A Practitioner’s Guide to Securing Your Hadoop Cluster
Your Speakers

- Mike Yoder
  - Software Engineer, Cloudera

- André Araújo
  - Solutions Architect, Cloudera

- Manish Ahluwalia
  - Software Engineer, Cloudera

- Mark Donsky
  - Product Management, Cloudera
Format

- Five sections
- Each section:
  - Introduce a security concept
  - Demo without it in place
  - How to enable
  - Demo with it in place
- Please hold questions until the end of each section
- Short break in the middle

- Slides are available at http://tinyurl.cloudera.com/shw-sjc-2017-slides
Agenda

▪ Prelude: Network Security – Mike
▪ Authentication – Mike
▪ Authorization – Mike
▪ Wire Encryption – Andre
▪ Encryption-at-rest – Manish
▪ Data Governance – Mark
▪ Final Thoughts – Mike
Prelude: Network Security
Don’t Put Your Hadoop Cluster on the Open Internet

- MongoDB ransomware
  - Tens of thousands of open MongoDB instances on the internet
  - With no security turned on
  - The attack: All data deleted or encrypted; ransom note left behind
- It has happened to Hadoop clusters, too
Basic Networking Checks

- Make sure your IP address isn’t an internet-exposed address
  - These are the private IP address ranges:
    - 10.* (10.0/8)
    - 172.16.* - 172.31.* (172.16/12)
    - 192.168.* (192.168/16)
- Use `nmap` from outside your corporate environment
- If in {AWS, Azure, GCE}, check networking configuration
Questions?
Authentication

Mike Yoder

Software Engineer
Cloudera
Authentication - Agenda

- Intro - identity and authentication
- **DEMO:** Hadoop with no authentication
- Kerberos and LDAP authentication
- Enabling kerberos and LDAP using Cloudera Manager
- **DEMO:** Actual strong authentication in Hadoop
- Questions
Identity

- Before we can talk about authentication, we must understand **identity**
- An object that uniquely identifies a user (usually)
  - Email account, Windows account, passport, driver’s license

- In Hadoop, identity largely means **username**
- Using a common source of identity is paramount
Identity Sources

- Individual Linux servers use `/etc/passwd` and `/etc/group` - Not scalable and prone to errors
- LDAP is the preferred way - Integrate at the Linux OS level   - RedHat SSSD   - Centrify - **All** applications running on the OS can use the same LDAP integration - Most enterprises use Active Directory - Some enterprises use a Linux-specific LDAP implementation
Identity and Authentication

- So you have an identity database, now what?
- Users and applications must **prove** their identities to each other
- This process is authentication
- Hadoop strong authentication is built around **Kerberos**
- Kerberos is built into Active Directory and this is the most common Hadoop integration
Hadoop Default “Authentication”

- Out of the box, Hadoop “authenticates” users by simply believing whatever username you tell it you are
- This includes telling Hadoop you are the hdfs user, a superuser!

- DEMO: Let’s see just how bad this is.
Kerberos

- To enable security in Hadoop, everything starts with Kerberos
- Every role type of every service has its own unique Kerberos credentials
- Users must prove their identity by obtaining a Kerberos ticket, which is honored by the Hadoop components
- Hadoop components themselves authenticate to each other for intra and inter service communication
Kerberos Authentication
LDAP and SAML

- Beyond just Kerberos, other components such as web consoles and JDBC/ODBC endpoints can authenticate users differently.
- **LDAP** authentication is supported for Hive, Impala, Solr, and web-based UIs.
- **SAML** (SSO) authentication is supported for Cloudera Manager, Navigator, and Hue.
- Some components support both Kerberos and LDAP authentication at the same time.
- Generally speaking, LDAP is a much easier authentication mechanism to use for external applications – No Kerberos software and configuration required!
- …just make sure wire encryption is also enabled to protect passwords.
Web UI LDAP Authentication

Users → Login Requests → Cloudera Manager → LDAP Authentication → Microsoft Active Directory

Cloudera Navigator → LDAP Authentication → Cloudera Hue
Impala Dual-mode Authentication
Enabling Kerberos

- Setting up Kerberos for your cluster is no longer a daunting task
- Cloudera Manager and Apache Ambari provide wizards to automate the provisioning of service accounts and the associated keytabs
- Both MIT Kerberos and Active Directory are supported Kerberos KDC types
- Again, most enterprises use Active Directory so let’s see what we need to set it up!
Active Directory Prerequisites

- At least one AD domain controller is setup with LDAPS
- An AD account for Cloudera Manager
- A dedicated OU in your desired AD domain
- An account that has create/modify/delete user privileges on this OU
- This is not a domain admin / administrative account!
- While not required, AD group policies can be used to further restrict the accounts
- Install openldap-clients on the CM server host, krb5-workstation on every host
- From here, use the wizard!
Cloudera Manager Kerberos Wizard

Before using the wizard, please ensure that you have performed the following steps:

- Set up a working KDC. Cloudera Manager supports MIT KDC and Active Directory.
  - Yes, I've set up a working KDC.

