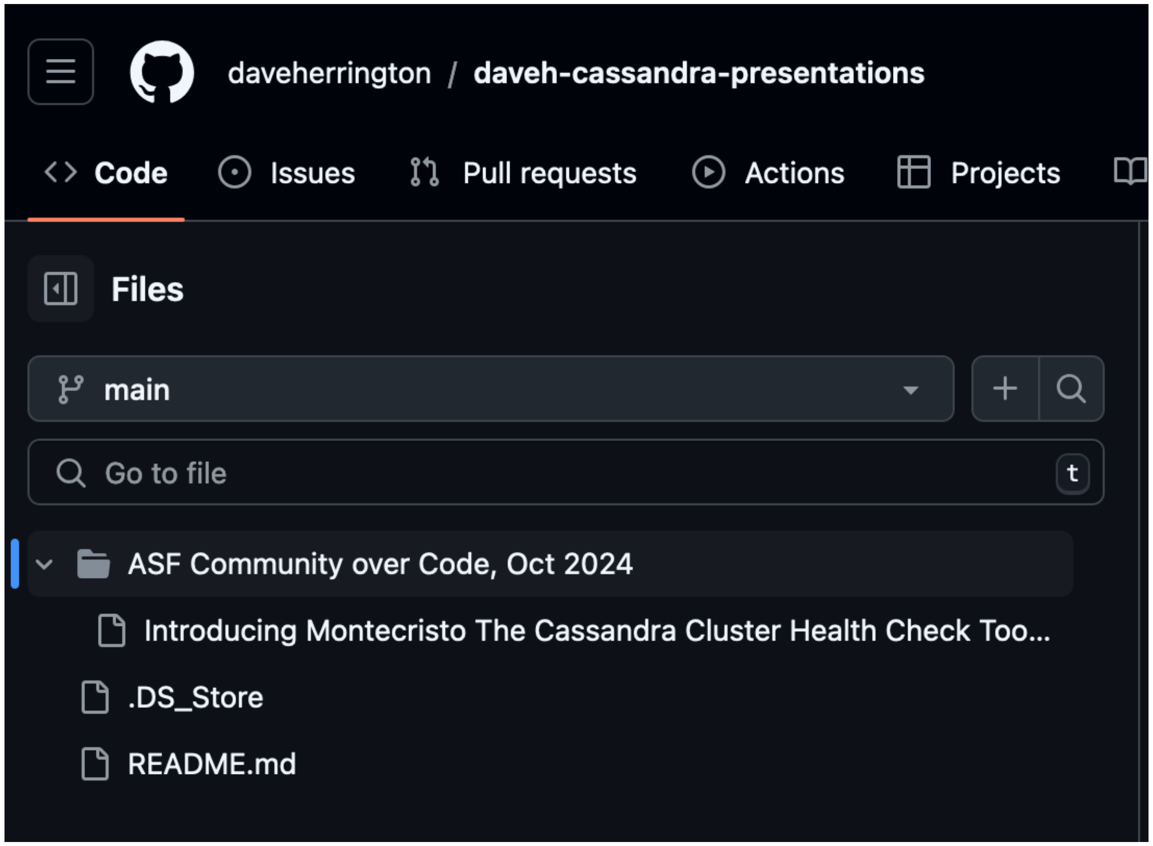
- DataStax Diagnostic Collector (ds-collector):
  - https://github.com/datastax/diagnostic-collection
- Montecristo
  - https://github.com/datastax-labs/Montecristo
- DSE and Apache Cassandra Expertise
  - www.rhinosource.com

Health Check Process: Q&A



Download This Presentation

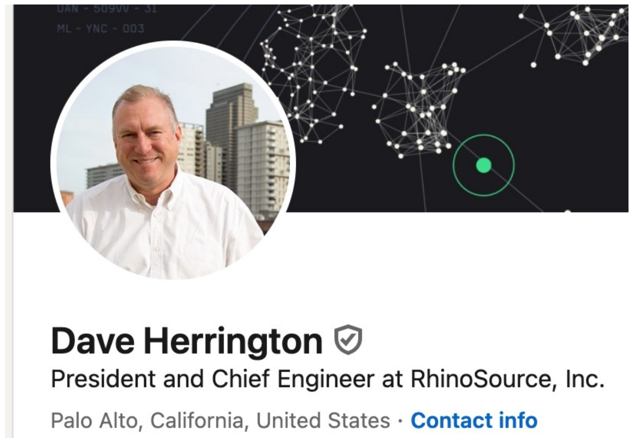
https://github.com/daveherrington/daveh-cassandra-presentations



daveherrington / daveh-cassandra-presentations

ASF Community over Code, Oct 2024
Introducing Montecristo The Cassandra Cluster Health Check Too...
.DS_Store
README.md

Please Contact Me



Dave Herrington
President and Chief Engineer at RhinoSource, Inc.
Palo Alto, California, United States · Contact info

https://www.linkedin.com/in/daveherrington/
GitHub: daveherrington
herry@rhinosource.com
Phone: 650-360-3143

We can run the Montecristo health check for you, if you collect and upload the diagnostic tarballs to our website:
www.rhinosource.com