A New Generation of Data Transfer Tools for Hadoop: Sqoop 2

Bilung Lee (blee at cloudera dot com)

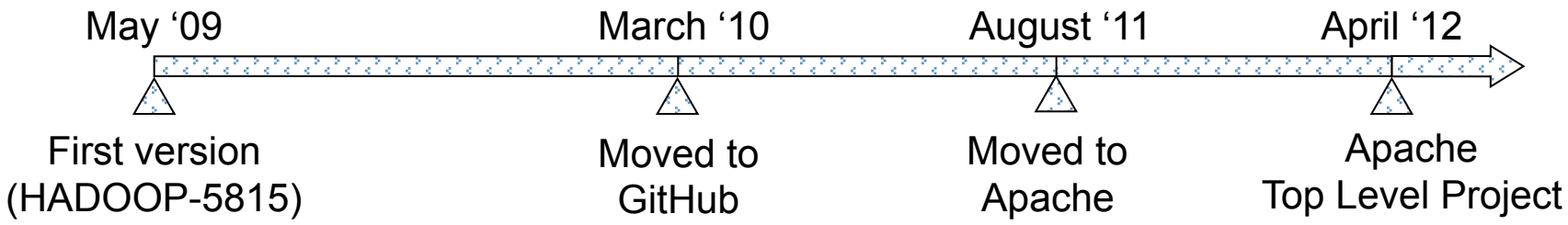
Kathleen Ting (kathleen at cloudera dot com)

Who Are We?

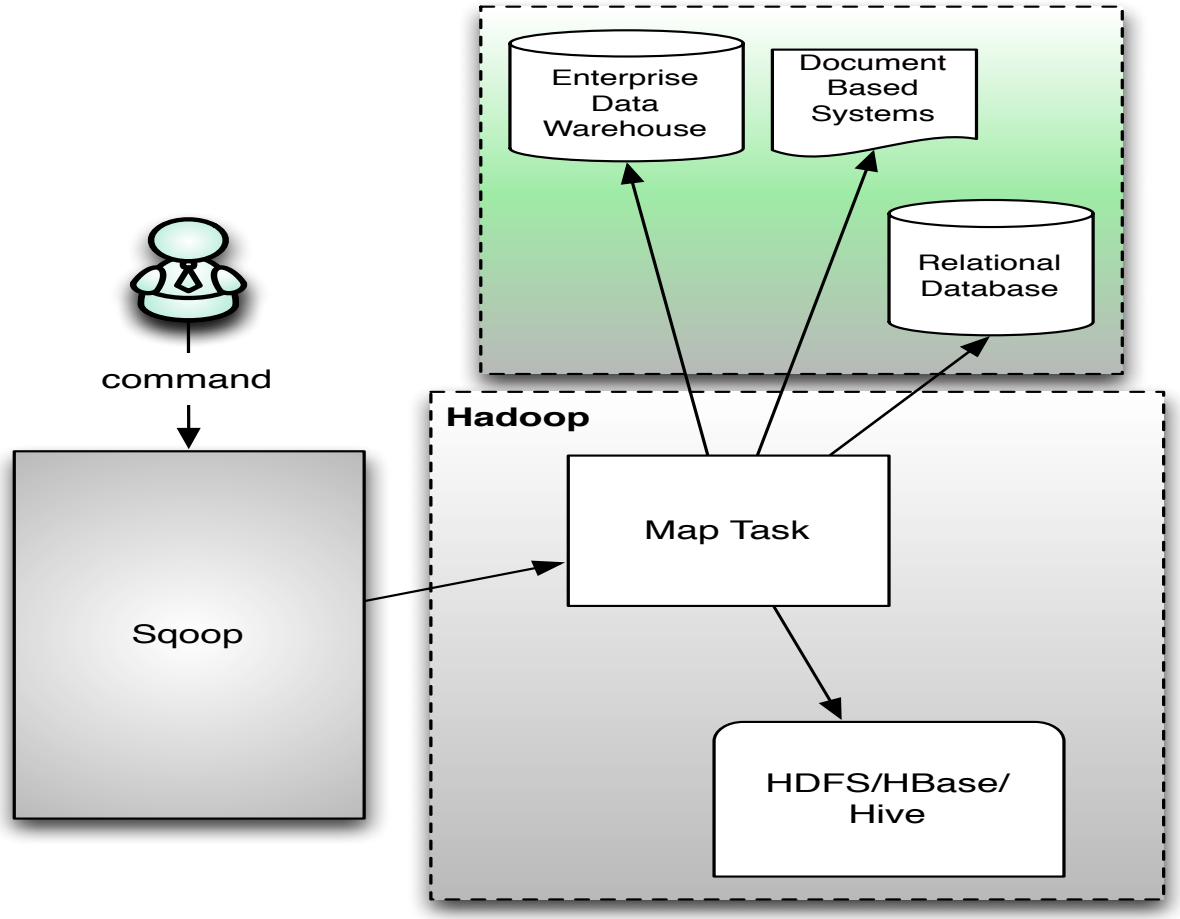
- Bilung Lee
  - Apache Sqoop Committer
  - Software Engineer, Cloudera
- Kathleen Ting
  - Apache Sqoop Committer
  - Support Manager, Cloudera