- The KDC should be configured to have non-zero ticket lifetime and renewal lifetime. CDH will not work properly if tickets are not renewable.
  - Yes, I've checked that the KDC allows renewable tickets.

- OpenLdap client libraries should be installed on the Cloudera Manager Server host if you want to use Active Directory. Also, Kerberos client libraries should be installed on ALL hosts.
  - Yes, I've installed the client libraries.

- Cloudera Manager needs an account that has permissions to create other accounts in the KDC.
  - Yes, I've created a proper account for Cloudera Manager.
### KDC Information

Specify information about the KDC. The properties below are used by Cloudera Manager to generate principals for CDH daemons running on the cluster.

**KDC Type**
- MIT KDC
- Active Directory

**KDC Server Host**
- ad.hadoop.com

**Kerberos Security Realm**
- HADOOP.COM

**Kerberos Encryption Types**
- aes256-cts
- aes128-cts
- rc4-hmac

**Active Directory Suffix**
- `ou=hadoop,DC=hadoop,DC=com`

**Active Directory Account Prefix**
- cdn_

**Active Directory Domain Controller Override**
- `my-ad-dc1.hadoop.com`
Cloudera Manager Kerberos Wizard

KDC Account Manager Credentials

Enter the credentials for the account that has permissions to create other users. Cloudera Manager will store it in encrypted form and use it whenever new principals need to be generated.

Username: cloudera_manager @ HADOOP.COM
Password: ********

Click through the remaining steps
Setting up LDAP Authentication

- CM -> Administration -> Settings
  - Click on category “External Authentication”
- Cloudera Management Services -> Configuration
  - Click on category “External Authentication”
- Hue / Impala / Hive / Solr -> Configuration
  - Search for “LDAP”
Post-Configuration

- Kerberos authentication is setup
- LDAP authentication is setup

DEMO: No more fake authentication!
Questions?
Authorization

Mike Yoder
Software Engineer
Cloudera
Authorization - Agenda

- Authorization – Overview
- DEMO: Default Authorization
- Configuration Stronger Authorization
- Apache Sentry
- Record Service
- DEMO: Strong Authorization
- Questions
Authorization - Overview

- Authorization dictates what a user is permitted to do
- Happens after a user has authenticated to establish identity
- Authorization policies in Hadoop are typically based on:
  - Who the user is and what groups they belong to
  - Role-based access control (RBAC)
- Many different authorization mechanisms in Hadoop components
Authorization in Hadoop

- HDFS file permissions (POSIX ‘rwx rwx rwx’ style)
- Yarn job queue permissions
- Sentry (Hive / Impala / Solr / Kafka)
- Cloudera Manager RBAC
- Cloudera Navigator RBAC
- Hue groups
- Hadoop KMS ACLs
- HBase ACLs
- etc.
Default Authorization Examples

- **HDFS**
  - Default umask is 022, making all new files *world readable*
  - Any authenticated user can execute hadoop shell commands

- **YARN**
  - Any authenticated user can submit and *kill jobs* for any queue

- **Hive metastore**
  - Any authenticated user can *modify the metastore* (CREATE/DROP/ALTER/etc.)

