DEVIEW
2019

대용량 멀티테넌트 시큐어 하둡 클러스터를 시행착오 없이 만들기

권병창

NAVER BIG DATA & AI PLATFORM

CONTENTS

DEVIEW
2019

DEVIEW
2019

1. Hadoop Security를 시작하기전 고려사항

1.1 Hadoop Security가 필요한 이유

개인정보 같은 민감한 정보가 포함된 데이터가 증가하고 있고
이를 빅데이터 분석을 하려는 요구도 증가하고 있다
민감한 정보는 권한이 있는 사용자만 접근할 수 있어야 한다
사용자가 많아질수록 보안을 사용자의 선의에만 맡길 수 없다
일부 사용자의 실수가 전체 사용자에게 영향을 주는 상황이 발생한다

1.2 필요했던 사례

하둡보안을 적용하지 않은 클러스터에서 다음과 같은 일들이 발생했었다

1. YARN 작업/앱을 모두 Kill 하는 스크립트가 수행됨
2. HDFS 의 임의의 디렉토리가 삭제됨
3. 민감한 데이터를 분석하려는 요구가 있었는데 할 수 없음
4. HBase shutdown() RPC 호출로 HBase 클러스터 셧다운 됨

1.3 진짜 필요한가?

작은 조직 내에서 사용하는 경우에는 꼭 필요하지 않다
닭 잡는 데 소 잡는 칼이 필요하지 않다

작은 조직의 판단은
클러스터 운영자와 사용자가 같은 공간 내에서 얼굴을 보며 만날 수 있는 규모
사용자를 직접 만날 일이 없는 규모라면 하둡 보안이 필요하다
민감한 데이터를 다루는 조직이라면 조직 규모에 상관없이 필수이다

2. Hadoop Security Overview

2.1 세 가지의 A

DEVIEW
2019

Security를 처음 시작할 때 자주 헷갈리는 개념
하둡뿐 아니라 대부분의 Security 관련된 곳에서 사용하는 개념

- Authentication
- Authorization
- Auditing

2.1.1 Authentication (인증)

사용자의 신분을 확인하는 과정
다양한 방법이 있다
 일반적으로 비밀번호를 많이 사용
 은행/게임/웹사이트에서 사용하는 OTP
 스마트폰의 얼굴/지문인식

Hadoop에서는 Kerberos, Delegation Token을 사용한다

2.1.2 Authorization (권한부여, 허가)

사용자가 요청한 리소스의 접근을 허가/거부를 확인하는 과정
이 단계는 Authentication 이 완료된 이후에 진행된다
일반적으로 Access Control Lists (ACL)로 관리한다
권한 부여를 안전하게 하려면 사용자가 안전한 방법으로 인증되어야 한다
허술한 인증으로는 권한 부여를 올바르게 할 수 없다

HDFS: POSIX Permissions, POSIX ACLs
YARN: Queue ACLs (admin-queue, submit-app), Job ACLs (view, modify)

2.1.3 Auditing (감사)

인증(Authentication) 및 권한 부여(Authorization) 과정과 결과를 audit log에 기록한다
Auditing은 공격자로부터 시스템을 보호하지 못한다
시스템이 공격당했을 때 언제/어디서/어떤 경로로 뚫렸는지를 확인할 수 있는 수단이다

HDFS: hdfs-audit.log
YARN: rm-audit.log, nm-audit.log
Kerberos KDC: krb5kdc.log

2.1.4 Authentication vs Authorization

DEVIEW 2019

두 개념이 섞여 있는 경우가 많이 있어서 헷갈릴 수 있다
/etc/passwd 에 두 개념이 모두 녹아 있다. 아래 같은 데이터가 있을 때

```
joe:X:1234:56:/home/joe:/bin/bash
```

사용자 joe 는 시스템에 로그인할 수 있는 권한이 (Authorization)이 있다
하지만 비밀번호가 일치(Authentication)해야지만 로그인할 수 있다

2.1.5 세 가지 A 의 실생활 사례

호텔에 숙박 할 때

1. 호텔 프런트에서 신분증을 제출한다 (Authentication)
2. 호텔 예약목록에 등록되어 있는지 확인한다 (Authorization)
3. 숙박부를 작성한다 (Auditing)

2.2 Kerberos

하둡의 인증은 커버러스를 사용한다

커버러스 특징

1. 중앙화된 서버에서 사용자 정보를 관리한다
2. 평문의 비밀번호를 네트워크로 절대 전송하지 않는다
3. 상대적으로 짧은 유효기간의 티켓을 사용한다
4. Single Sign On

2.2.1 Kerberos 용어

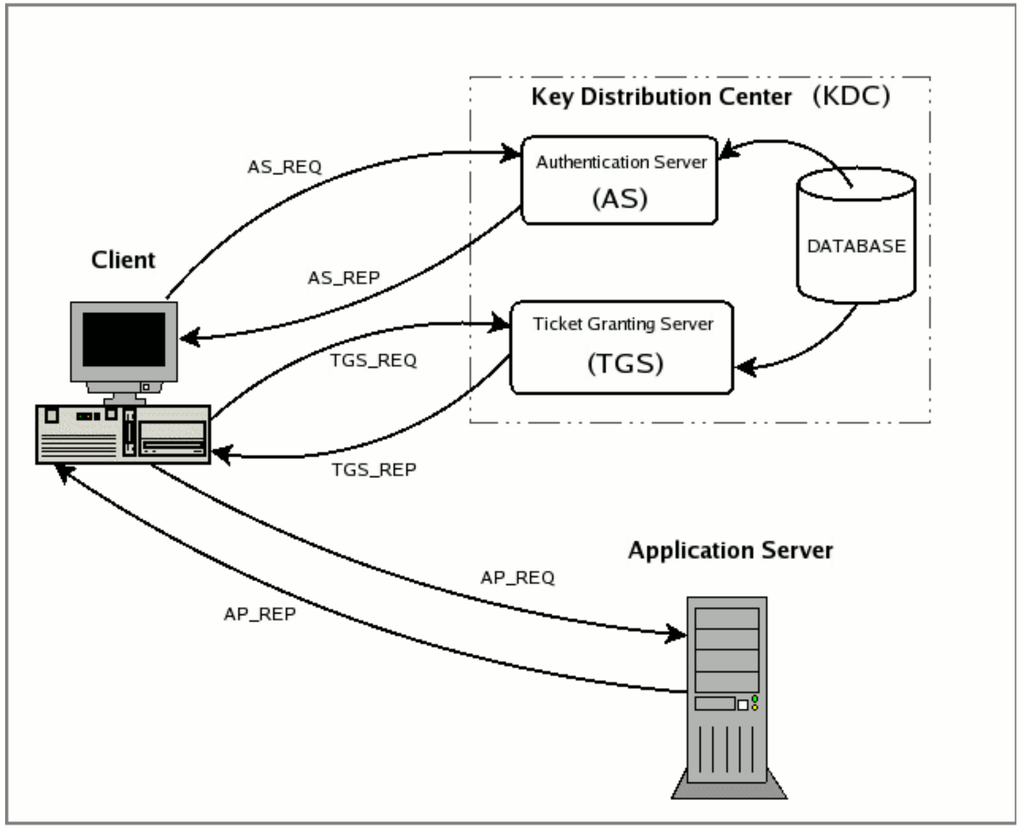
DEVIEW 2019

용어	설명
Realm	커버러스 서버가 인증을 제공하는 범위. 관습적으로 대문자의 도메인을 사용한다. 대소문자 구분한다 예> X.NAVER.COM
Principal	커버러스의 사용자 ID, 사용자는 사람이 아닐 수 있다 예> nn/nn1.example.com@X.NAVER.COM foo@X.NAVER.COM
Keytab	Key Table의 약자. Principal의 비밀번호를 저장해둔 파일
Ticket	클라이언트의 요청으로 커버러스 서버(KDC) 에서 발급하며 클라이언트가 서비스 접속할 때 사용한다

2.2.2 KDC 구성 요소

DEVIEW 2019

용어	설명
KDC	Key Distribution Center의 약자. AS(Authentication Server), TGS (Ticket Granting Server), Principals과 비밀번호를 저장한 DB로 구성되어 있다*
Authentication Server (AS)	클라이언트의 요청으로 TGT (Ticket Granting Ticket) 발급하는 서버 이때 클라이언트는 비밀번호를 AS에 보내지 않고 AS는 TGT를 요청한 클라이언트의 비밀번호로 암호화해서 보낸다 클라이언트는 자신의 비밀번호로 TGT를 복호화한다**
Ticket Granting Ticket (TGT)	TGS(Ticket Granting Server)에 접속할 있는 티켓*** TGT 발급이 성공하면 로그인이 완료된 상태
Ticket Granting Server (TGS)	TGT(... Ticket) 를 가지고 있으면 TGS에 접속할때 비밀번호는 필요 없다**** 클라이언트가 접속하려는 서비스의 티켓을 발급하는 서버 서비스 티켓을 발급받으면 해당 서비스에 비밀번호 없이 접속할 수 있다



* 커버러스 특징1, ** 특징2, *** 특징3, **** 특징4

https://zeroshell.org/kerberos/kerberos-operation

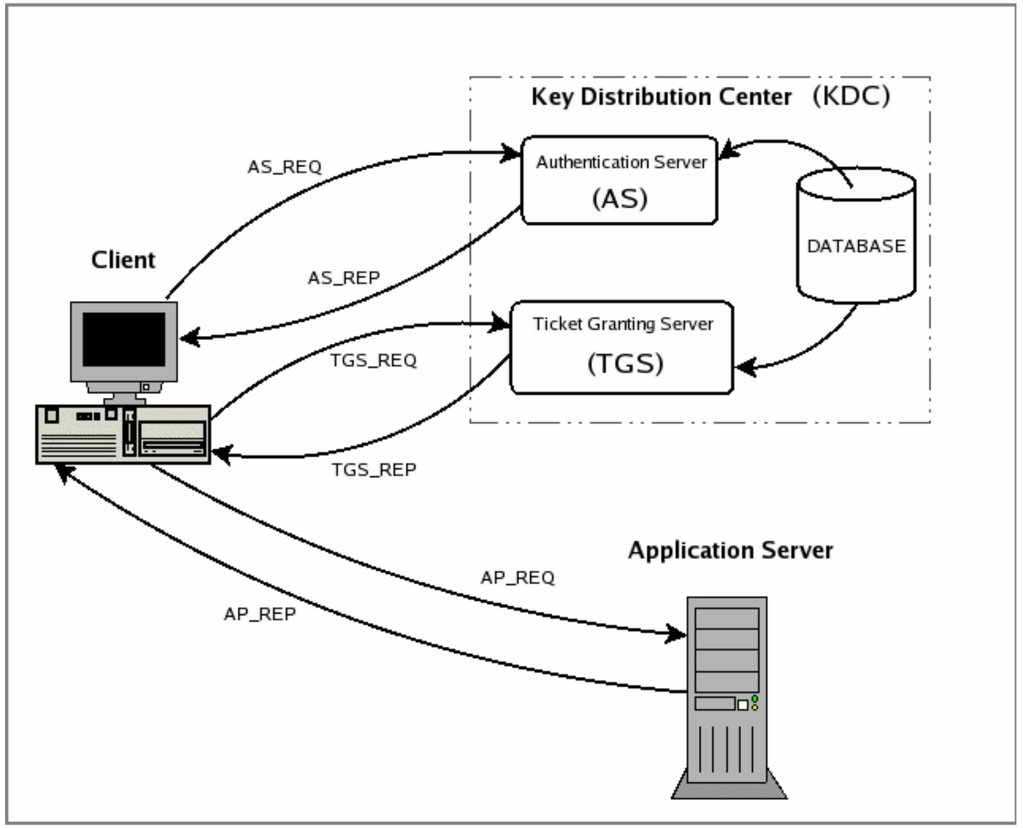
2.2.2 KDC 구성 요소

DEVIEW 2019

용어	설명
KDC	Key Distribution Center의 약자. AS(Authentication Server)와 TGS(Ticket Granting Server), Principals과 비밀번호를 저장한 DB로 구성되어 있다*
Authentication Server (AS)	클라이언트의 요청으로 TGT (Ticket Granting Ticket) 발급하는 서버 이때 클라이언트는 비밀번호를 AS에 보내지 않고 AS는 TGT를 요청한 클라이언트의 비밀번호로 암호화해서 보낸다 클라이언트는 자신의 비밀번호로 TGT를 복호화한다**
Ticket Granting Ticket (TGT)	TGS(Ticket Granting Server)에 접속할 수 있는 티켓*** TGT 발급이 성공하면 로그인이 완료된 상태
Ticket Granting Server (TGS)	TGT(... Ticket) 를 가지고 있으면 TGS에 접속할때 비밀번호는 필요 없다**** 클라이언트가 접속하려는 서비스가 있으면 TGS는 서버 서비스 티켓을 발급받으면 해당 서비스에 비밀번호 없이 접속할 수 있다

1. 중앙화된 서버에서 사용자 정보를 관리한다
2. 평문의 비밀번호를 네트워크로 절대 전송하지 않는다
3. 상대적으로 짧은 유효기간의 티켓을 사용한다
4. Single Sign On

* 커버러스 특징1, ** 특징2, *** 특징3, **** 특징4



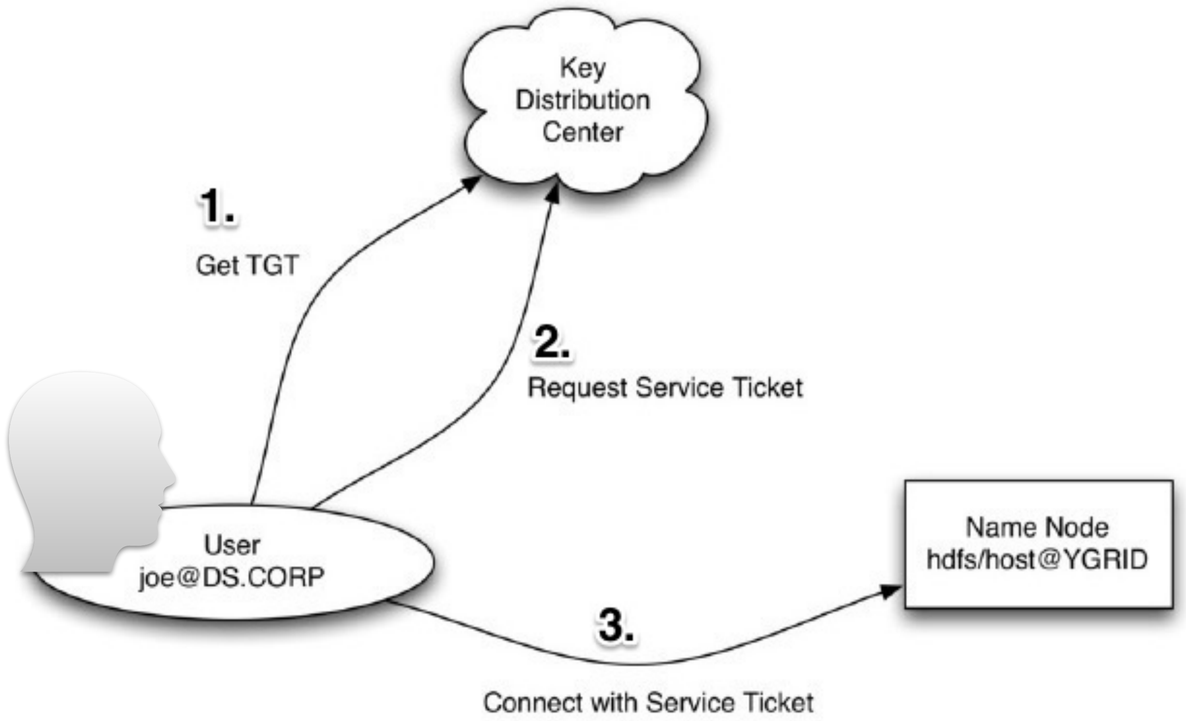
https://zeroshell.org/kerberos/kerberos-operation