What is Sqoop?

- Bulk data transfer tool
  - Import/Export from/to relational databases, enterprise data warehouses, and NoSQL systems
  - Populate tables in HDFS, Hive, and HBase
  - Integrate with Oozie as an action
  - Support plugins via connector based architecture



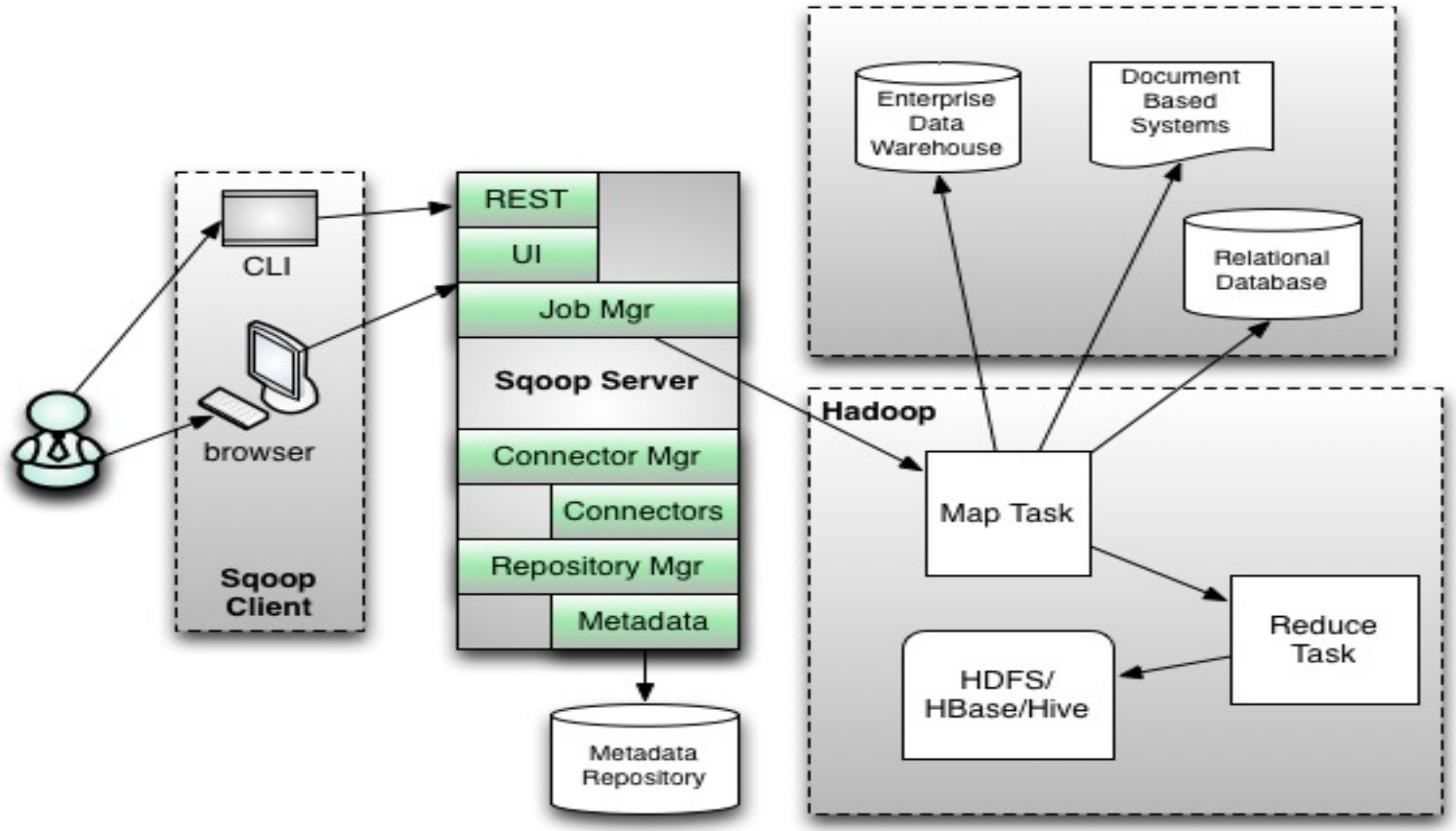
Sqoop 1 Architecture



Sqoop 1 Challenges

- Cryptic, contextual command line arguments
- Tight coupling between data transfer and output format
- Security concerns with openly shared credentials
- Not easy to manage installation/configuration
- Connectors are forced to follow JDBC model

Sqoop 2 Architecture



Sqoop 2 Themes

- Ease of Use
- Ease of Extension
- Security

Sqoop 2 Themes

- Ease of Use
- Ease of Extension
- Security

Ease of Use

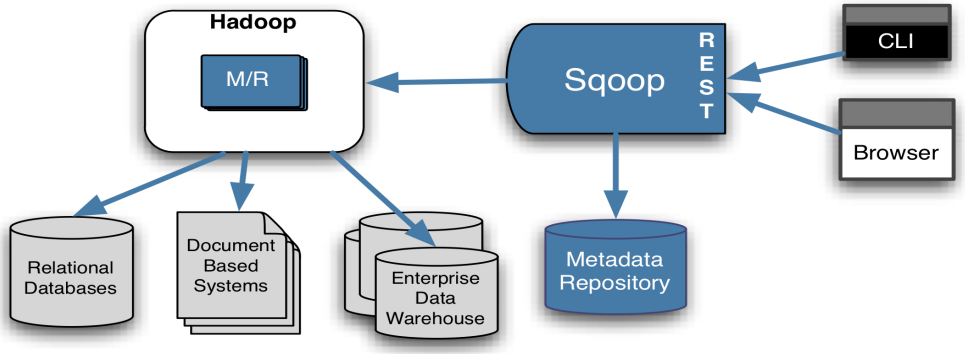
Sqoop 1	Sqoop 2
Client-only Architecture	Client/Server Architecture
CLI based	CLI + Web based
Client access to Hive, HBase	Server access to Hive, HBase
Oozie and Sqoop tightly coupled	Oozie finds REST API

Sqoop 1: Client-side Tool

- Client-side installation + configuration
  - Connectors are installed/configured locally
  - Local requires root privileges
  - JDBC drivers are needed locally
  - Database connectivity is needed locally

Sqoop 2: Sqoop as a Service

- Server-side installation + configuration
  - Connectors are installed/configured in one place