- **DEMO:** Let’s see just how bad this is.
# Configuring HDFS Authorization

- Set default umask to 026
- Setup hadoop-policy.xml (Service Level Authorization)

<table>
<thead>
<tr>
<th>Default Umask</th>
<th>HDFS-1 (Service-Wide)</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>dfs.umaskmode, fs.permissions.umask-mode</td>
<td>026</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Authorized Groups</th>
<th>HDFS-1 (Service-Wide)</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>prod_cdh_users</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Authorized Admin Groups</th>
<th>HDFS-1 (Service-Wide)</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>prod_cdh_admins</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Configuring Yarn Authorization

- Setup the YARN admin ACL
Apache Sentry

- Provides centralized RBAC for several components
  - **Hive / Impala**: Databases, tables, views, columns
  - **Solr**: Collections, documents, indexes
  - **Kafka**: Cluster, topic, consumer group
Configuring Sentry

- Cloudera Manager -> Add Service -> Sentry
- Hive
  - Set Sentry service
  - Disable HiveServer2 impersonation
- Impala
  - Set Sentry Service
- HDFS
  - Enable Sentry HDFS Synchronization
  - Enable extended ACLs
  - Specify path prefixes
Record Service

- Currently in public beta
- Layer that enforces fine-grained Sentry permissions
  - Without RS, file based access is all-or-nothing
  - View / Column permissions for Spark and MapReduce
- Opens up possibilities for dynamic data masking and tokenization
- Can swap out entire storage layer.

Diagram:
- Execution Layer (Spark, MapReduce, etc.)
- Record Service
- Storage Layer (HDFS/S3/etc.)
- Sentry
Post Configuration

- HDFS setup with a better umask and service level authorization
- YARN setup with restrictive admin ACLs
- Hive, Impala, and HDFS setup with Sentry integration

- **DEMO**: No more default authorization holes!
Authorization - Summary

- HDFS file permissions (POSIX ‘rwx rwx rwx’ style)
- Yarn job queue permissions
- Sentry (Hive / Impala / Solr / Kafka)
- Cloudera Manager RBAC
- Cloudera Navigator RBAC
- Hue groups
- Hadoop KMS ACLs
- HBase ACLs
- etc.
Questions
Encryption of Data in Transit

André Araújo

Solutions Architect
Cloudera
Agenda

▪ Why encryption of data on the wire is important
▪ Technologies used in Hadoop
  - SASL “Privacy”
  - TLS
▪ For each:
  - Demo without
  - Discussion
  - Enabling in Cloudera Manager
  - Demo with it enabled
Why Encrypt Data in Transit?

- Networking configuration (firewalls) can mitigate some risk
- Attackers may already be inside your network
- Data and credentials (usernames and passwords) have to go into and out of the cluster
- Regulations around transmitting sensitive information
- Let’s see this for real using wireshark
Demo

- Transfer data into a cluster
- Simple file transfer: “hadoop fs –put”
- Attacker sees file contents go over the wire
Two Encryption Technologies

▪ SASL “confidentiality” or “privacy” mode
  - Protects core hadoop

▪ TLS – Transport Layer Security
  - Used for “everything else”
SASL

- Simple Authentication and Security Layer
- Not a protocol, but a framework for passing authentication steps between a client and server
- Pluggable with different authentication types
  - GSS-API for Kerberos (Generic Security Services)
- Can provide transport security
  - “auth-int” – integrity protection: signed message digests
  - “auth-conf” – confidentiality: encryption
SASL Encryption - Setup

- First, enable Kerberos
- HDFS:
  - Hadoop RPC Protection
  - Datanode Data Transfer Protection
  - Enable Data Transfer Encryption
  - Data Transfer Encryption Algorithm
  - Data Transfer Cipher Suite Key Strength
SASL Encryption - Setup

- Hbase
  - HBase Thrift Authentication
  - Hbase Transport Security
Demo 2

- Put a file into HDFS again
- But this time with SASL encryption turned on
TLS

- Transport Layer Security
  - The successor to SSL – Secure Sockets Layer
  - The term SSL was deprecated 15 years ago, but we still use it
  - TLS is what’s behind https:// web pages

Let’s what happens with no TLS (an http connection to Hue)
TLS - Certificates

- TLS relies on certificates for authentication
- You’ll need one certificate per machine
- Certificates:
  - Cryptographically prove that you are who you say you are
  - Are issued by a “Certificate Authority” (CA)
  - Have a “subject”, an “issuer” and a “validity period”
  - Many other attributes, like “Extended Key Usage”
  - Let’s look at an https site
TLS – Certificate Authorities

▪ “Homemade” CA using openssl
  - Suitable for test/dev clusters only

▪ Internal Certificate Authority
  - A CA that is trusted widely inside your organization, but not outside
  - Commonly created with Active Directory Certificate Services
  - Web browsers need to trust it as well

▪ External Certificate Authority
  - A widely known CA like VeriSign, GeoTrust, Symantec, etc
  - Costs $$$ per certificate
You