2.2.3 Kerberos Ticket

KDC (TGS) 에서 발급한 암호화된 데이터 구조
비밀번호 없이 서비스에 접속할 수 있게 해주는 수단
티켓은 유효기간이 있다
- 일반적으로 24h 사용 (ticket_lifetime)
- 아침에 출근해서 한번 로그인해서 티켓을 발급받으면 퇴근할 때까지 재 로그인을 할 필요 없다

유효기간이 만료되기 전에 갱신(renew) 할 수 있으나 무한히 갱신할 수 없고 제한이 있다
- 일반적으로 7d 사용 (renew_lifetime)
- 갱신할 때에는 비밀번호가 없어도 된다
- 24h 유효기간이 만료되면 7d 이내여도 renew 할 수 없다

2.2.4 네임노드 접속 흐름



https://www.slideshare.net/hadoopusergroup/hadoop-security-preview#8

2.3 Delegation Token

하둡의 인증은 커버러스와 Delegation Token을 사용한다
커버러스를 분산 작업에 사용할 경우 KDC 가 병목이 된다
커버러스를 보완하기 위해서 하둡에서 Delegation Token을 도입함

1. 클라이언트는 커버러스 인증 후에 네임노드에서 Delegation Token 발급
2. 이후부터 클라이언트는 Delegation token을 사용해서 네임노드에 인증한다

리소스매니저, 히스토리서버, 타임라인서버 등에서도 Delegation Token 발급함

2.3.1 Delegation Token: 시간 속성

DEVIEW 2019

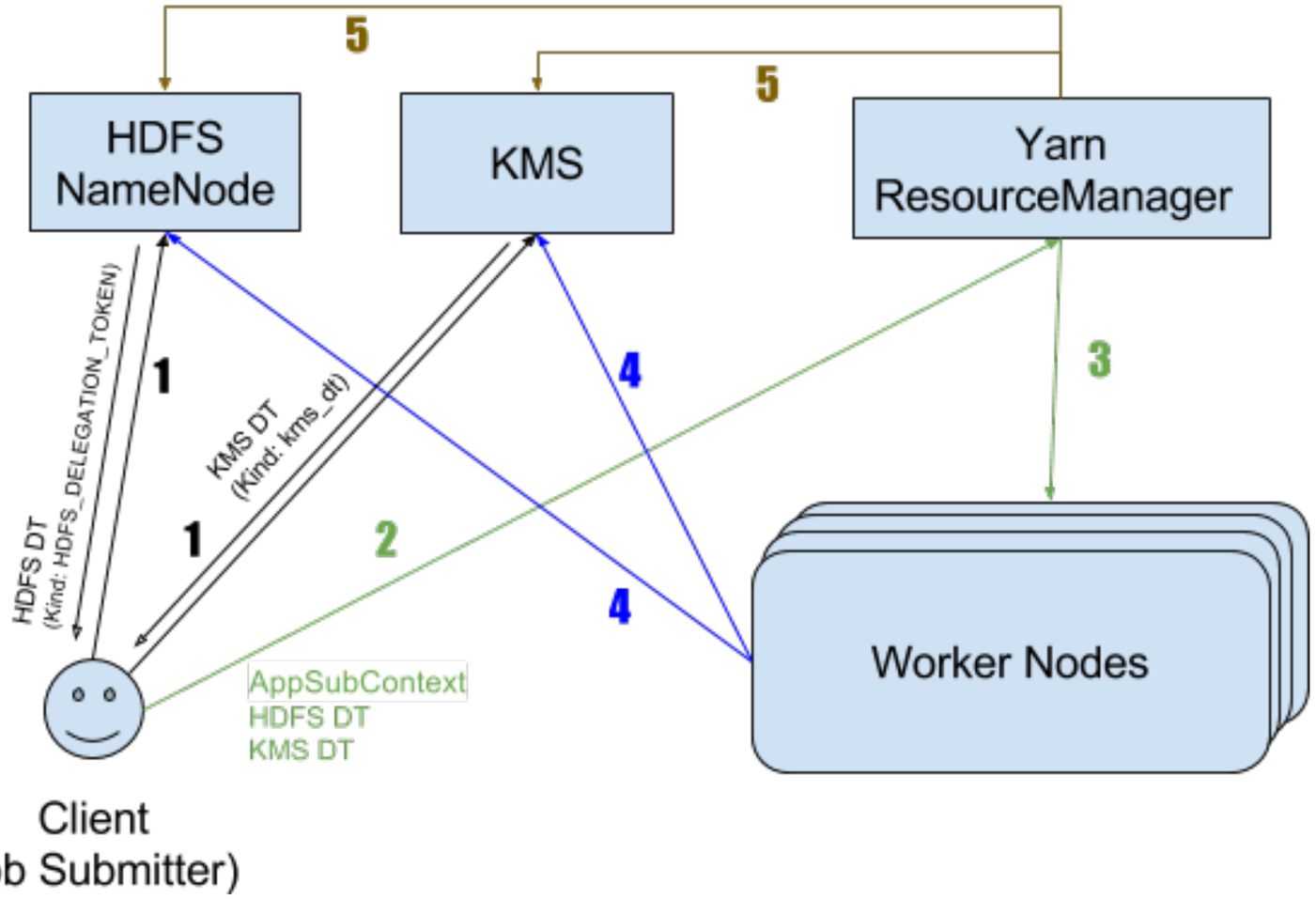
Delegation token 에는 2가지의 시간 속성이 있다
- renew-interval: 24시간
- max-lifetime: 7일

1. Delegation token은 24시간(renew-interval)내에 renew 하지 않으면 무효화된다
2. renew 로 유효기간이 연장되지만 최대 7일(max-lifetime)을 초과해서 연장할 수 없다
3. 7일이 지나면 Delegation token은 무효가 되어서 사용할 수 없다

2.3.2 MapReduce 제출 흐름

DEVIEW 2019

1. 커버러스 인증으로 네임노드에 Delegation Token (이하 DT) 발급(get) 요청
2. DT 을 포함하여 MapReduce 작업을 리소스매니저(이하 RM)에 제출
3. RM 에서 DT 유효성체크 후, MR Job 런칭
4. map, reduce task 에서 배포된 DT로 네임노드 접속
5. a. RM 에서 주기적으로 네임노드에 DT 갱신(renew) 요청
   b. 작업이 완료되면 RM 에서 네임노드에 DT 삭제(cancel) 요청



https://blog.cloudera.com/hadoop-delegation-tokens-explained/

2.4 Kerberos Ticket vs Delegation Token

DEVIEW 2019

	Kerberos Ticket	Delegation Token (DT)
갱신(renew) 이 가능한가?	가능	가능
유효기간(soft limit)*	1일	1일
유효기간(hard limit)*	7일	7일
서버에서 데이터를 저장하나?	KDC 에서 Ticket을 저장하지 않음	DT를 발급한 서버에서 DT를 저장함
갱신(renew)할 때 데이터 변경되나?	KDC 에서 새로운 티켓을 발급받아서 티켓이 변경됨	DT는 새로 발급받지 않아서 변경 없고 서버에서만 유효기간이 연장됨
로그아웃 방법	로컬에 저장된 티켓을 삭제	서버에 DT 삭제를 요청해야 함
누가 renew 를 할 수 있나?**	티켓을 가지고 있는 클라이언트	DT 발급을 요청할 때 renewer를 지정함 지정된 renewer만 renew를 할 수 있음

* 서버설정으로 운영자가 변경할 수 있다

** 분산작업(MR, Spark, Tez)에서 Delegation token의 renewer 는 일반적으로 리소스매니저이고 renew-interval(soft-limit)의 90% 일 때 renew 한다

2.4 Kerberos Ticket vs Delegation Token

DEVIEW 2019

	Kerberos Ticket	Delegation Token (DT)
갱신(renew) 이 가능한가?	가능	가능
유효기간(soft limit)*	1일	1일
유효기간(hard limit)*	7일	7일
서버에서 데이터를 저장하나?	KDC 에서 Ticket을 저장하지 않음	DT를 발급한 서버에서 DT를 저장함
갱신(renew)할 때 데이터 변경되나?	KDC 에서 새로운 티켓을 발급받아서 티켓이 변경됨	DT는 새로 발급받지 않아서 변경 없고 서버에서만 유효기간이 연장됨
로그아웃 방법	로컬에 저장된 티켓을 삭제	서버에 DT 삭제를 요청해야 함
누가 renew 를 할 수 있나?**	티켓을 가지고 있는 클라이언트	DT 발급을 요청할 때 renewer를 지정함 지정된 renewer만 renew를 할 수 있음

이런 특징으로 커버러스 KDC 는 분산환경에서 병목이 됨

* 서버설정으로 운영자가 변경할 수 있다

** 분산작업(MR, Spark, Tez)에서 Delegation token의 renewer 는 일반적으로 리소스매니저이고 renew-interval(soft-limit)의 90% 일 때 renew 한다

2.5 SPNEGO

Simple and Protected GSSAPI Negotiation Mechanism

하둡에는 HTTP REST API도 지원하고 Web-UI도 있다
HTTP 프로토콜에서 Kerberos를 지원하는 프로토콜이다

발음은 /spenay-go/ /스페네이고/

2.6 LDAP

Lightweight Directory Access Protocol
디렉터리 서비스
파일 시스템의 디렉터리(폴더)는 아니다
현실 세계의 주소록을 컴퓨터로 옮겨놓은 프로토콜
사용자 계정과 그룹관리를 위해 사용한다
Hadoop Security에서 필수는 아니지만 사용하면 클러스터 운영이 매우 편해진다

2.6.1 LDAP 이 필요한 이유

시큐어 환경에서 MapReduce 같은 모든 작업은
각 노드에서 하둡 계정과 일치하는 OS 계정으로 실행되어야 한다
하둡 계정이 100개가 있으면 모든 노드의 OS에 100개의 계정이 모두 있어야 한다
각 노드를 LDAP 과 연동하고 LDAP 서버에서 계정을 추가하면
연동된 모든 노드에 계정이 추가된다

DEVIEW
2019

3.Secure Hadoop Cluster 설치과정

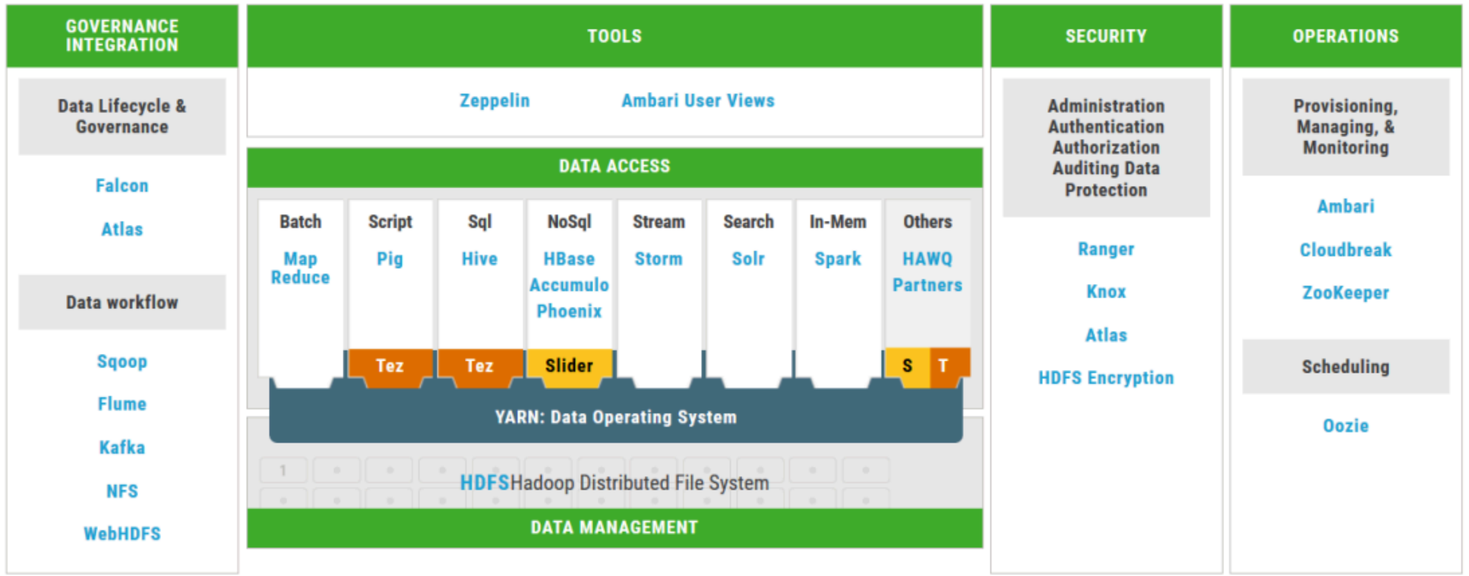
3.1 사용한 패키지

Ambari-2.7.3

HDP-3.1.1

MIT Kerberos 1.15.1

OpenLDAP 2.4.44



https://web.mit.edu/kerberos/

http://www.openldap.org/

https://docs.cloudera.com/HDPDocuments/HDP3/HDP-3.1.4

https://www.cloudera.com/products/open-source/apache-hadoop/apache-ambari.html

3.2 커버러스 KDC 와 LDAP 서버

커버러스 KDC 와 LDAP 서버설치는 Ambari 에서 지원하지 않는다
운영자가 직접 설치해야한다
SPoF 가 되지 않게 HA(High Availability) 를 고려해야한다
MIT Kerberos 와 OpenLDAP 에서 HA 를 지원한다

3.3 설치흐름

DEVIEW
2019

1. 커버러스 KDC 와 LDAP 서버 설치*
2. 클러스터의 모든 노드를 LDAP 연동*
3. Ambari 로 클러스터 설치
4. Ambari Kerberos Wizard 로 시큐어 클러스터로 변경
5. 대규모 환경에 적합하게 Hadoop 설정 변경
6. Ambari, Ranger, Zeppelin 등의 Web-UI LDAP 연동*