  - Managed by administrator and run by operator
  - JDBC drivers are needed in one place
  - Database connectivity is needed on the server



Client Interface

- Sqoop 1 client interface:
  - Command line interface (CLI) based
  - Can be automated via scripting

- Sqoop 2 client interface:
  - CLI based (in either interactive or script mode)
  - Web based (remotely accessible)
  - REST API is exposed for external tool integration

Sqoop 1: Service Level Integration

- Hive, HBase
  - Require local installation
- Oozie
  - von Neumann(esque) integration:
    - Package Sqoop as an action
    - Then run Sqoop from node machines, causing one MR job to be dependent on another MR job
    - Error-prone, difficult to debug

Sqoop 2: Service Level Integration

- Hive, HBase
  - Server-side integration
- Oozie
  - REST API integration

Ease of Use

Sqoop 1	Sqoop 2
Client-only Architecture	Client/Server Architecture
CLI based	CLI + Web based
Client access to Hive, HBase	Server access to Hive, HBase
Oozie and Sqoop tightly coupled	Oozie finds REST API

Sqoop 2 Themes

- Ease of Use
- Ease of Extension
- Security

Ease of Extension

Sqoop 1	Sqoop 2
Connector forced to follow JDBC model	Connector given free rein
Connectors must implement functionality	Connectors benefit from common framework of functionality
Connector selection is implicit	Connector selection is explicit