Certificate Authority

<table>
<thead>
<tr>
<th>CSR</th>
<th>Subject</th>
<th>Public Key</th>
<th>Private Key</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Certificate</th>
<th>Subject</th>
<th>Public Key</th>
<th>Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root</td>
<td>Valid Dates</td>
<td>Issuer</td>
<td>Subject</td>
</tr>
<tr>
<td>Intermediate</td>
<td>Public Key</td>
<td>Signature</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Subject</th>
<th>Valid Dates</th>
<th>Issuer</th>
<th>Subject</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intermediate</td>
<td>Public Key</td>
<td>Signature</td>
<td></td>
</tr>
<tr>
<td>Root</td>
<td>Signature</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
TLS – Certificate File Formats

- Two different formats for storing certificates and keys
  - PEM
    - “Privacy Enhanced Mail” (yes, really)
    - Used by openssl; programs written in python and C++
  - JKS
    - Java KeyStore
    - Used by programs written in Java
- The Hadoop ecosystem uses both
- Therefore you must translate private keys and certificates into both formats
TLS – Key Stores and Trust Stores

▪ Keystore
  - Used by the server side of a TLS client-server connection
  - JKS: Contains private keys and the host’s certificate; Password protected
  - PEM: typically one certificate file and one password-protected private key file

▪ Truststore
  - Used by the client side of a TLS client-server connection
  - Contains certificates that the client trusts: the Certificate Authorities
  - JKS: Password protected, but only for an integrity check
  - PEM: Same concept, but no password
  - There is a system-wide certificate store for both PEM and JKS formats.
**TLS – Key Stores and Trust Stores**

1. openssl rsa -in `hostname` -f.key.temp -out `hostname` -f.key
2. cat server.pem int-cA.pem > `hostname` -f.pem
3. openssl pkcs12 -export -in `hostname` -f.pem -inkey `hostname` -f.key -out `hostname` -f.pfx
4. keytool -importkeystore -srstoretype PKCS12 -srkeystore `hostname` -f.pfx -srcstoretype keystore -srcstore `hostname` -f.jks
5. keytool -importcert -file root-cA.pem -alias root-CAG -keystore truststore.jks
6. cp $JAVA_HOME/jre/lib/security/cacerts $JAVA_HOME/jre/lib/security/jssecacerts
TLS – Securing Cloudera Manager

▪ CM Web UI - [https://]

▪ CM Agent -> CM Server communication – 3 “Levels” of TLS use
  - Level 1: Encrypted but no certificate verification. Akin to clicking on [https://]
  - Level 2: Agent verifies the server’s certificate
  - Level 3: Agent and Server verify each other’s certificate. This is called TLS mutual authentication: each side is confident that it’s talking to the other
    - Note: TLS level 3 requires that certificates are suitable for both “TLS Web Server Authentication” and “TLS Web Client Authentication”
  - Very Sensitive Information goes over this channel
  - Like Kerberos Keytabs. Therefore, set up TLS in CM first before Kerberos
<table>
<thead>
<tr>
<th>Configuration</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use TLS Encryption for Admin Console</td>
<td>Enabled</td>
</tr>
<tr>
<td>Use TLS Encryption for Agents</td>
<td>Enabled</td>
</tr>
<tr>
<td>Use TLS Authentication of Agents to Server</td>
<td>Enabled</td>
</tr>
<tr>
<td>Cloudera Manager TLS/SSL Server JKS Keystore File Location</td>
<td>/opt/cloudera/security/jks/keystore.jks</td>
</tr>
<tr>
<td>Cloudera Manager TLS/SSL Server JKS Keystore File Password</td>
<td>Password</td>
</tr>
<tr>
<td>Cloudera Manager TLS/SSL Certificate Trust Store File</td>
<td>/opt/cloudera/security/jks/truststore.jks</td>
</tr>
<tr>
<td>Cloudera Manager TLS/SSL Certificate Trust Store Password</td>
<td>Password</td>
</tr>
</tbody>
</table>

**TLS Level 1**

**TLS Level 3**

**CM Web UI**

---

#StrataHadoop
The CM Agent Settings

- Agent /etc/cloudera-scm-agent/config.ini

```ini
use_tls=1  ← TLS Level 1
verify_cert_file= full path to CA certificate.pem file ← TLS Level 2
client_key_file= full path to private key.pem file
client_keypw_file= full path to file containing password for key
client_cert_file= full path to certificate.pem file
```