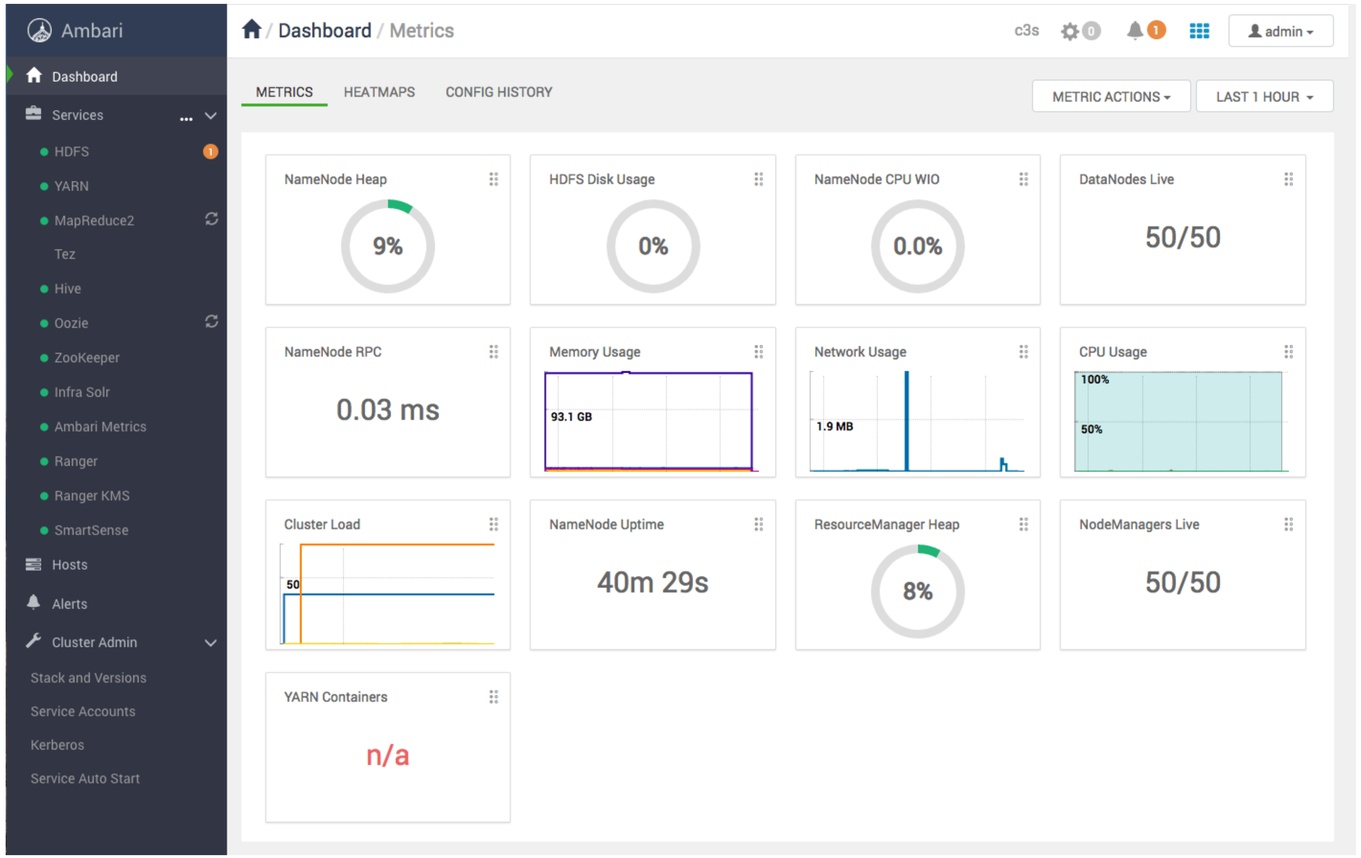
* 설치/연동은 Appendix 참고

3.4 설치는 매우 쉽습니다

1. 복잡한 설정을 Ambari 가 모두 해줘서 클러스터 설치와 커버러스 적용은 매우 쉽다
2. 상황에 따라 다양한 이슈가 발생할 수 있다
3. 다양한 이슈경험과 처리를 위해서 설치과정의 반복학습은 매우 중요하다
4. 테스트용 장비에서 구성을 달리하여 설치과정을 여러차례 반복한다
5. 노드 추가/디커미션등의 운영업무도 반복학습을 한다

3.5 Ambari Dashboard

DEVIEW 2019

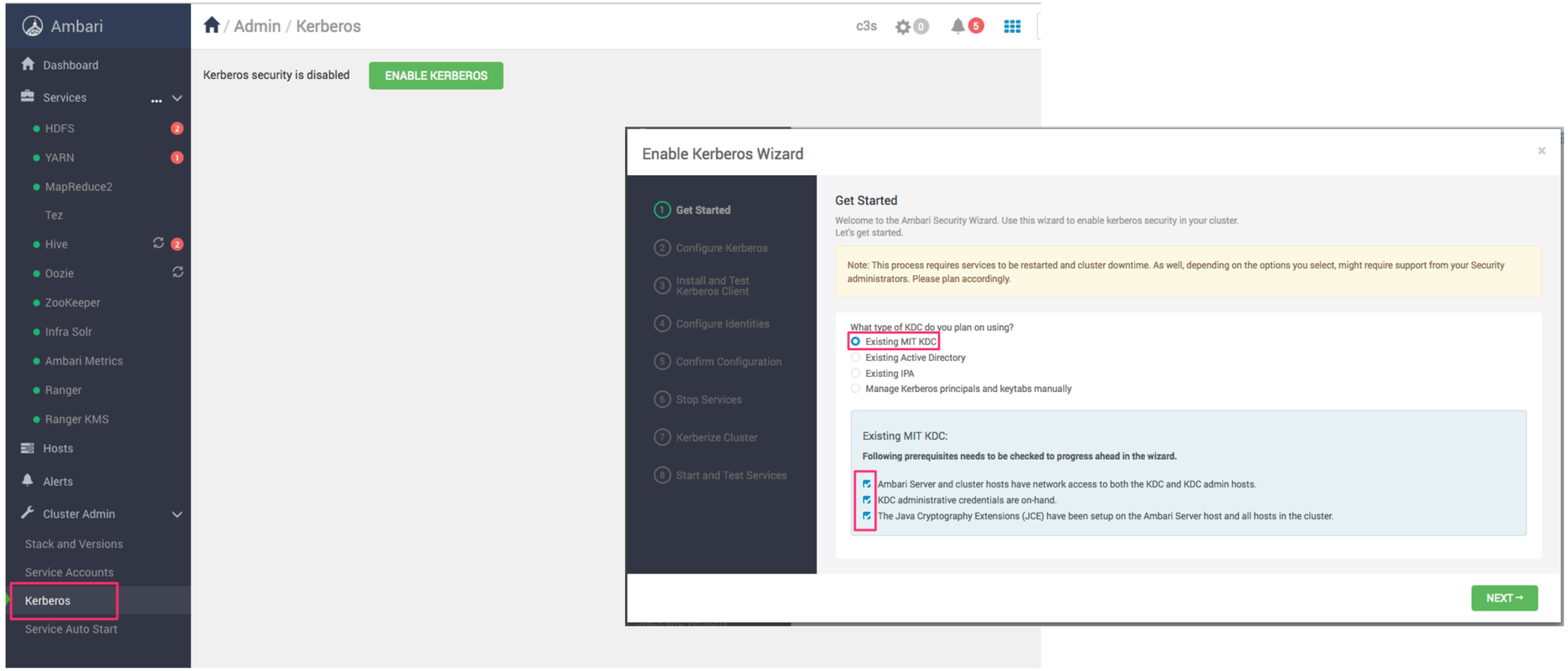


Ambari 로 보안설정이 없는 클러스터 설치완료된 화면

https://docs.cloudera.com/HDPDocuments/Ambari-2.7.3.0/bk_ambari-installation/content/ch_Getting_Ready.html

3.5.1 Kerberos Wizard

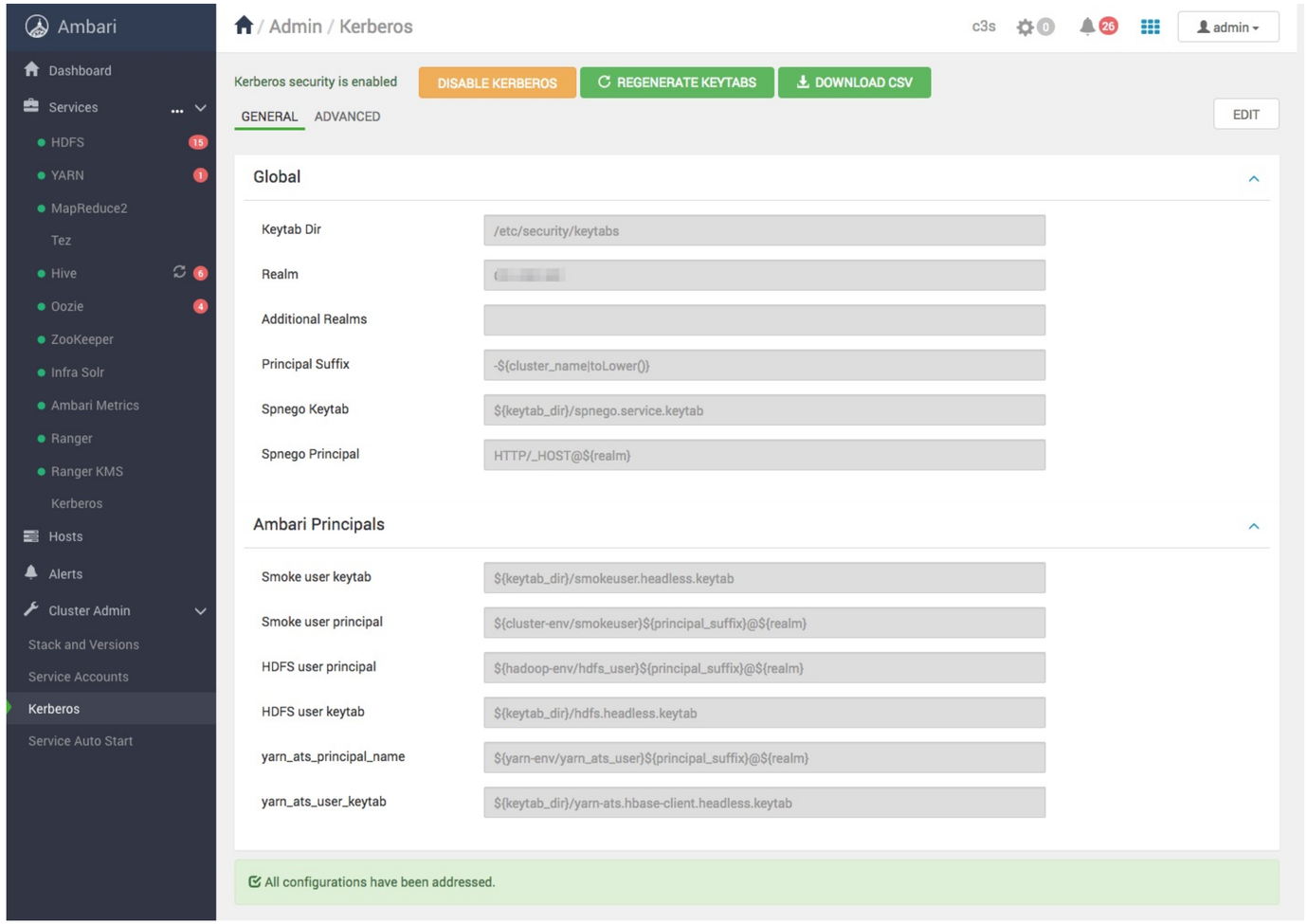
DEVIEW 2019



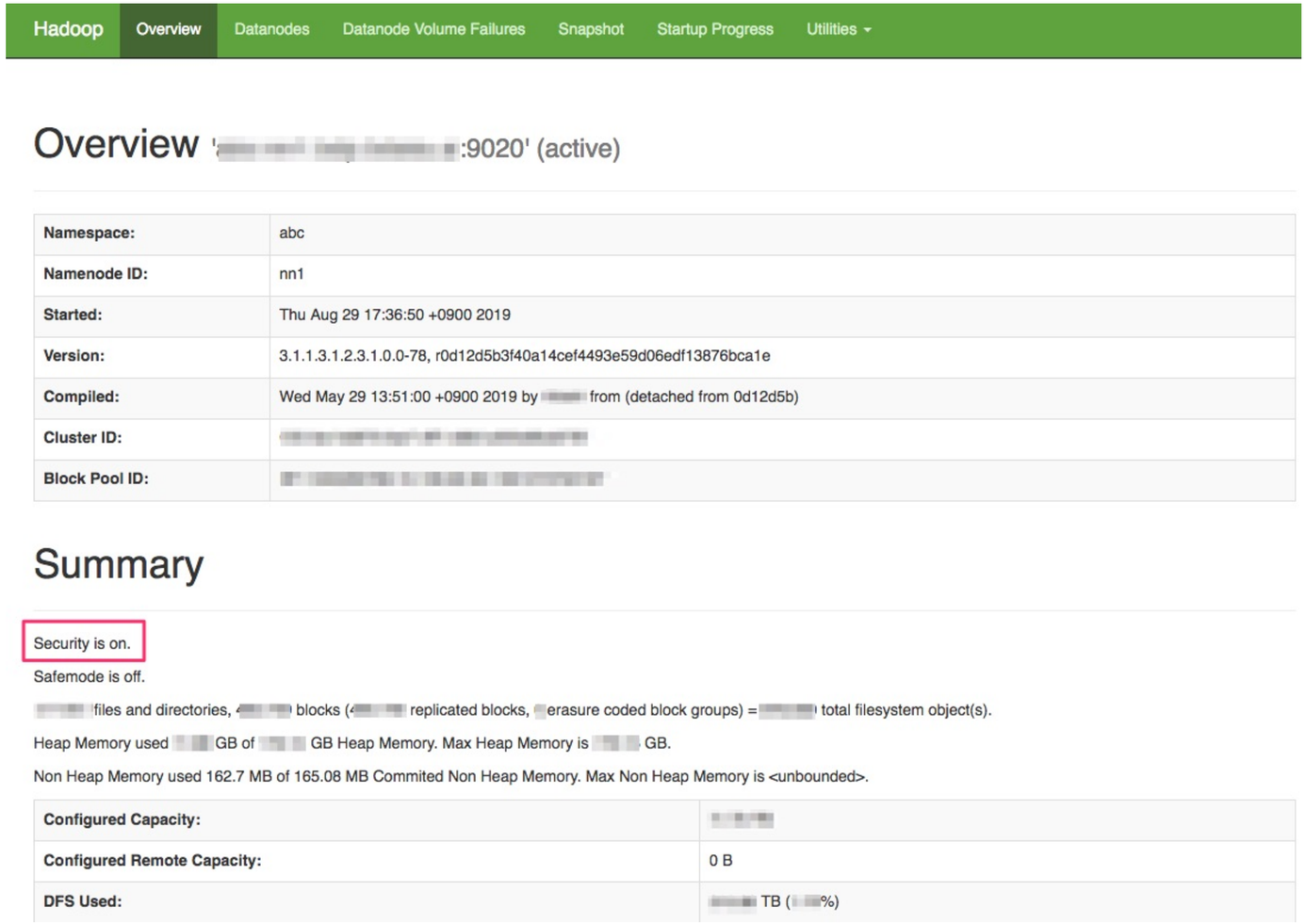
3.5.2 Kerberos 적용 후의 변화

DEVIEW 2019

변경된 Hadoop Security 관련 설정을 확인할 수 있다



3.5.3 Kerberos 적용 후의 변화



3.6 하둡 설정 변경

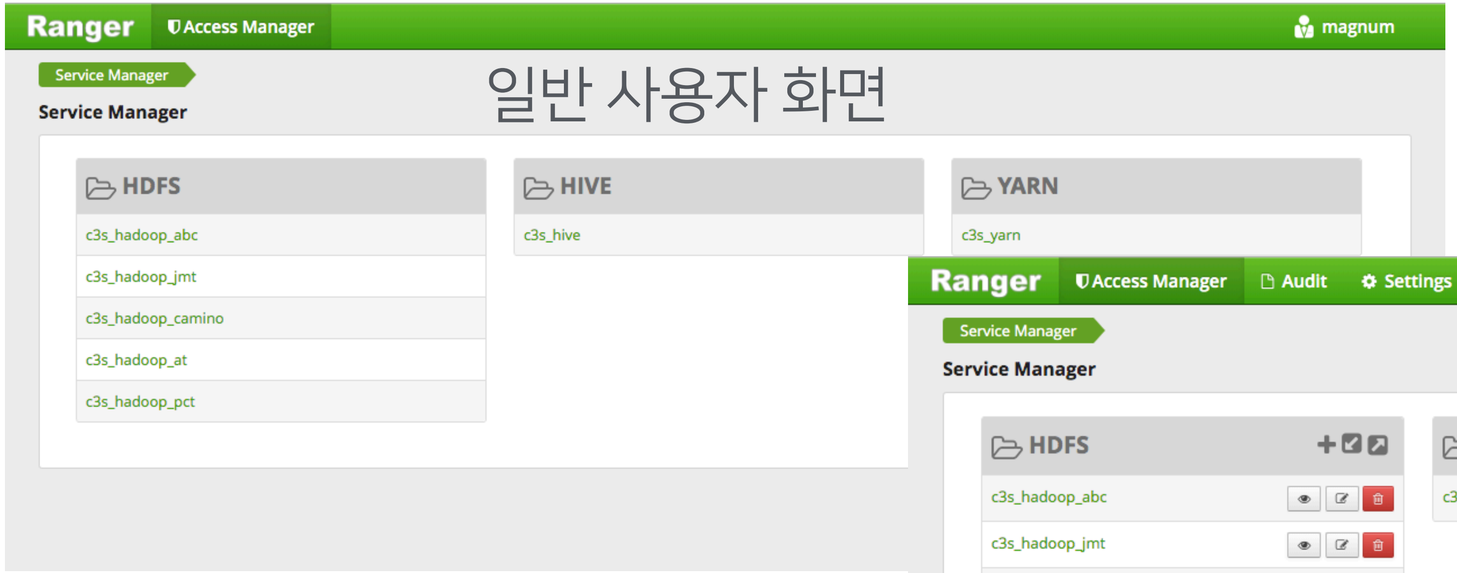
Ambari 가 제안하는 설정은 대규모 환경에서 최적은 아니다.
DataProc(a.k.a C3)에서 사용하는 설정은 Appendix 참고