Sqoop 1: Implementing Connectors

- Connectors are forced to follow JDBC model
  - Connectors are limited/required to use common JDBC vocabulary (URL, database, table, etc)
- Connectors must implement all Sqoop functionality they want to support
  - New functionality may not be available for previously implemented connectors

Sqoop 2: Implementing Connectors

- Connectors are not restricted to JDBC model
  - Connectors can define own domain
- Common functionality are abstracted out of connectors
  - Connectors are only responsible for data transfer
  - Common Reduce phase implements data transformation and system integration
  - Connectors can benefit from future development of common functionality

Different Options, Different Results

Which is running MySQL?

```
$ sqoop import --connect jdbc:mysql://localhost/db \
--username foo --table TEST
```

```
$ sqoop import --connect jdbc:mysql://localhost/db \
--driver com.mysql.jdbc.Driver --username foo --table TEST
```

- Different options may lead to unpredictable results
  - Sqoop 2 requires explicit selection of a connector, thus disambiguating the process

Sqoop 1: Using Connectors

- Choice of connector is implicit
  - In a simple case, based on the URL in --connect string to access the database
  - Specification of different options can lead to different connector selection
  - Error-prone but good for power users

Sqoop 1: Using Connectors

- Require knowledge of database idiosyncrasies
  - e.g. Couchbase does not need to specify a table name, which is required, causing --table to get overloaded as backfill or dump operation
  - e.g. --null-string representation is not supported by all connectors
- Functionality is limited to what the implicitly chosen connector supports

Sqoop 2: Using Connectors

- Users make explicit connector choice
  - Less error-prone, more predictable
- Users need not be aware of the functionality of all connectors
  - Couchbase users need not care that other connectors use tables

Sqoop 2: Using Connectors

- Common functionality is available to all connectors
  - Connectors need not worry about common downstream functionality, such as transformation into various formats and integration with other systems

Ease of Extension

Sqoop 1	Sqoop 2
Connector forced to follow JDBC model	Connector given free rein
Connectors must implement functionality	Connectors benefit from common framework of functionality
Connector selection is implicit	Connector selection is explicit

Sqoop 2 Themes

- Ease of Use
- Ease of Extension
- Security

Security

Sqoop 1	Sqoop 2
Support only for Hadoop security	Support for Hadoop security and role-based access control to external systems
High risk of abusing access to external systems	Reduced risk of abusing access to external systems
No resource management policy	Resource management policy

Sqoop 1: Security

- Inherit/Propagate Kerberos principal for the jobs it launches
- Access to files on HDFS can be controlled via HDFS security
- Limited support (user/password) for secure access to external systems

Sqoop 2: Security

- Inherit/Propagate Kerberos principal for the jobs it launches
- Access to files on HDFS can be controlled via HDFS security
- Support for secure access to external systems via role-based access to connection objects
  - Administrators create/edit/delete connections
  - Operators use connections

Sqoop 1: External System Access

- Every invocation requires necessary credentials to access external systems (e.g. relational database)
  - Workaround: create a user with limited access in lieu of giving out password
    - Does not scale
    - Permission granularity is hard to obtain
- Hard to prevent misuse once credentials are given

Sqoop 2: External System Access

- Connections are enabled as first-class objects
  - Connections encompass credentials
  - Connections are created once and then used many times for various import/export jobs
  - Connections are created by administrator and used by operator
    - Safeguard credential access from end users
- Connections can be restricted in scope based on operation (import/export)
  - Operators cannot abuse credentials

Sqoop 1: Resource Management

- No explicit resource management policy
  - Users specify the number of map jobs to run
  - Cannot throttle load on external systems