TLS Level 3
TLS for CM-Managed Services

- CM requires that all files (jks and pem) are in the same location on each machine
- For each service (HDFS, Hue, Hbase, Hive, Impala, …)
  - Search the configuration for “TLS”
  - Check the “enable” boxes
  - Provide keystore, truststore, and passwords
Hive Example

```plaintext
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable TLS/SSL for HiveServer2</td>
<td>HIVE-1 (Service-Wide)</td>
</tr>
<tr>
<td>hive.server2.enable.SSL</td>
<td></td>
</tr>
<tr>
<td>hive.server2.use.SSL</td>
<td></td>
</tr>
<tr>
<td>HiveServer2 TLS/SSL Server JKS Keystore File Location</td>
<td>/opt/cloudera/security/jks/keystore.jks</td>
</tr>
<tr>
<td>hive.server2.keystore.path</td>
<td></td>
</tr>
<tr>
<td>HiveServer2 TLS/SSL Server JKS Keystore File Password</td>
<td>......</td>
</tr>
<tr>
<td>hive.server2.keystore.password</td>
<td></td>
</tr>
<tr>
<td>HiveServer2 TLS/SSL Certificate Trust Store File</td>
<td>/opt/cloudera/security/jks/truststore.jks</td>
</tr>
<tr>
<td>HiveServer2 TLS/SSL Certificate Trust Store Password</td>
<td>......</td>
</tr>
</tbody>
</table>
```
TLS - Troubleshooting

▪ To examine certificates
  - `openssl x509 -in <cert>.pem -noout -text`
  - `keytool -list -v -keystore <keystore>.jks`

▪ To attempt a TLS connection as a client
  - `openssl s_client -connect <host>:<port>`
    - This tells you all sorts of interesting TLS things
Let's try to attack an https connection to Hue.

Note that this is only one example, TLS protects many, many things in hadoop.
Conclusions

▪ You need to encrypt information on the wire
▪ Technologies used are SASL encryption and TLS
▪ TLS requires certificate setup
Questions?
HDFS Encryption at Rest

Manish Ahluwalia

Software Engineer
Cloudera
Agenda

- Why Encrypt Data
- Demo
- HDFS Encryption
- Demo
- Questions
Why store encrypted data?

- Customers often are mandated to protect data at rest
  - PCI
  - HIPAA
  - National Security
  - Company confidential
- Encryption of data at rest helps mitigate certain security threats
  - Rogue administrators (insider threat)
  - Compromised accounts (masquerade attacks)
  - Lost/stolen hard drives
Demo

- How to access HDFS data from Linux storage bypassing HDFS authorization
Options for encrypting data

- Disk/Block
- File System
- Database
- Application

Level of effort vs. Security
Architectural Concepts

- Encryption Zones
- Keys
- Key Management Server
Encryption Zones

- An HDFS directory in which the contents (including subdirs) are encrypted on write and decrypted on read.
- An EZ begins life as an empty directory
- Rename/Move in/out of an EZ are prohibited
- Encryption is transparent to application with no code changes
EZ Keys, Data Encryption Keys, and Encrypted Data Encryption Keys

1. EZ key + DEK → EDEK
2. EDEK + EZ key → DEK
3. DEK + File → Encrypted File
4. Encrypted File + DEK → File
Key Handling

1. startFile request

2. New EDEK retrieved from cache

3. EDEK persisted to the file’s metadata

4. EDEK returned to client

5. Client calls KMS to decrypt EDEK into DEK

6. Client uses DEK to write encrypted data to HDFS
Key Management Server (KMS)

- KMS sits between client and key server
  - E.g. Cloudera Navigator Key Trustee
- Provides a unified API and scalability
- REST API
- Does not actually store keys (backend does that), but does cache them
- ACLs on per-key basis
HDFS Encryption Configuration

- `hadoop key create <keyname> -size <keySize>
- `hdfs dfs -mkdir <path>
- `hdfs crypto -createZone -keyName <keyname> -path <path>
KMS Per-User ACL Configuration

- White lists (check for inclusion) and black lists (check for exclusion)
- `etc/hadoop/kms-acls.xml`
  - `hadoop.kms.acl.CREATE`
  - `hadoop.kms.blacklist.CREATE`
  - ... `DELETE`, `ROLLOVER`, `GET`, `GET_KEYS`, `GET_METADATA`, `GENERATE_EEK`, `DECRYPT_EEK`
  - `hadoop.kms.acl.<keyname>.<operation>`
  - `MANAGEMENT`, `GENERATE_EEK`, `DECRYPT_EEK`, `READ`, `ALL`
Best practices