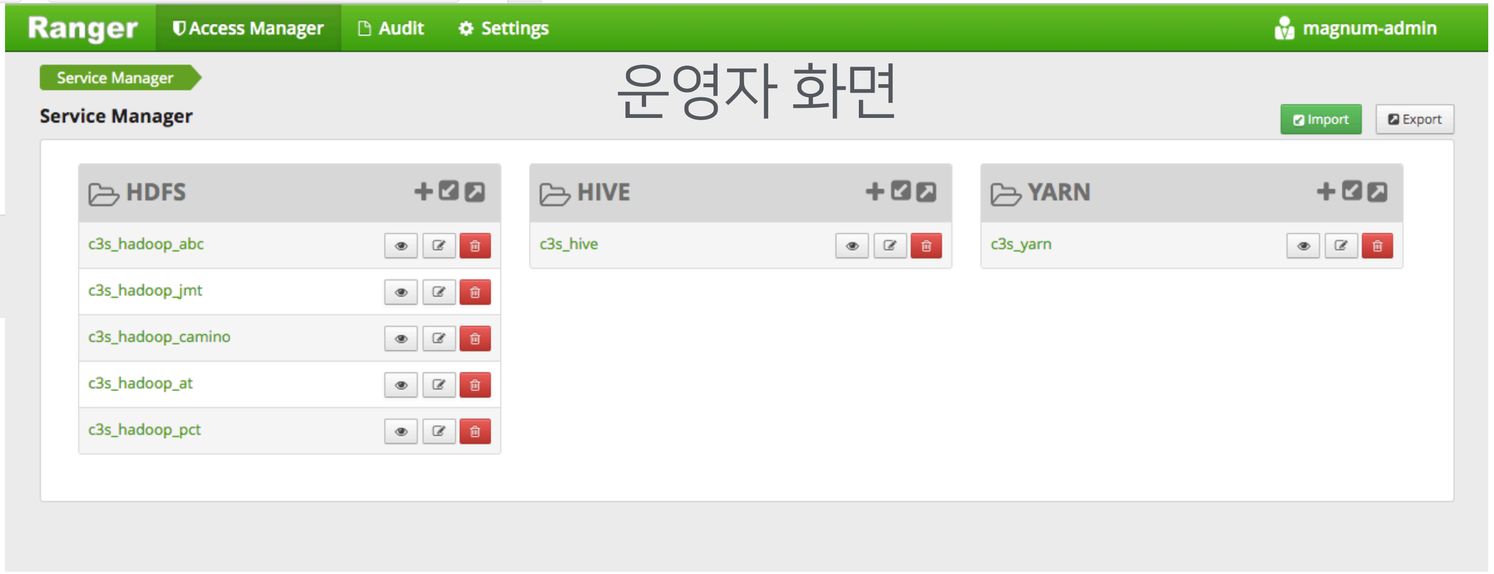
3.7 Web-UI의 LDAP 연동

DEVIEW 2019

Ambari, Ranger, Zeppelin 같은 Web-UI에 LDAP 을 연동한다

LDAP 을 연동하면 LDAP 서버에 저장된 사용자 정보로 Web-UI 로그인이 가능하다





DEVIEW
2019

4.클러스터 사용자들의 하둡 사용 패턴 변화

4.1 개별계정을 사용한다

공용계정을 사용하지 않고 개별계정을 사용한다
개별계정은 사람 단위일 수도 있고 업무 단위일 수도 있다
업무 단위일 경우 각 업무를 담당하는 사람들이 공용으로 사용하는 계정이 될 수 있다
각자의 상황에 맞게 정책을 정해야 한다
중요한 건 모든 사용자가 같은 계정을 사용하지 않는다

4.1.1 하둡 계정 만들기

하둡 계정 foo를 만드는 절차

1. Kerberos Principal (foo@X.NAVER.COM) 추가 후 Keytab 파일 생성
   - Keytab 파일에 저장된 키(비밀번호)는 랜덤하게 생성된다
   - 사용자가 키(비밀번호)를 변경할 수 없다
2. LDAP 에 계정 foo 추가
   - 키보드로 입력 가능한 비밀번호를 사용한다.
3. hdfs:///user/foo 홈 디렉터리 생성

사용자에게 Keytab 파일과 LDAP 비밀번호를 알려준다

4.1.2 LDAP 비밀번호 변경하기

DEVIEW 2019

LDAP 을 연동한 Web-UI에서 비밀번호 변경은 되지 않는다

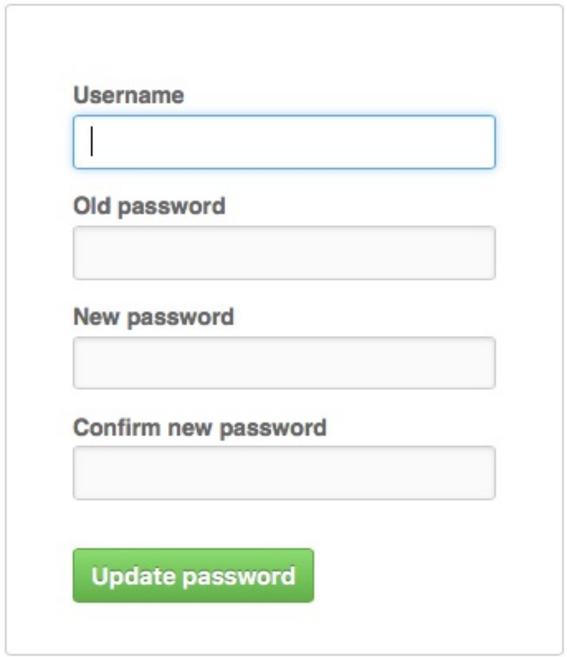
LDAP 을 연동한 호스트에서 passwd 명령으로 변경 가능

- LDAP 계정/비밀번호로 ssh 접속 가능하게 설정할 경우

비 개발자는 쉽지 않은 일이다

사용자 편의를 위해서 비밀번호 변경 Web-UI 운영

- https://github.com/jirutka/ldap-passwd-webui
- MIT license



4.2 Hadoop CLI

모든 Hadoop 관련 커맨드를 사용하기 전에 커버러스 인증이 필수
MapReduce 작업을 제출할 때에도 커버러스 인증을 해야 한다
배치작업일 경우 Keytab 파일을 사용해서 사람의 개입 없이 인증할 수 있다

4.2.1 커버러스 유틸리티 사용

DEVIEW 2019

커맨드	설명
kinit	커버러스 인증 (로그인)
klist	로컬에 저장된 티켓목록 조회
kpasswd	비밀번호 변경
kdestroy	로컬에 저장된 티켓 삭제 (로그아웃)

4.2.2 커버러스 유틸리티 사용: 비밀번호

DEVIEW 2019

커맨드	설명
kinit	커버러스 인증 (로그인)
klist	로컬에 저장된 티켓목록 조회
kpasswd	비밀번호 변경
kdestroy	로컬에 저장된 티켓 삭제 (로그아웃)

```
$ kinit magnum@X.NAVER.COM
Password for magnum@X.NAVER.COM:

$ klist -5
Ticket cache: FILE:/tmp/krb5cc_10000
Default principal: magnum@X.NAVER.COM

Valid starting     Expires            Service principal
10/18/19 14:51:50  10/19/19 14:51:48  krbtgt/X.NAVER.COM@X.NAVER.COM
        renew until 10/25/19 14:51:48

$ kdestroy

$ klist -5
klist: No credentials cache found (ticket cache FILE:/tmp/krb5cc_10000)
```

키보드로 비밀번호 입력

4.2.3 커버러스 유틸리티 사용: Keytab

DEVIEW 2019

커맨드	설명
kinit	커버러스 인증 (로그인)
klist	로컬에 저장된 티켓목록 조회
kpasswd	비밀번호 변경
kdestroy	로컬에 저장된 티켓 삭제 (로그아웃)

```
$ kinit -kt foo.keytab foo@X.NAVER.COM

$ klist -5
Ticket cache: FILE:/tmp/krb5cc_10000
Default principal: foo@X.NAVER.COM

Valid starting     Expires            Service principal
10/18/19 14:53:50  10/19/19 14:53:48  krbtgt/X.NAVER.COM@X.NAVER.COM
        renew until 10/25/19 14:53:48

$ kdestroy

$ klist -5
klist: No credentials cache found (ticket cache FILE:/tmp/krb5cc_10000)
```

Keytab 파일을 사용하면 사람의 개입 없이 인증이 가능하다

4.2.4 Hadoop CLI: hdfs 커맨드 실패

DEVIEW 2019

커버러스 인증을 하지 않아서 hdfs 커맨드 실패

```
$ klist -5
klist: No credentials cache found (ticket cache FILE:/tmp/krb5cc_10000)
$
$ hdfs dfs -ls /tmp/
19/10/18 15:15:27 WARN ipc.Client: Exception encountered while connecting to the server : org.apache.
hadoop.security.AccessControlException: Client cannot authenticate via:[TOKEN, KERBEROS]
19/10/18 15:15:27 WARN ipc.Client: Exception encountered while connecting to the server : org.apache.
hadoop.security.AccessControlException: Client cannot authenticate via:[TOKEN, KERBEROS]
```

4.2.5 Hadoop CLI: hdfs 커맨드 성공

커버러스 인증 후 hdfs 커맨드 성공

```
$ kinit -kt foo.keytab foo@X.NAVER.COM

$ klist -5
Ticket cache: FILE:/tmp/krb5cc_10000
Default principal: foo@X.NAVER.COM

Valid starting     Expires            Service principal
10/18/19 14:53:50  10/19/19 14:53:48  krbtgt/X.NAVER.COM@X.NAVER.COM
        renew until 10/25/19 14:53:48

$ hdfs dfs -ls /tmp/

Found 8 items
drwxr-xr-x   - admin              hdfs          0 2019-02-01 16:48 /tmp/aa
drwx------   - ambari-qa          hdfs          0 2019-01-27 16:26 /tmp/ambari-qa
drwxr-xr-x   - hdfs               hdfs          0 2019-01-22 19:51 /tmp/entity-file-history
drwx-wx-wx   - hive               hdfs          0 2019-07-10 18:41 /tmp/hive
-rw-r--r--   3 hdfs               hdfs       1024 2019-01-27 17:46 /tmp/id7e0a31ae_date462719
```

4.2.5 Hadoop CLI: Delegation token

hdfs fetchdt 커맨드로 Delegation token 발급

```
$ kinit -kt foo.keytab foo@X.NAVER.COM
$ hdfs fetchdt token_file
```

"token_file" 파일명으로 Delegation Token 발급

```
19/10/22 15:06:10 INFO hdfs.DFSClient: Created token for foo: HDFS_DELEGATION_TOKEN owner=foo@X.NAVER
.COM, renewer=, realUser=, issueDate=1571724370749, maxDate=1572588370749, sequenceNumber=233498, mas
terKeyId=630 on ha-hdfs:abc

$ kdestroy
$ klist -5
klist: No credentials cache found (ticket cache FILE:/tmp/krb5cc_1000)

$ export HADOOP_TOKEN_FILE_LOCATION=token_file
```

커버러스 로그아웃

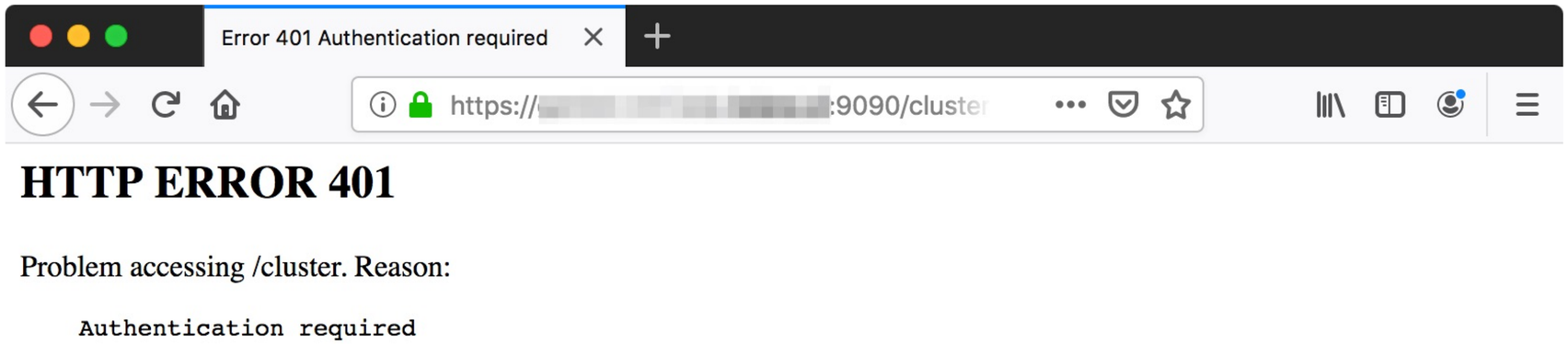
환경변수로 Delegation Token 파일 사용

```
$ hdfs dfs -ls /tmp/
Found 8 items
drwxr-xr-x   - admin          hdfs          0 2019-02-01 16:48 /tmp/aa
drwx------   - ambari-qa      hdfs          0 2019-01-27 16:26 /tmp/ambari-qa
drwxr-xr-x   - hdfs           hdfs          0 2019-01-22 19:51 /tmp/entity-file-history
drwx-wx-wx   - hive           hdfs          0 2019-07-10 18:41 /tmp/hive
```

4.3 SPNEGO 인증이 적용된 Web-UI 접속

DEVIEW 2019

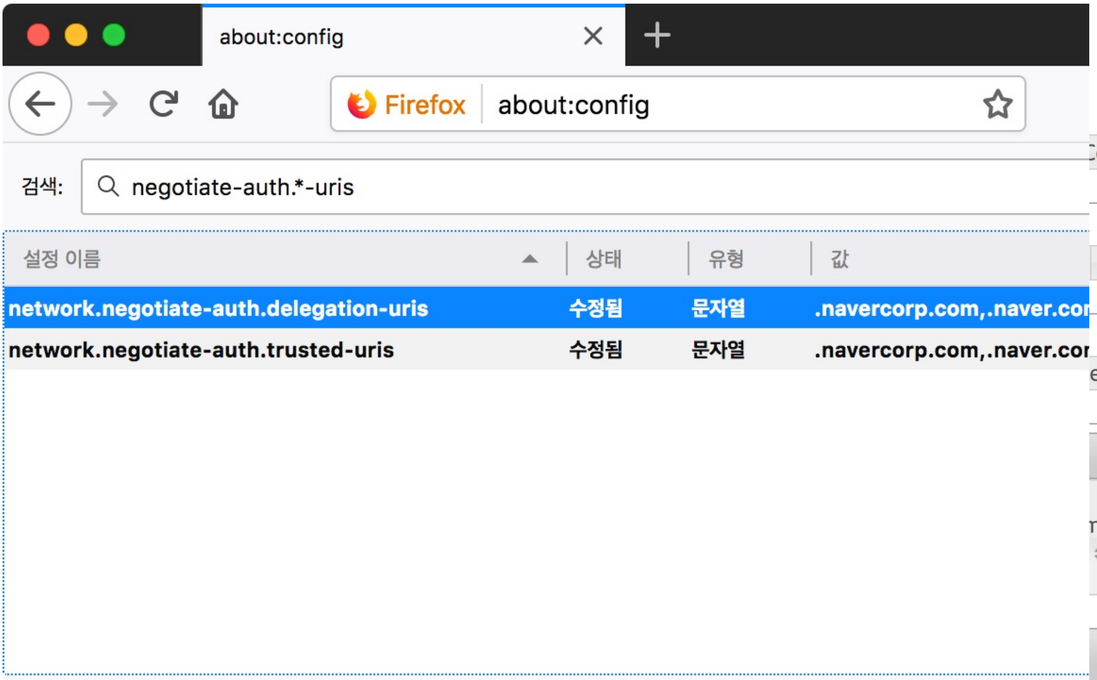
웹브라우저로 리소스매니저, 네임노드 Web-UI에 접속할 때에도 커버러스 인증이 필요하다
인증을 하지 않을 경우 401 에러가 반환된다
Web-UI 에 커버러스 인증을 위한 로그인 창은 없다