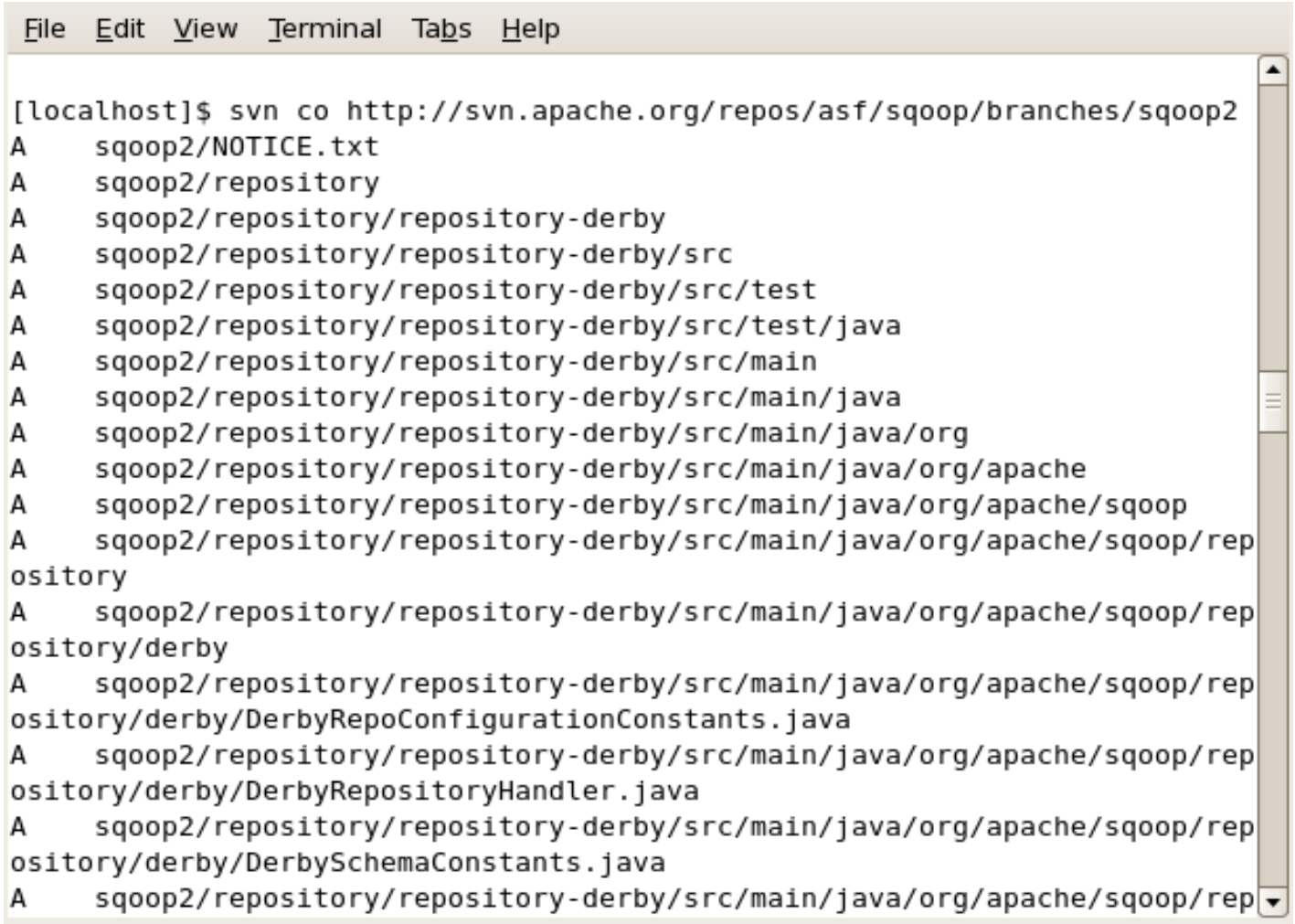
Sqoop 2: Resource Management

- Connections allow specification of resource management policy
  - Administrators can limit the total number of physical connections open at one time
  - Connections can also be disabled