- Enable authentication (Kerberos)
- Enable TLS/SSL
- Use KMS acls to setup KMS roles, blacklist HDFS admins and grant per key access
- Do not use the KMS with default JCEKS backing store
- Use hardware that offers AES-NI instruction set
  - Install openssl-devel so Hadoop can use Openssl crypto codec
- Make sure you have enough entropy on all the nodes
  - Run rngd or haveged
Best practices

- Do not run KMS on master or worker nodes
- Run multiple instances of KMS for high availability and load balancing
- Harden KMS instance and use internal firewall so only KMS and ssh etc. ports are reachable from known subnets
- Make secure backups of KMS
HDFS Encryption - Summary

- Good performance (4-10% hit) with AES-NI
- No mods to existing applications
- Prevents attacks at the filesystem and below
- Data is encrypted all the way to the client
- Key management is independent of HDFS
- Can prevent HDFS admin from accessing secure data
Demo

- Accessing HDFS encrypted data from Linux storage

<table>
<thead>
<tr>
<th>User</th>
<th>Group</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>hdfs_admin</td>
<td>cdh_admin</td>
<td>HDFS Admin</td>
</tr>
<tr>
<td>kms_admin</td>
<td>cdh_admin</td>
<td>KMS Admin</td>
</tr>
<tr>
<td>alice</td>
<td>cdh_user</td>
<td>User with DECRYPT_EEK access to key1</td>
</tr>
<tr>
<td>bob</td>
<td>cdh_user</td>
<td>User with DECRYPT_EEK access to key2</td>
</tr>
</tbody>
</table>
Questions?
Hadoop Data Governance

Mark Donsky

Director, Product Management
Cloudera
Data Governance
Frequently Asked Questions

What data do I have?
How did the data get here?
Who used the data?
How has the data been used?
How do I answer these questions at scale?
Compliance + Productivity = Adoption

**Compliance/Governance**
- Am I prepared for an audit?
- Who's accessing sensitive data?
- What are they doing with the data?
- Is sensitive data governed and protected?

**End User Productivity**
- How can I find explore data sets on my own?
- Can I trust what I find?
- How do I use what I find?
- How do I find and use related data sets?
What makes governance so difficult?

<table>
<thead>
<tr>
<th>Hadoop governance challenges</th>
<th>Cloud governance challenges</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Variety, Volume, Velocity</td>
<td>• Multiple storage types:</td>
</tr>
<tr>
<td>• Multiple compute types:</td>
<td>HDFS, S3, ADLS, etc.</td>
</tr>
<tr>
<td>Spark, Hive, Pig, MR,</td>
<td>• Transient clusters</td>
</tr>
<tr>
<td>MR2, Sqoop, etc.</td>
<td>• Long-running clusters</td>
</tr>
<tr>
<td>• Multiple third-party tools</td>
<td>• Shared Hive Metastores</td>
</tr>
</tbody>
</table>

Yet the business still needs one set of trusted governance artifacts
Governance: the Foundation of Data Management

Compliance
Track, understand and protect access to data
- Am I prepared for an audit?
- Who’s accessing sensitive data?
- What are they doing with the data?
- Is sensitive data governed and protected?

Stewardship
Manage and organize data assets at scale
- How can I efficiently manage data lifecycle, from ingest to purge?
- How can I efficiently organize and classify all my data?
- How can I efficiently make data available to my end users?

End User Productivity
Effortlessly find and trust data sets
- How can I find and explore data sets on my own?
- Can I trust what I find?
- How do I use what I find?
- How do I find and use related data sets?

Administration
Boost user productivity and cluster performance
- Is my data optimized to support current access patterns?
- How can I optimize for future workloads?
- How can I migrate workloads to Hadoop risk-free?