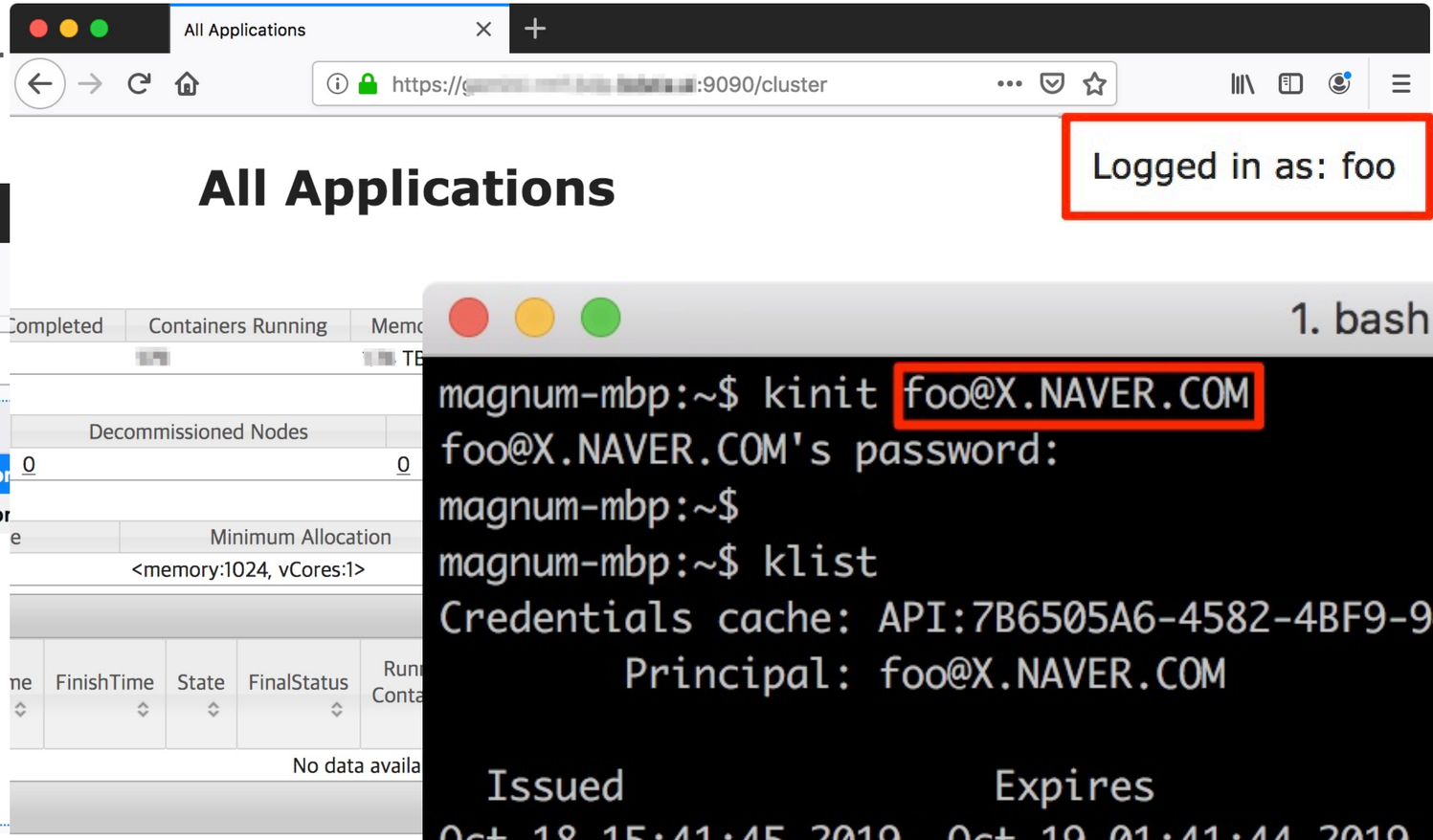


4.3.1 SPNEGO Web-UI 접속: macOS

DEVIEW 2019

1. 하둡 클러스터의 /etc/krb5.conf 를 macOS 의 /etc/krb5.conf 에 복사한다
2. 브라우저에 관련 설정을 한다 (Appendix 참고)
3. 셸에서 kinit 명령으로 커버러스 인증
4. Web-UI에 접속한다

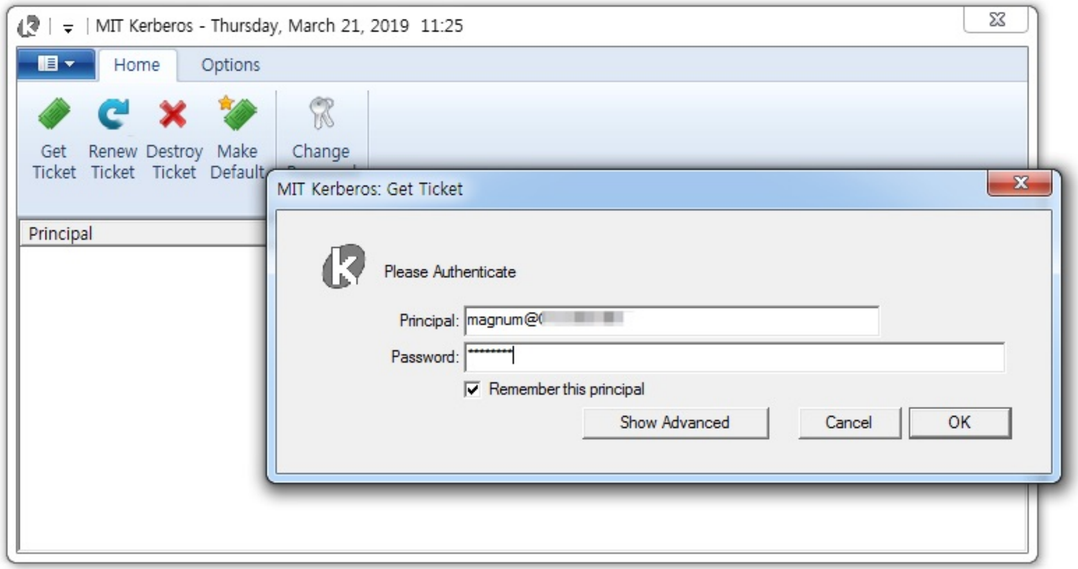


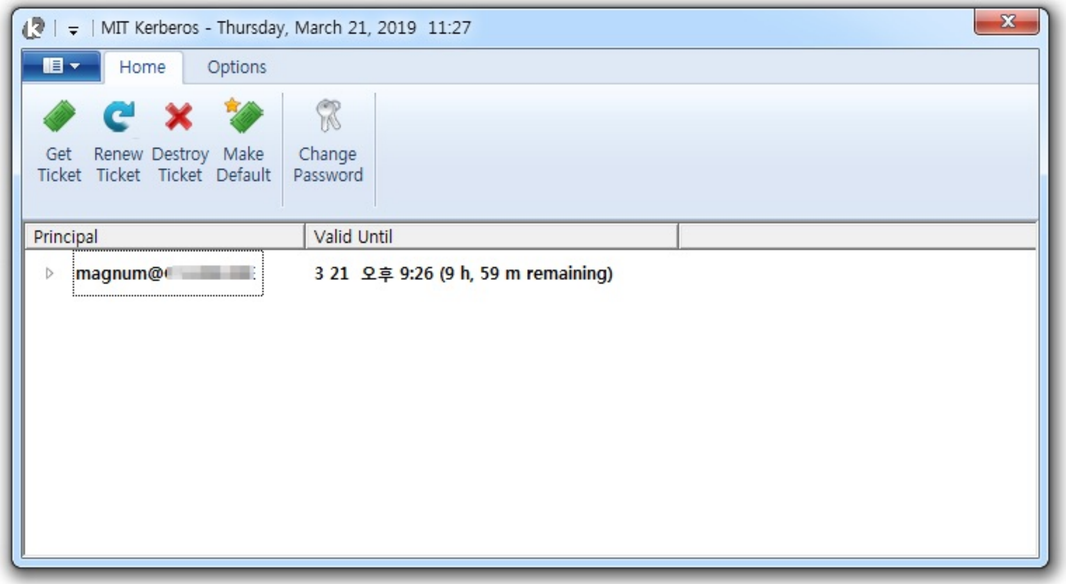


4.3.2 SPNEGO Web-UI 접속: Windows

DEVIEW 2019

1. MIT Kerberos for Windows 4.1을 설치 (https://web.mit.edu/kerberos/dist/)
2. 하둡 클러스터의 /etc/krb5.conf 를 C:₩ProgramData₩MIT₩Kerberos5₩krb5.ini 복사
3. 브라우저에 관련 설정을 한다 (Appendix 참고)
4. Windows 커버러스 앱에서 인증 후 Web-UI에 접속한다
   - Keytab 을 사용할 경우 커맨드 창에서 kinit 명령을 사용한다

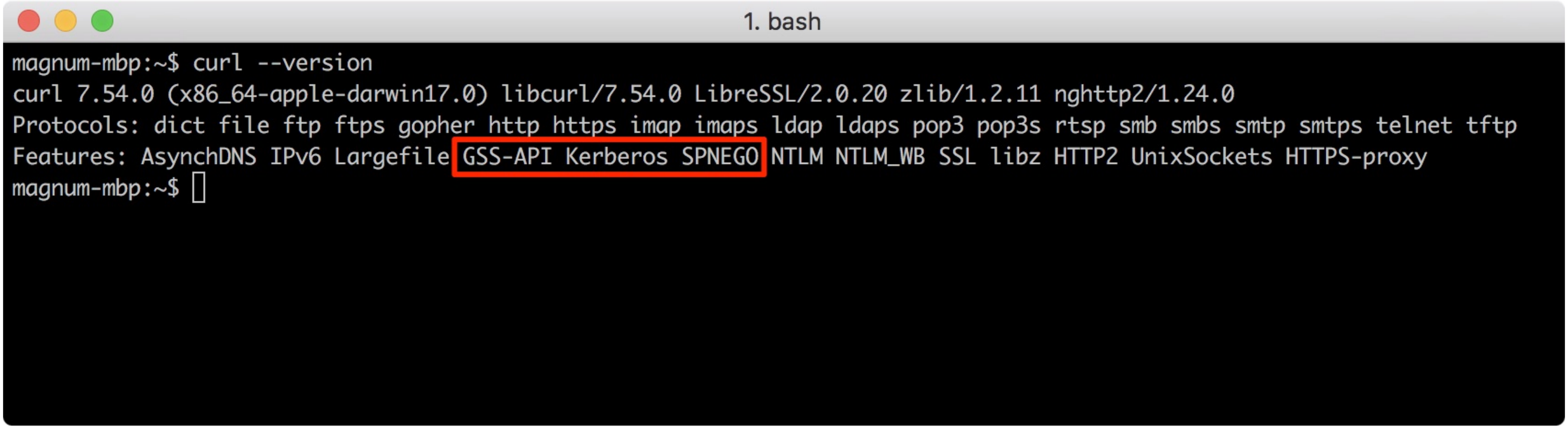




4.3.3 SPNEGO REST API 접속: curl

DEVIEW 2019

하둡에서 지원하는 REST API도 커버러스 인증이 필요하다
curl 빌드 방법에 따라선 SPNEGO 를 지원하지 않을 수 있다



4.3.3 SPNEGO REST API 접속: curl

커버러스 인증 후에 curl을 사용하고 `-u : --negotiate` 옵션을 추가한다

1. bash

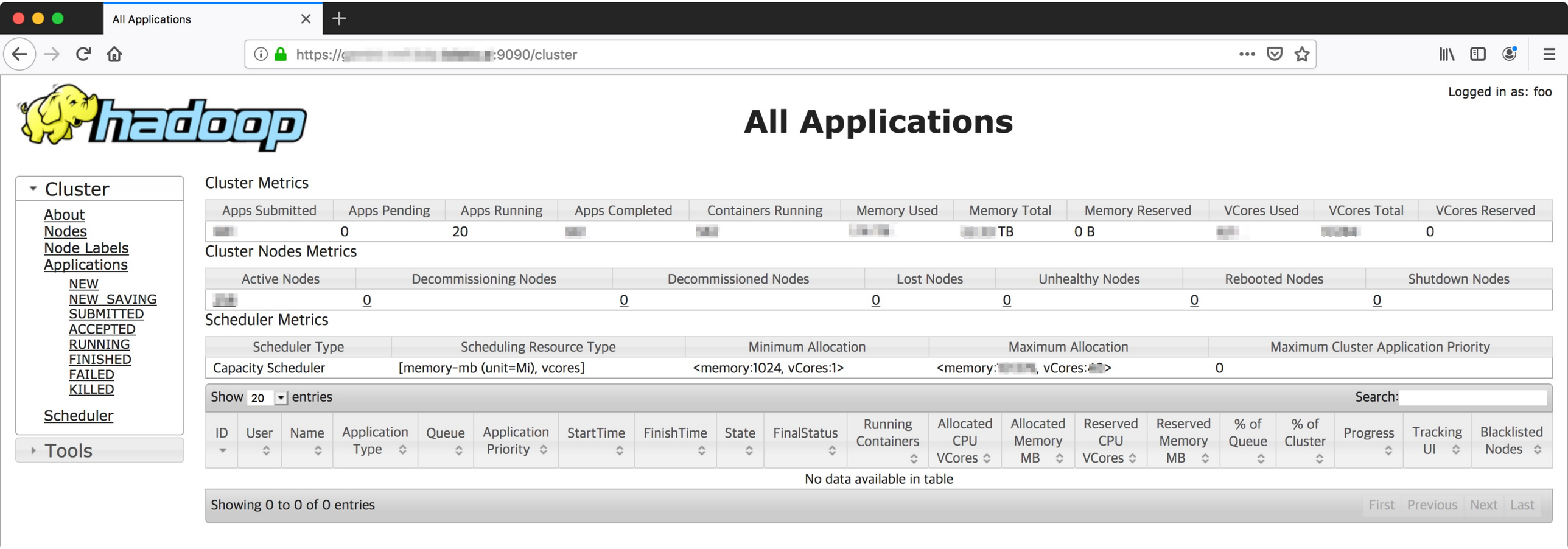
```
magnum-mbp:~$ kinit foo@X.NAVER.COM
foo@X.NAVER.COM's password:
magnum-mbp:~$
magnum-mbp:~$ klist
Credentials cache: API:7B6505A6-4582-4BF9-945C-30699E957F40
        Principal: foo@X.NAVER.COM

  Issued                Expires               Principal
Oct 18 16:03:04 2019  Oct 19 02:03:02 2019  krbtgt/X.NAVER.COM@X.NAVER.COM
magnum-mbp:~$
magnum-mbp:~$ curl -s -u : --negotiate https://[redacted]:9090/ws/v1/cluster | python -m json.tool
{
    "clusterInfo": {
        "haState": "ACTIVE",
        "haZooKeeperConnectionState": "CONNECTED",
        "hadoopBuildVersion": "3.1.1.3.1.2-7 from [redacted]",
        "hadoopVersion": "3.1.1.3.1.2-7",
        "hadoopVersionBuiltOn": "2019-08-12T02:27Z",
        "id": 1571131888165,
```