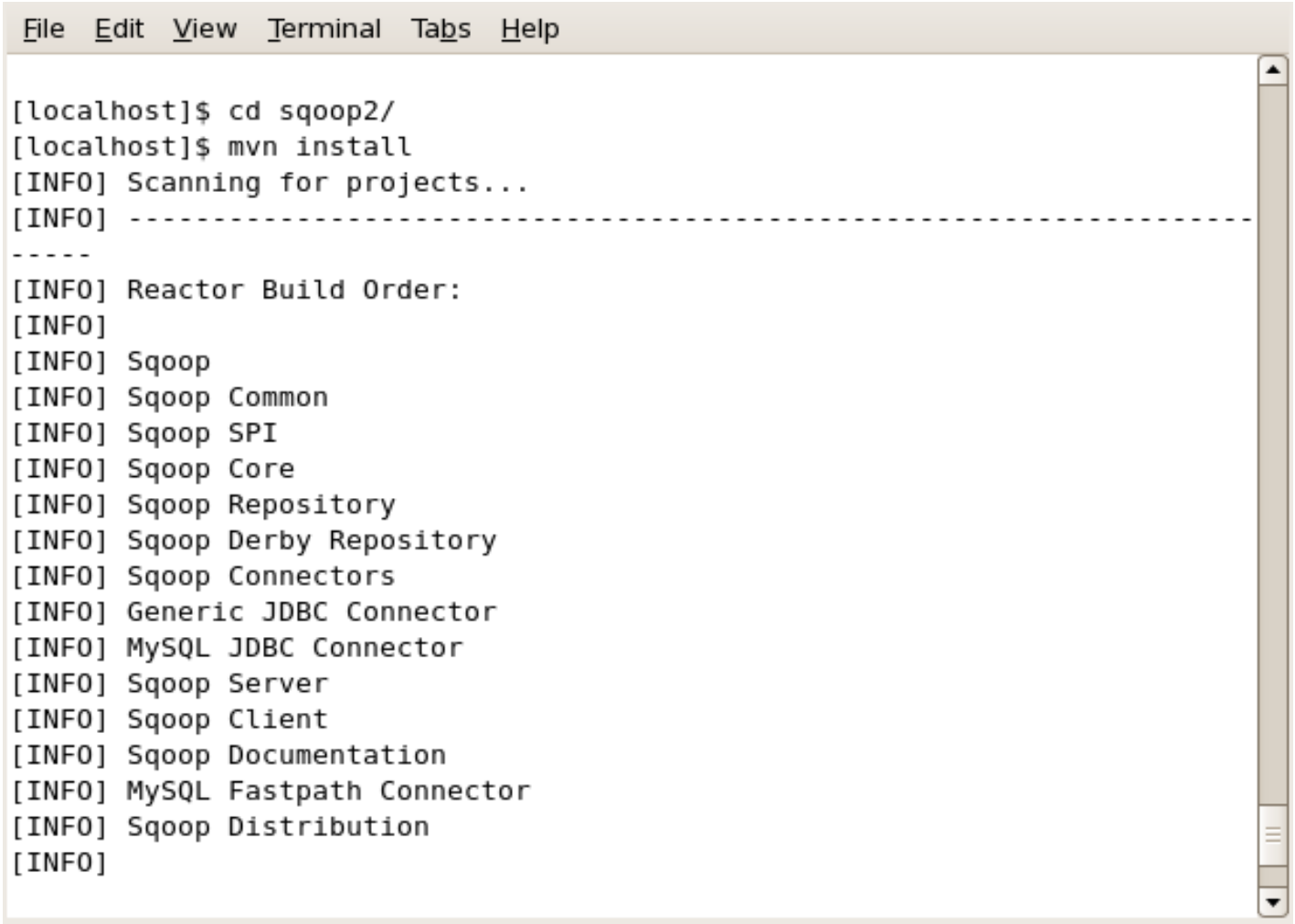
Security

Sqoop 1	Sqoop 2
Support only for Hadoop security	Support for Hadoop security and role-based access control to external systems
High risk of abusing access to external systems	Reduced risk of abusing access to external systems
No resource management policy	Resource management policy