Hadoop Governance Foundation
- Centralized audits
- Unified metadata catalog
- Comprehensive lineage
- Data policies

#StrataHadoop
Hadoop Governance Requirements

- Unified metadata catalog
- Centralized audits
- Comprehensive lineage
- Data policies
## Unified Metadata Catalog

### Technical Metadata
- All files in directory `/sales`
- All files with permissions 777
- Anything older than 7 years
- Any not accessed in the past 6 months

### Managed Metadata
- Sales data from last quarter for the Northeast region
- Protected health information
- Business glossary definitions
- Data sets associated with clinical trial X

### Custom Metadata
- Tables that I want to share with my colleagues
- Data sets that I want to retrieve later
- Data sets that are organized by my personal classification scheme (e.g., “quality = high”)

### Challenges
- Technical metadata in Hadoop is component-specific
- Curated/custom attributes: Hive metastore has comments, and HDFS has extended attributes, but:
  - Not searchable
  - No validation
- Aggregated analytics are not possible
  - How many files are older than two years?
Data Policies

- **Goal:** Manage and automate the information lifecycle from ingest to purge/cradle to grave, based on the unified metadata catalog.

- Once you find data sets, you’ll likely need to do something with them:
  - Tag every new file that lands in /sales as sales data.
  - Send an alert whenever a sensitive data set has permissions 777.
  - Purge all files that are older than seven years.

**Challenges**

- Oozie workflows can be difficult to configure.
- Event-triggered oozie workflows are limited to very few technical metadata attributes, such as directory path.
- Data stewards prefer to define, view, and manage data policies in a metadata-centric fashion.
Centralized Audits

**Goal:** Collect all audit activity in a single location
- Redact sensitive data from the audit logs to simplify compliance with regulations
- Perform holistic searches to identify data breaches quickly
- Publish securely to enterprise tools

**Challenges**
- Each component has its own audit log, but:
- Sensitive data may exist in the audit log
- Select * from transactions where cc_no = “1234 5678 9012 3456”
- It’s difficult to do holistic searches
  - What did user a do yesterday?
  - Who accessed file f?
- Integration with enterprise SIEM and audit can be complex
Comprehensive Lineage

Challenges

• Most uses of lineage require column-level lineage

• Hadoop does not capture lineage in an easily-consumable format

• Lineage must be collected automatically and cover all compute engines

• Third-party tools and custom-built applications need to augment lineage
Use Cases
**Use Cases: Compliance**

**Compliance**
Track, understand and protect access to data

Am I prepared for an audit?
Who’s accessing sensitive data?
What are they doing with the data?
Is sensitive data governed and protected?

**Hadoop Data Governance & Management**
- Centralized audits
- Unified metadata catalog
- Comprehensive lineage
- Data policies

**Enterprise Metadata Repository**

**Enterprise Auditing & Security**

**Common use cases:**
- Security breach detection
- Data access tracking for PCI compliance
- Audit defense
# Use Cases: Administration

## Administration
- Boost user productivity and cluster performance
- Is my data optimized to support current access patterns?
- How can I optimize for future workloads?
- How can I migrate workloads to Hadoop risk-free?

## Visibility
- Distribution of data objects
- Workloads by engine

## Optimization
- Sub-optimal query patterns
- "Rogue" users
- Capacity planning

## Patterns
- Data churn over time
- Table clusters
- Frequent users

## Unexpected Behavior
- Hive tables suddenly missing
  - `rm -rf /user/hive/warehouse`
Demo
Questions
Final Thoughts
Compliance

- We have shown how an EDH environment can be secured end-to-end
- Is this enough to be compliant?
  - PCI DSS
  - HIPAA
  - Internal compliance – PII data handling
- All of the security features discussed (and others not covered because of time) are enough to cover technical requirements for compliance
- However, compliance also requires additional **people** and **process** requirements
- Cloudera has worked with customers to achieve PCI DSS compliance as well as others – **you can do it too!**
Public Cloud Security

- Many Hadoop deployments occur in the public cloud
- Security considerations presented today all still apply
- Complementary to native cloud security controls

- [Cloudera blog post - How-to: Deploy a secure enterprise data hub on AWS](http://blog.cloudera.com/blog/2016/05/how-to-deploy-a-secure-enterprise-data-hub-on-aws/)
Looking Ahead

- The Hadoop ecosystem is vast, and it can be a daunting task to secure everything.
- Understand that **no system is completely secure**.
- However, the proper security controls coupled with regular reviews can **mitigate** your exposure to threats and vulnerabilities.
- Pay attention to new components in the stack, as these components often **do not** have the same security features in place.
  - Kafka only recently added wire encryption and Kerberos authentication.
  - Spark only recently added wire encryption.
  - Many enterprises were using both of these in production before those features were available!
Final Questions?

Thank you!