4.4 Job ACL 활용

DEVIEW 2019

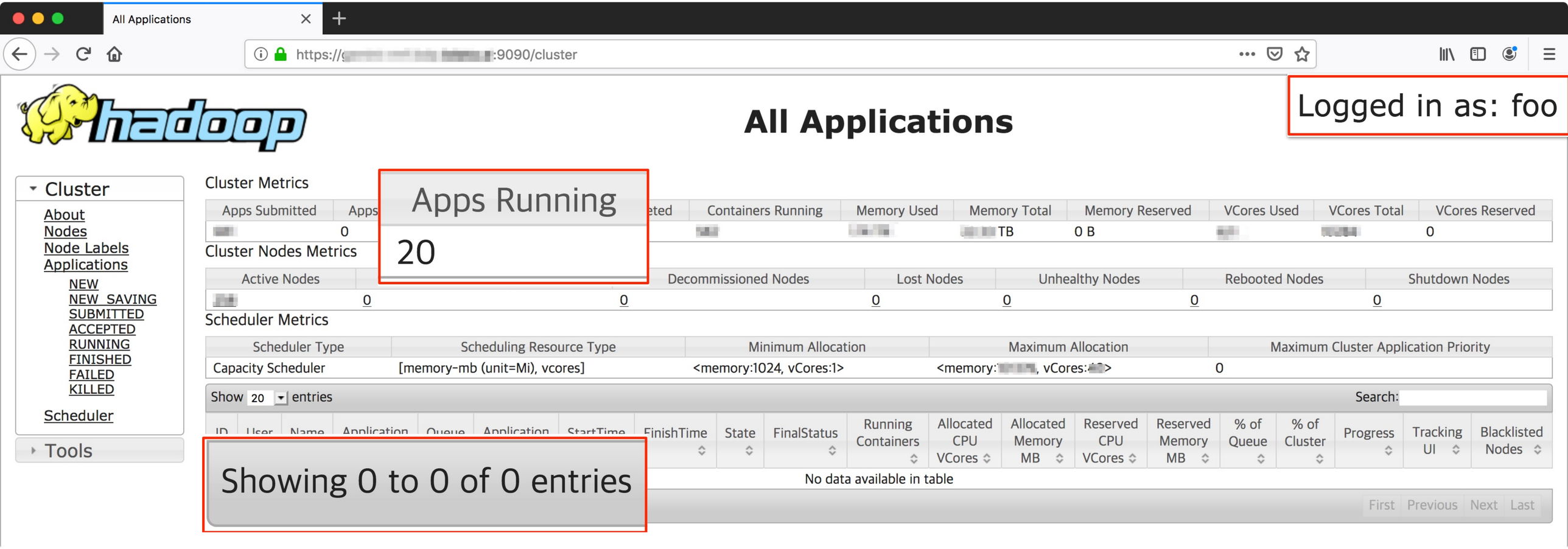
리소스매니저와 히스토리서버에서 자신의 작업만 조회할 수 있고 다른 사람의 작업은 조회할 수 없다



4.4 Job ACL 활용

DEVIEW 2019

리소스매니저와 히스토리서버에서 자신의 작업만 조회할 수 있고 다른 사람의 작업은 조회할 수 없다



4.4.1 Job ACL 활용 예

Job ACL을 사용하면 나의 작업을 다른 사용자에게 보여줄 수 있다

- 사용 예
  * 옆자리 동료에게 실패한 작업 분석 문의
  * 배치작업의 담당자가 다수일 때 모든 담당자가 해당 배치작업을 조회할 수 있어야 한다

ACL 종류	설명
view-acl	작업을 볼 수 있는 권한 (작업 조회)
modify-acl	작업을 수정할 수 있는 권한 (작업 죽이기)

MapReduce, Spark 같은 프레임워크마다 설정 방법이 마련되어 있다

4.4.2 Job ACL 설정

클라이언트 설정이다. 작업명세에 사용자가 직접 설정할 수 있다

	view-acl	modify-acl*
MapReduce	mapreduce.job.acl-view-job	mapreduce.job.acl-modify-job
Spark	spark.ui.view.acls	spark.modify.acls
Tez	tez.am.view-acls	tez.am.modify-acls

* view-acl 없이 modify-acl 권한만 있으면 해당 작업조회(view)는 할 수 없지만
Application ID를 알면 작업을 죽일 수(modify) 있다

4.5 배치작업의 최대 실행 시간 제한

DEVIEW
2019

MapReduce, Tez, Spark 작업을 7일(운영자가 설정 가능) 초과해서 실행시킬 수 없다
7일이 넘어가면 Delegation token이 무효화 되어서
네임노드 접속이 실패하면서 작업이 실패한다

oozie shell-action으로 종료되지 않는 작업을 실행하는 경우가 있는데 시큐어 환경에서는 할 수 없다

4.5.1 Spark Streaming 은?

Spark Streaming 작업은 무한히 실행되어야 한다
이런 경우에는 아래 설정을 사용해서 작업제출할때 Keytab 파일을 제공하면 무한히 실행시킬 수 있다

Property Name	Default	Meaning
spark.yarn.keytab	(none)	The full path to the file that contains the keytab for the principal specified above. This keytab will be copied to the node running the YARN Application Master via the YARN Distributed Cache, and will be used for renewing the login tickets and the delegation tokens periodically. Equivalent to the --keytab command line argument.
spark.yarn.principal	(none)	Principal to be used to login to KDC, while running on secure clusters. Equivalent to the --principal command line argument. (Works also with the "local" master.)

https://spark.apache.org/docs/latest/running-on-yarn.html#yarn-specific-kerberos-configuration

4.6 네트워크 암호화

DEVIEW
2019

Ambari가 제안하는 설정의 기본동작은 네트워크 암호화를 하지 않는다
네트워크 암호화는 암호화/복호화 오버헤드로 인한 성능 저하가 있고
모든 사용자가 네트워크 암호화가 필요한 건 아니어서
일괄적으로 모든 사용자의 환경을 암호화하는 건 바람직하지 않을 수 있다
하둡에서 사용자의 클라이언트 설정으로 암호화 여부를 선택 할 수 있다

4.6.1 프로토콜 종류

하둡에서는 크게 3가지 프로토콜을 사용한다

프로토콜	설명
Hadoop RPC	Hadoop API 에서 사용 hdfs dfs -ls 명령이 Hadoop RPC (네임노드 RPC) 호출
Direct TCP/IP	클라이언트와 데이터노드 사이에 통신(읽기/쓰기)할 때 사용
HTTP	HTTP REST API, MapReduce suffle FSImage Operation, webhdfs

https://blog.cloudera.com/how-to-set-up-a-hadoop-cluster-with-network-encryption/

4.6.2 Hadoop RPC 암호화

DEVIEW 2019

서버 core-site.xml 설정	hadoop.rpc.protection=authentication,privacy
설정의미	서버에서 Quality of Protection(QOP) 을 authentication, privacy 2가지 지원 클라이언트에서 사용할 QOP 를 선택한다
통신 채널 암호화를 위한 클라이언트 core-site.xml 설정*	hadoop.rpc.protection=privacy

* 클라이언트 설정도 , 구분자로 나열할 수 있다. 여러가지를 설정할 경우 서버와 일치하는 QOP 중에 클라이언트 설정에서 먼저 출현한 QOP를 사용한다
클라이언트에 hadoop.rpc.protection=privacy,authentication 으로 설정할 경우 서버에서 지원하는 QOP와 모두 일치한다. 그러면 먼저 출현한 privacy를 사용한다
서버와 클라이언트의 QOP가 일치하는 게 없으면 통신은 실패한다

4.6.3 데이터노드 통신 암호화

DEVIEW 2019

서버 hdfs-site.xml 설정	dfs.data.transfer.protection=authentication,privacy
설정의미	서버에서 Quality of Protection(QOP) 을 authentication, privacy 2가지 지원 클라이언트에서 사용할 QOP 를 선택한다
통신 채널 암호화를 위한 클라이언트 hdfs-site.xml 설정*	dfs.data.transfer.protection=privacy

* 클라이언트 설정도 , 구분자로 나열할 수 있다. 여러가지를 설정할 경우 서버와 일치하는 QOP 중에 클라이언트 설정에서 먼저 출현한 QOP를 사용한다
클라이언트에 dfs.data.transfer.protection=privacy,authentication 으로 설정할 경우 서버에서 지원하는 QOP와 모두 일치한다. 그러면 먼저 출현한 privacy를 사용한다
서버와 클라이언트의 QOP가 일치하는 게 없으면 통신은 실패한다

4.6.4 HTTP 암호화

클러스터 운영자가 https를 사용할 수 있도록 TLS (SSL) 를 적용해야 한다
사용자는 http, https를 선택적으로 사용할 수 있다

hdfs dfs -ls webhdfs://ns/tmp
hdfs dfs -ls swebhdfs://ns/tmp

5.운영을 편하게 하기 위한 팁

5.1 사용자 계정 관리

항목	개인계정	서비스계정	운영자계정
용도	테스트/개발	서비스운영	클러스터 관리
보유 가능 계정 개수	1개	n개 (제한없음)	1개 (운영자에게만발급)
그룹명*	users	services	hadoop-admins
서비스용 YARN 큐에 작업 제출**	불가능	가능	가능
개발용 YARN 큐에 작업 제출**	가능	가능	가능
커버러스 크리덴셜	비밀번호	Keytab 파일	비밀번호

* 리눅스 OS의 User, Group 개념의 그룹이다. LDAP 으로 관리한다

** YARN Queue ACLs 에 그룹명을 명시해서 제출 가능 여부를 제어한다

5.1 사용자 계정 관리: 커버러스 크리덴셜

항목	개인계정	서비스계정	운영자계정
커버러스 크리덴셜	비밀번호	Keytab 파일	비밀번호

개인 PC 에서도 커버러스 인증이 필요한데 개인 PC 에서 Keytab 파일 사용은 매우 번거롭다
개발/테스트용도인 개인계정은 키보드로 입력할 수 있는 비밀번호를 사용한다

- 필요하다면 사용자가 직접 개인계정의 Keytab 파일을 만들 수 있다

서비스계정 작업의 Job ACL에 개인계정을 등록하면
리소스매니저 Web-UI에서 개인계정으로 작업을 조회/수정 할 수 있다

5.2 다수의 운영자 계정관리

각 서버의 계정은 admin 권한을 가지고 있다
클러스터 운영자가 1명일 경우 각 서버의 계정을 사용해도 크게 불편함은 없다

	admin 계정
HDFS	hdfs
YARN	yarn
OOZIE	oozie
Ambari	admin
Ranger	admin

5.2 다수의 운영자 계정관리

클러스터 운영자가 다수일 경우 각 서버의 계정을 사용할 경우 아래 같은 불편함이 있다.

1. 운영자중에 누가 작업했는지 audit log에 명확히 기록되지 않는다
2. YARN의 Keytab 을 운영자의 PC에 복사해야 하는데 번거롭다
   - 사용자 작업을 디버깅하려면 리소스매니저에 admin 권한이 있는 yarn 유저로 접속해야 한다
3. Ambari/Ranger admin 비밀번호를 다수의 운영자끼리 공유해야 한다
   - 모두가 외워야 하니 어려운 비밀번호를 사용하지 않을 가능성이 높다

5.2 다수의 운영자 계정관리

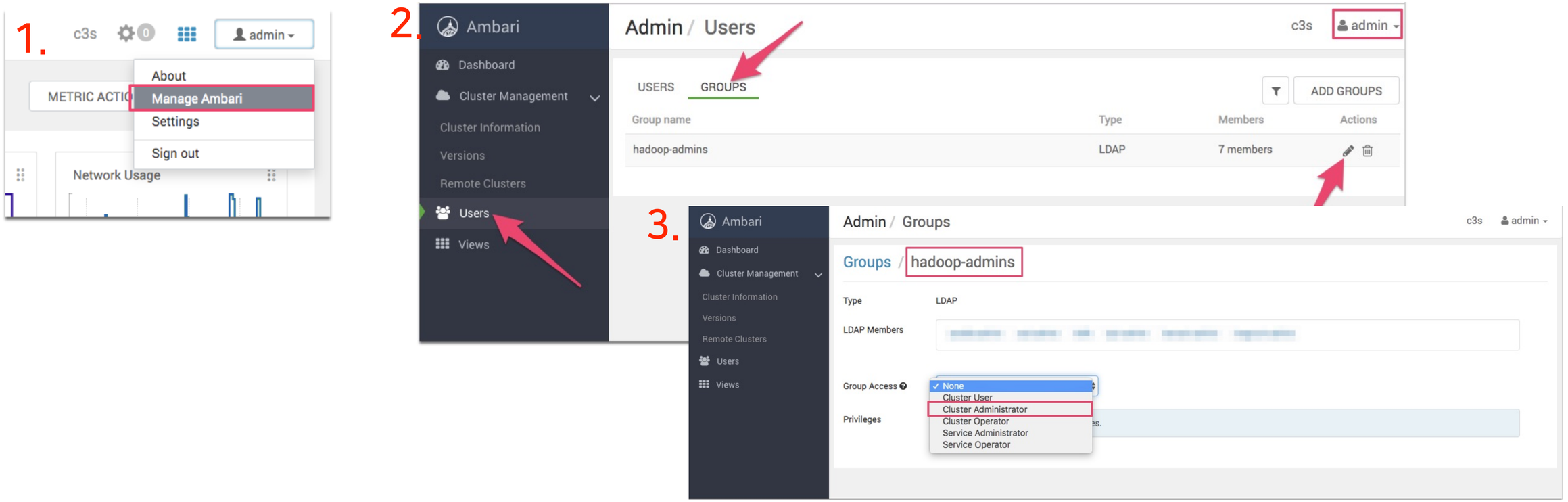
운영자 그룹 hadoop-admins 를 만들고 각 서버에서 해당 그룹에 속한 사용자는 admin 권한을 가지도록 설정

hdfs-site.xml	dfs.permissions.superusergroup=hadoop-admins dfs.cluster.administrators= hdfs,hadoop-admins
yarn-site.xml	yarn.admin.acl=yarn hadoop-admins
mapred-site.xml	mapreduce.cluster.administrators= hadoop,hadoop-admins
oozie-site.xml	oozie.service.AuthorizationService.admin.groups=hadoop-admins
spark-defaults.conf	spark.admin.acls.groups=hadoop-admins spark.history.ui.admin.acls.groups=hadoop-admins spark.ui.view.acls.groups=hadoop-admins
ranger-ugsync-site.xml	ranger.usersync.group.based.role.assignment.rules=ROLE_SYS_ADMIN:g:hadoop-admins