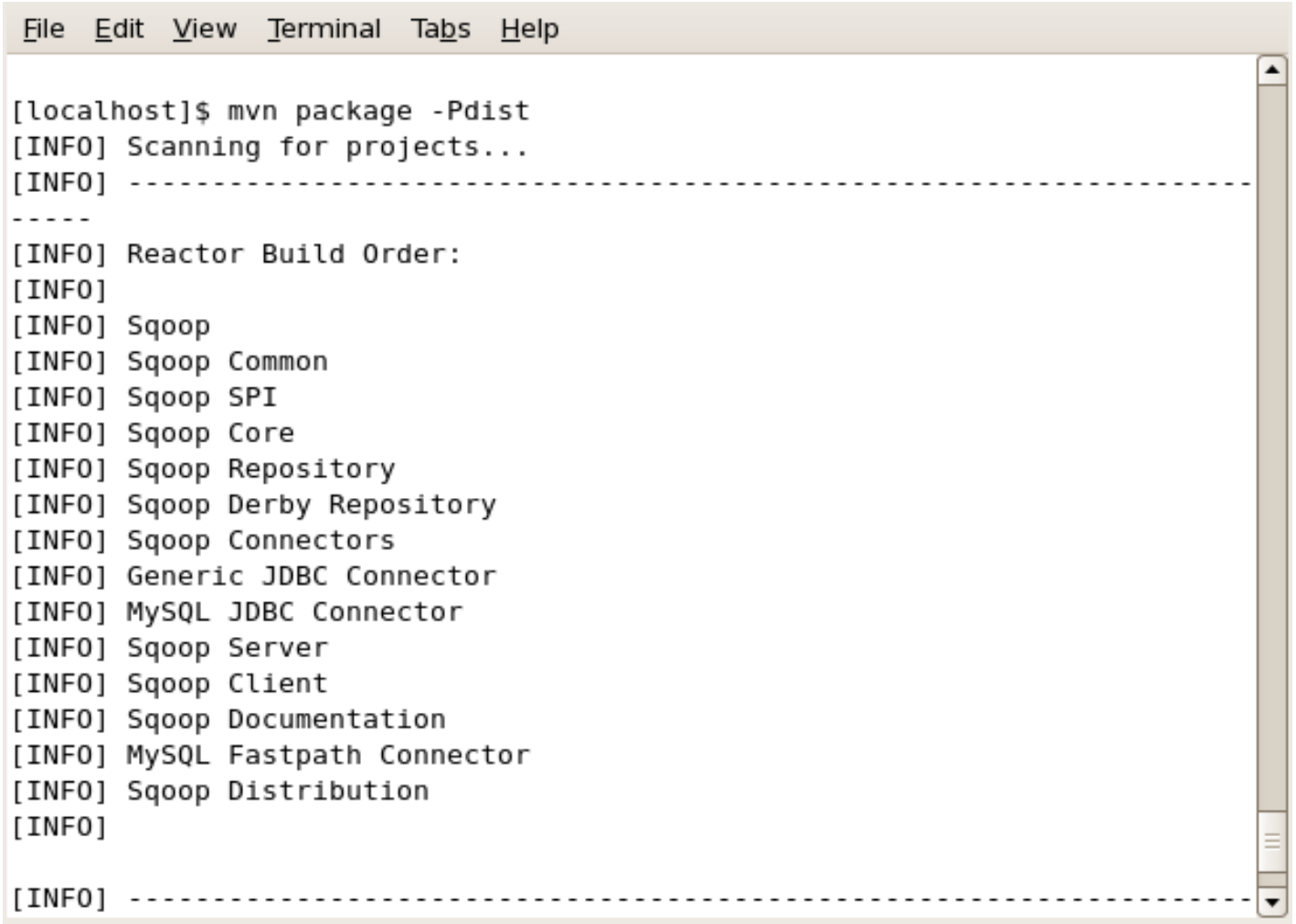
Demo Screenshots



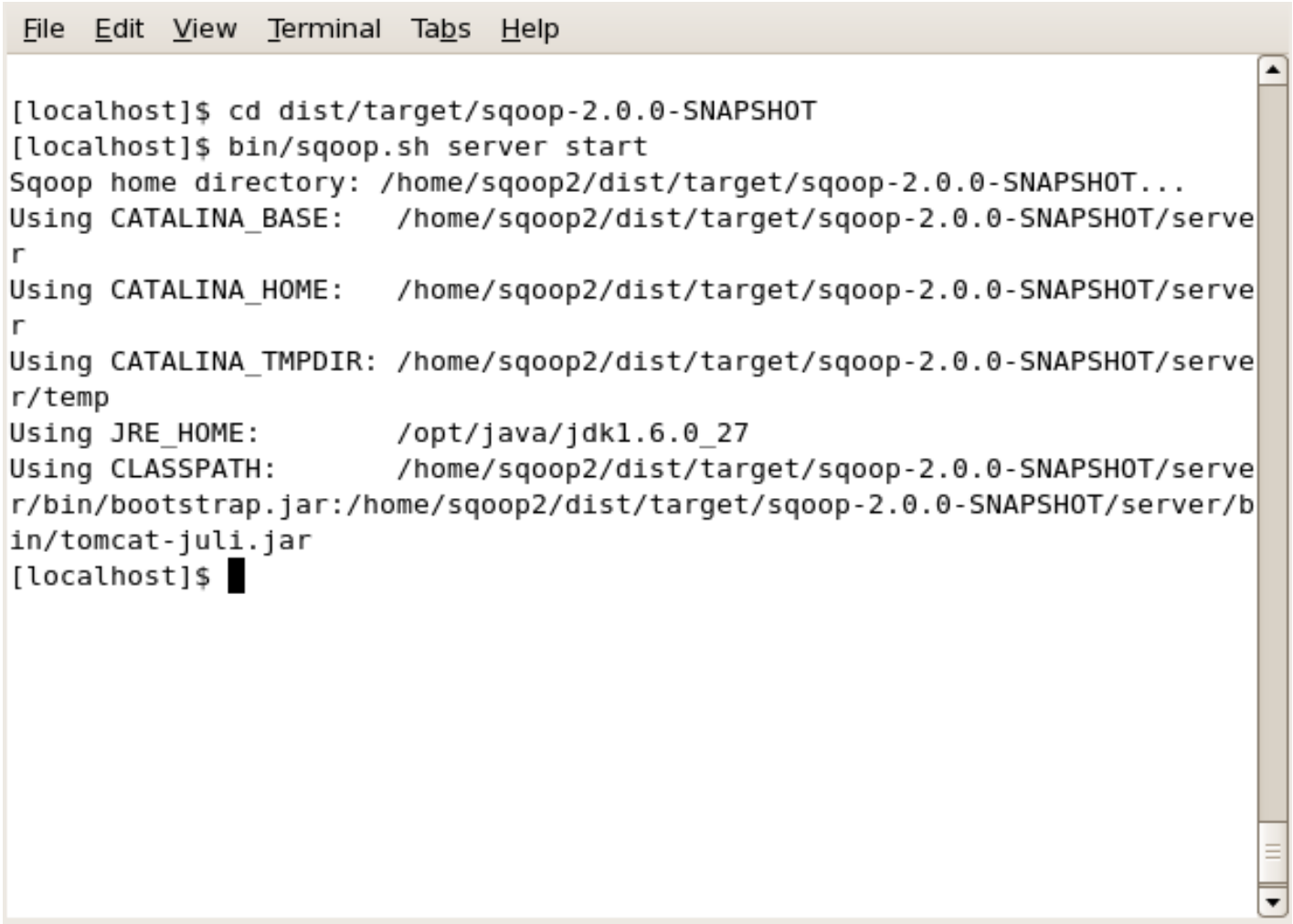
Demo Screenshots



Demo Screenshots



Demo Screenshots



Demo Screenshots

File Edit View Terminal Tabs Help

```
[localhost]$ bin/sqoop.sh client
Sqoop home directory: /home/sqoop2/dist/target/sqoop-2.0.0-SNAPSHOT...
Jun 8, 2012 10:42:22 PM java.util.prefs.FileSystemPreferences$2 run
INFO: Created user preferences directory.
Sqoop Shell: Type 'help' or '\h' for help.

sqoop:000> show version
Usage: show version
-a,--all         Display all versions
-c,--client      Display client version
-p,--protocol    Display protocol version
-s,--server      Display server version

sqoop:000> show version --all
Server version:
  Sqoop 2.0.0-SNAPSHOT revision 1346742
  Compiled by root on Fri Jun  8 22:38:45 PDT 2012
Client version:
  Sqoop 2.0.0-SNAPSHOT revision 1346742
  Compiled by root on Fri Jun  8 22:38:45 PDT 2012
Protocol version:
  [1]
```

Takeaway

Sqoop 2 Highights:

- Ease of Use: Sqoop as a Service
- Ease of Extension: Connectors benefit from shared functionality
- Security: Connections as first-class objects and role-based security

Current Status: work-in-progress

- Sqoop2 Development:
  http://issues.apache.org/jira/browse/SQOOP-365
- Sqoop2 Blog Post:
  http://blogs.apache.org/sqoop/entry/apache_sqoop_highlights_of_sqoop
- Sqoop2 Design:
  http://cwiki.apache.org/confluence/display/SQOOP/Sqoop+2

Current Status: work-in-progress

- Sqoop2 Quickstart:
  http://cwiki.apache.org/confluence/display/SQOOP/Sqoop2+Quickstart
- Sqoop2 Resource Layout:
  http://cwiki.apache.org/confluence/display/SQOOP/Sqoop2+-+Resource+Layout
- Sqoop2 Feature Requests:
  http://cwiki.apache.org/confluence/display/SQOOP/Sqoop2+Feature+Requests

SQOOP WANT YOU