5.2 다수의 운영자 계정관리

DEVIEW 2019

Ambari 는 LDAP 연동후에 아래 절차로 진행한다

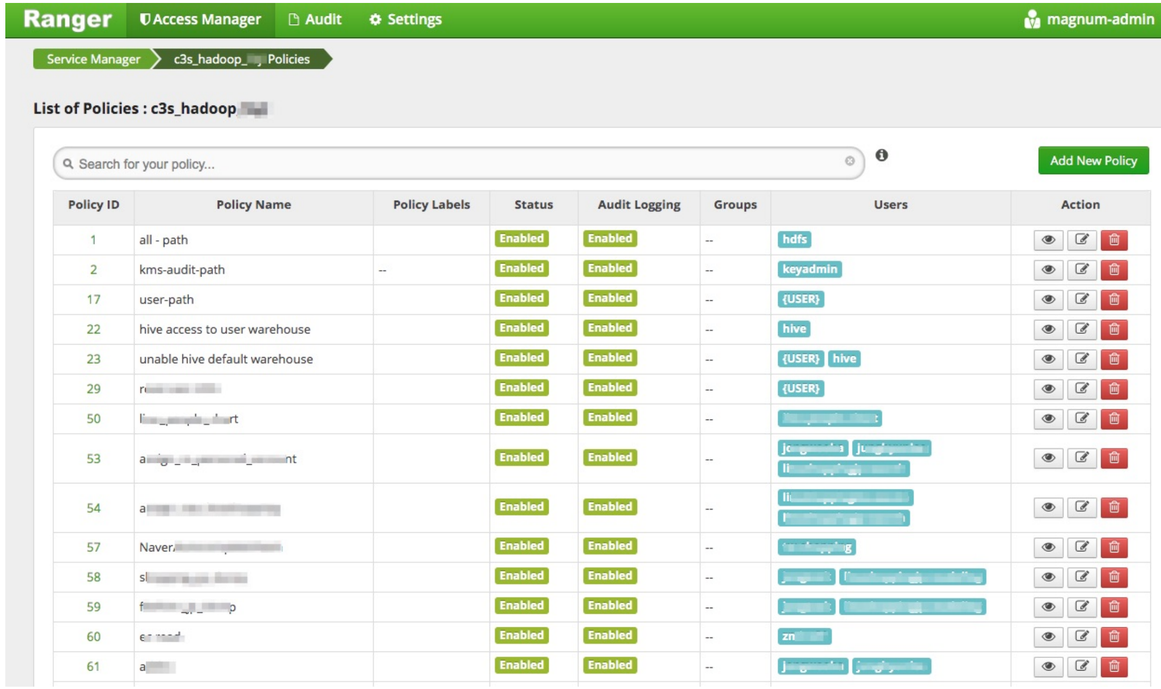


5.3 Apache Ranger 로 ACL 관리

DEVIEW 2019

Ranger Web-UI 가 있어서 운영자/사용자 둘 다 설정한 ACL을 일목요연하게 관리하기 쉽다

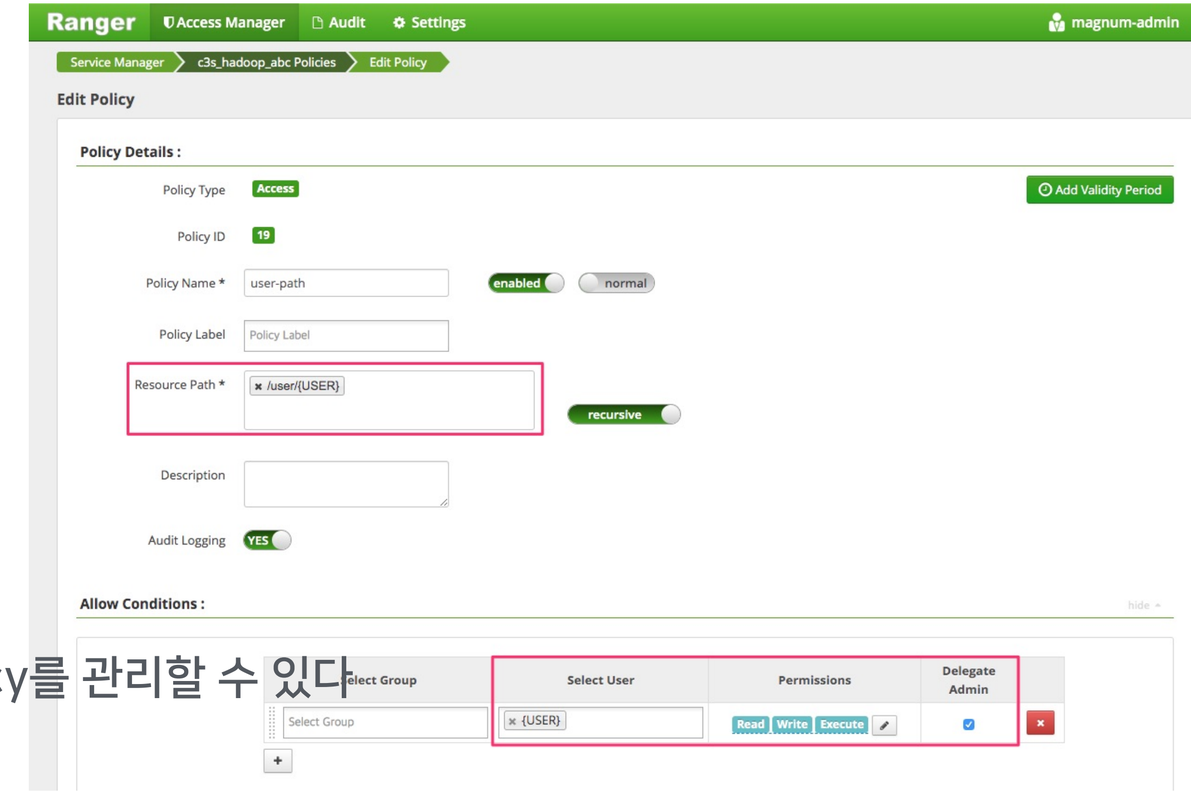
사용자가 Ranger로 ACL을 관리할 수 있게 하려면 운영자가 **사용자에게 관리 권한을 위임**해야 한다



5.3.1 Ranger HDFS Policy 설정

운영자 권한으로 아래같이 설정한다 ({USER}는 Ranger 예약어)

- Resource Path: /user/{USER}
  - recursive=on
- Allow Conditions
  - Select User: {USER}
  - Permissions: Read, Write, Execute
  - Delegate Admin: true



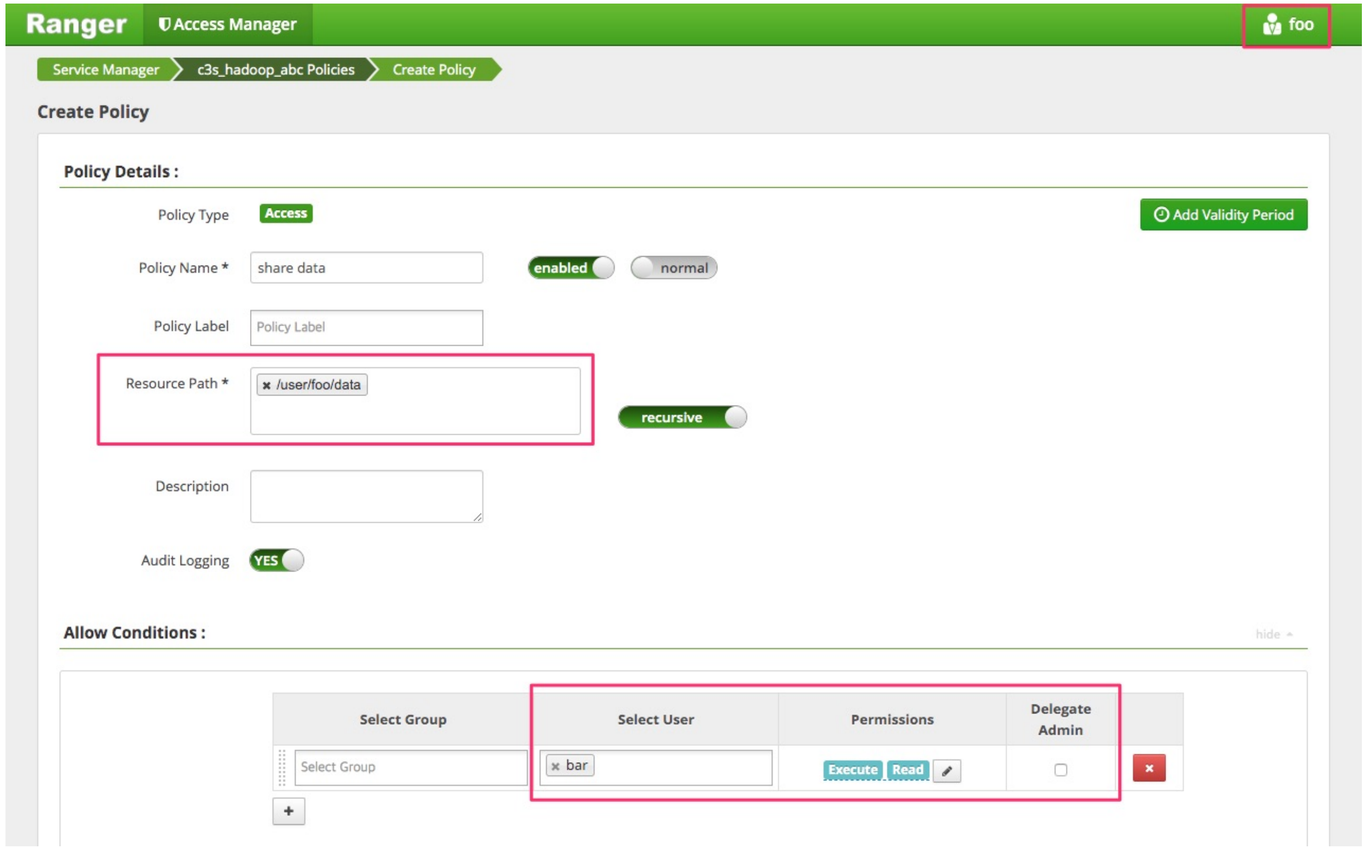
사용자는 자신의 홈 디렉터리 하위 경로에 대해 모든 권한을 가지며
Admin 권한도 위임받아서 자신 홈 디렉터리에 대해서 Ranger Policy를 관리할 수 있다

신규사용자가 추가될 때마다 Policy를 추가할 필요가 없다

5.3.2 Ranger HDFS Policy 사용 사례

사용자 foo가 자신의 /user/foo/data 디렉터리를
사용자 bar에게 공유하려고 할 경우
사용자 foo는 아래 같은 Policy를 설정한다

Resource Path: /user/foo/data
Allow Conditions
  Select User: bar
  Permissions: read/execute
  Delegate admin: false



5.3.3 Ranger HIVE Policy 설정

운영자 권한으로 아래같이 설정한다 ({USER}는 Ranger 예약어)

[database]: {USER}__db_* [table]: * [Hive Column]: *

Allow Conditions

Select User: {USER}

Permissions: Select All

Delegate Admin: true

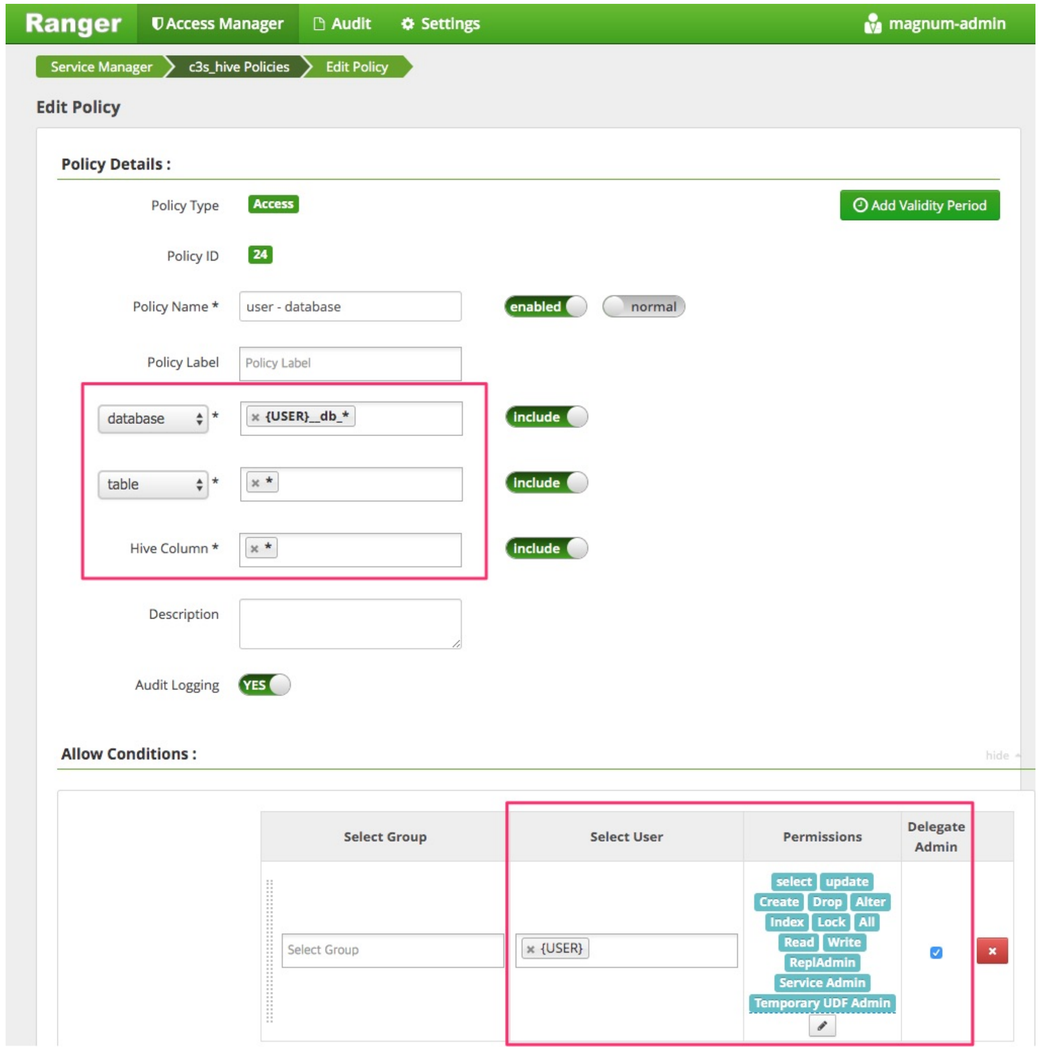
default DB는 사용할 수 없고 사용자가 DB를 직접 만들 수 있으나

DB 명은 {USER}__db_* 패턴(e.g. foo__db_bar)으로만 만들 수 있고

자신의 DB에 대해서 모든 권한을 가지며

Admin 권한도 위임받아서 자신의 DB에 대해서 Ranger Policy를 관리할 수 있다

신규사용자가 추가될 때마다 Policy를 추가할 필요가 없다



5.4 Hive DB/테이블이 사용하는 경로제한

DEVIEW
2019

5.4.1 문제

Hive DB, 테이블을 만들 경우 hive.metastore.warehouse.dir 의 하위에 디렉터리가 만들어진다
이 경로는 모든 사용자가 같이 사용하는 공용디렉터리이다

대부분의 사용자는 공용디렉터리의 파일 개수/용량에는 관심을 가지지 않는다
사용자들의 무관심과 잘못된 파티셔닝등의 이유로
파일개수가 증가하고 네임노드의 메모리 압박이 발생한다

5.4.2 운영자의 번거로움

DEVIEW
2019

주기적인 사용자와의 커뮤니케이션
- 파일 개수 좀 줄여주세요
- 사용하지 않는 오래된 데이터 삭제해주세요

매우 번거로운 일이다

네임노드 파일개수 관련: DEVIEW 2017 멀티테넌트 하둡 클러스터 운영 경험기

5.4.3 해결책 (1/2)

안정적인 클러스터 운영을 위해서 사용자의 홈 디렉터리에
파일 개수 제한(name quotas), 용량 제한(space quotas)을 한다

Hive DB, 테이블 경로를 사용자 홈 디렉터리만 사용할 수 있게 제한을 하면 파일 개수로 인한 운영자의 스트레스를 많이 줄일 수 있다

5.4.3 해결책 (2/2)

DDL 쿼리에 LOCATION을 지정할 수 있다

```
# CREATE DATABASE foo__db_bar LOCATION `hdfs://ns/tmp/foo__db_bar.db`;
```

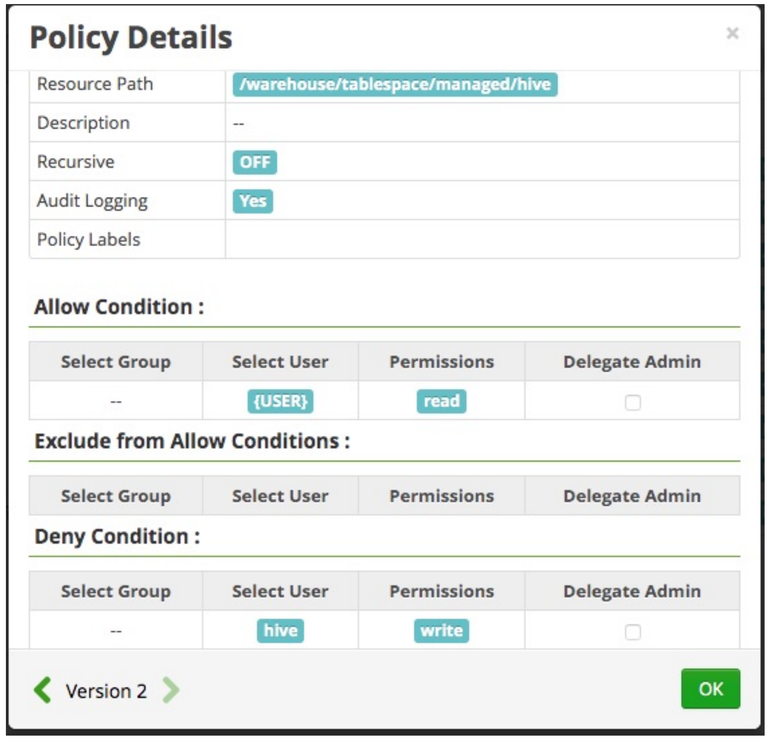
경험상 LOCATION을 지정해달라고 안내하는 건 사용자가 잊기 쉬워서 큰 효과가 없다
시스템상 LOCATION 사용을 강제해야 한다

하지만 HIVE에는 LOCATION을 강제할 방법이 마련되어 있지 않다
Ranger HDFS ACL 기능을 사용해서 LOCATION 지정을 강제할 수 있다

5.4.4 Ranger HDFS Policy (1/2)

디폴트 warehouse 경로를 사용하지 못하게 하려면 해당 경로에 hive 유저와 사용자 권한으로 디렉터리를 만들 수 없게 한다

Resource Path: /warehouse/tablespace/managed/hive
  recursive=off
Allow Conditions
  Select User: {USER}
  Permissions: read
  Delegate admin: false
Deny Conditions
  Select User: hive
  Permissions: write
  Delegate admin: false



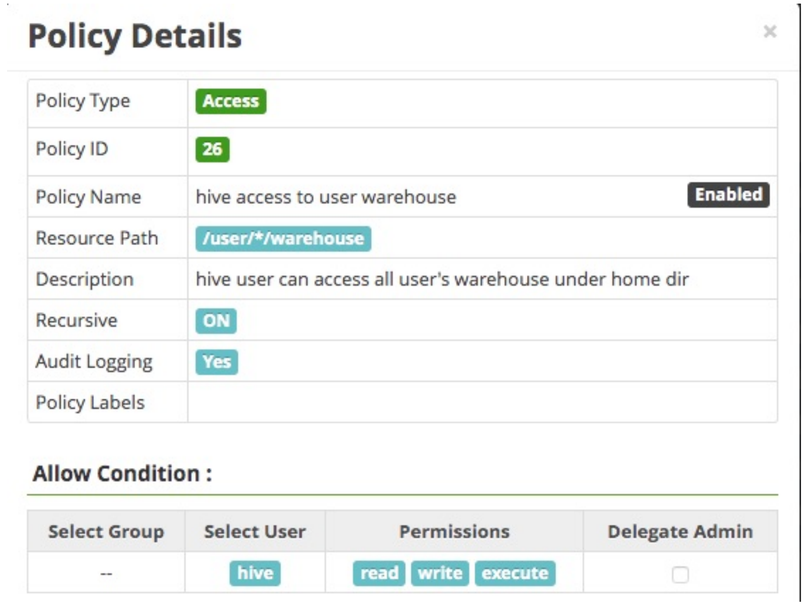
5.4.4 Ranger HDFS Policy (2/2)

DB는 홈 디렉터리의 warehouse 이하의 경로만 사용하는 규칙을 만듦

- /user/foo/warehouse/foo__db_data.db (o)
- /user/foo/hive/foo__db_data.db (x)

Hive DB가 사용할 디렉터리는 사용자 hive로 만들어지니
사용자 hive가 홈 디렉터리의 warehouse 경로에 접근할 수 있게 Policy 추가

Resoure Path: /user/*/warehouse
  recursive=on
Allow Conditions
  Select User: hive
  Permissions: read/write/execute
  Delegate admin: false



5.4.5 단점

LOCATION을 지정하지 않으면 쿼리가 실패한다. 에러메시지만 보고 LOCATION 누락을 판단하기 쉽지 않다.
사용자의 반복된 질문을 받지 않으려면 안내 문서를 잘 작성해놔야 한다

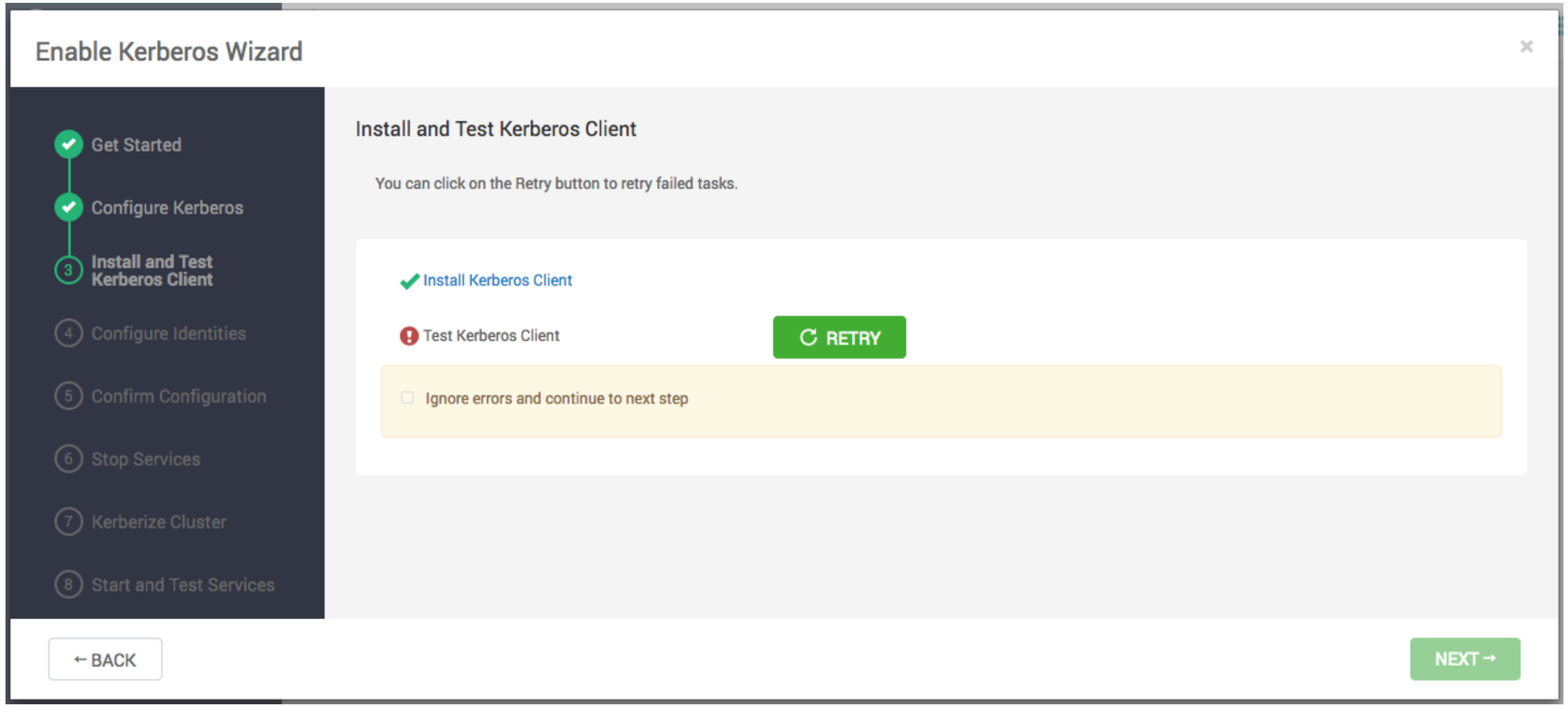
```
0: jdbc:hive2://hs2-> CREATE DATABASE foo__db_bar;
ERROR : FAILED: Execution Error, return code 1 from org.apache.hadoop.hive.ql.exec.DDLTask.
MetaException(message:java.security.AccessControlException: Permission denied: user=foo, access=WRITE,
inode="/warehouse/tablespace/managed/hive":hive:hadoop:drwx------

0: jdbc:hive2://hs2-> CREATE DATABASE foo__db_bar LOCATION '/user/foo/warehouse/foo__db_bar.db';
No rows affected (1.121 seconds)
```

DEVIEW
2019

6.트러블 슈팅

6.1 Kerberos Wizard 의 Test 실패 (1/2)

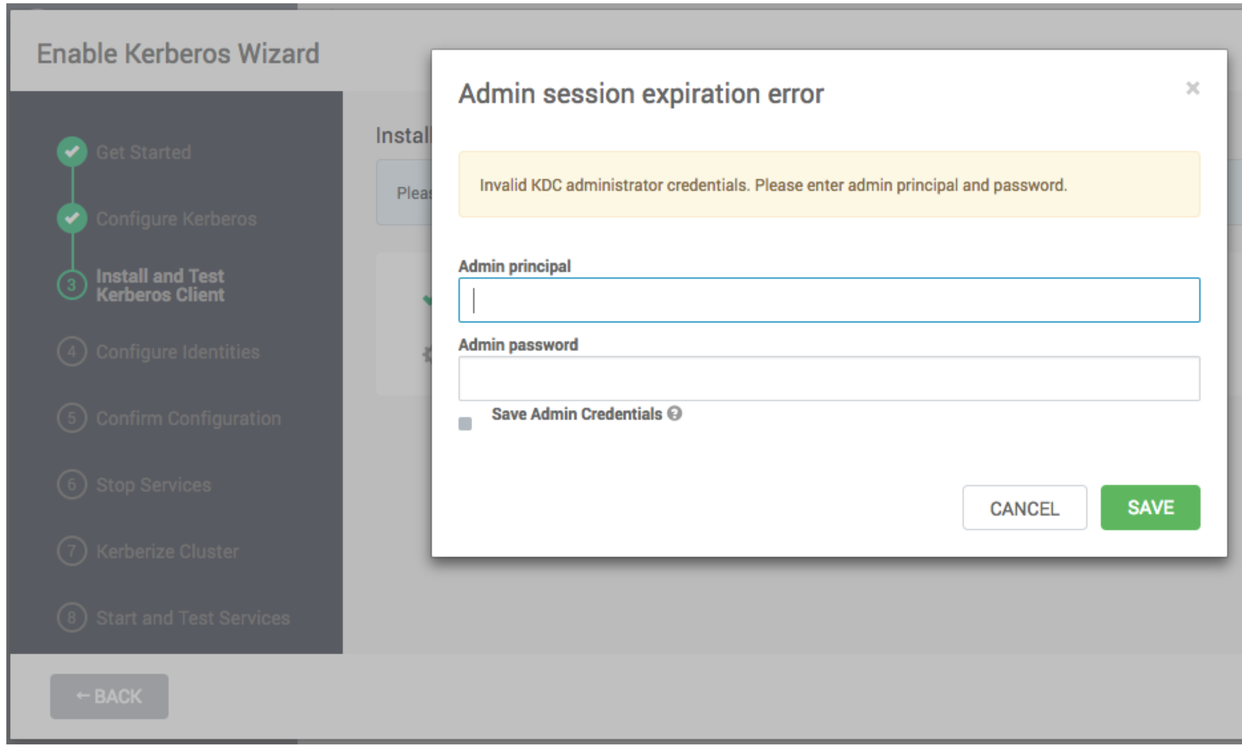


6.1 Kerberos Wizard 의 Test 실패 (2/2)

아래 단계로 확인한다

1. kerberos admin Principal 정보를 정확히 입력한다
2. ambari-server를 설치한 호스트도 ambari-agent를 설치해야 한다
3. ambari-server 로그를 확인한다. 커버러스 관련 에러 로그가 있을 것이다

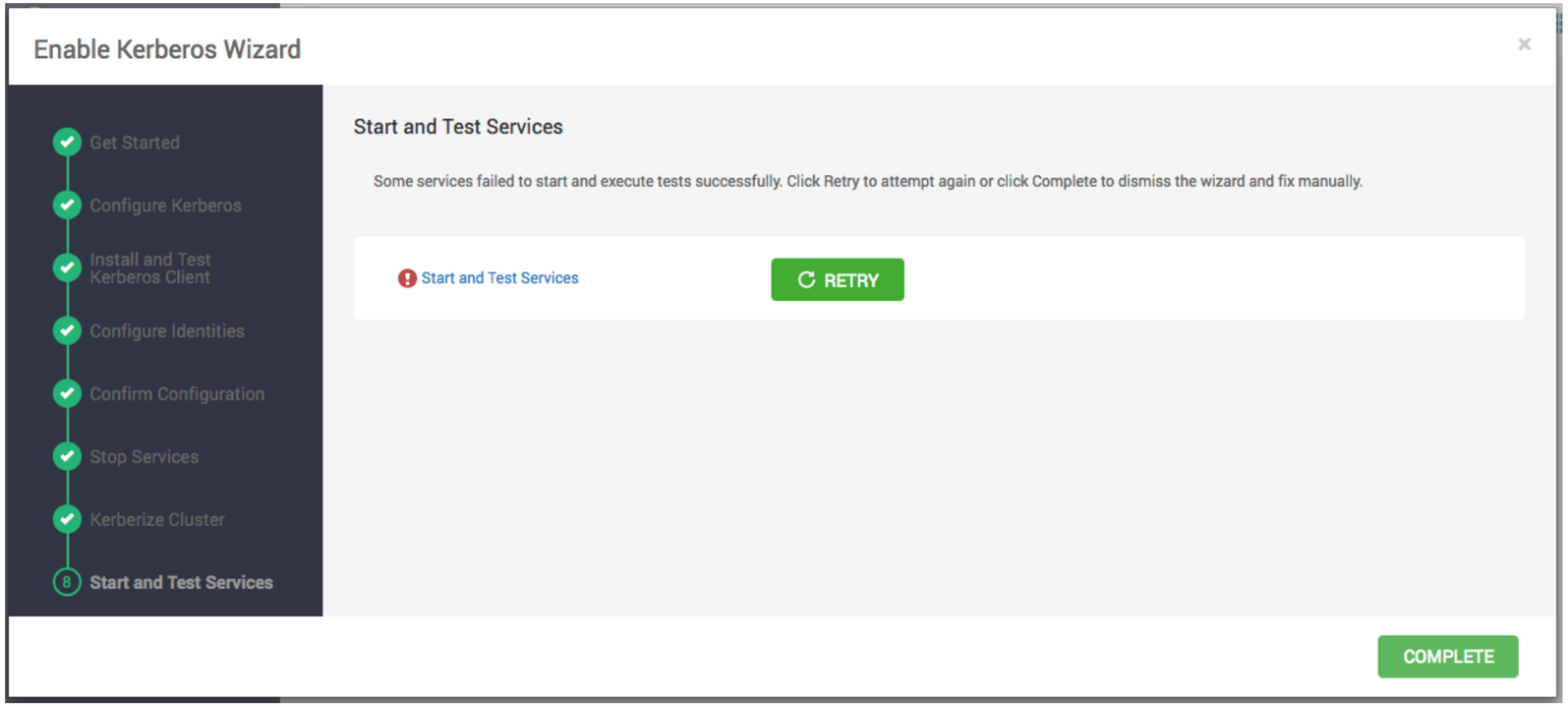
/var/log/ambari-server/ambari-server.log



6.2 Kerberos Wizard 마지막 단계 실패

DEVIEW 2019

1. Retry 를 반복하면 성공할때 도 있다
2. COMPLETE 버튼 클릭해서 Wizard 종료후에 START 실패한 서비스 디버깅 후 START 를 할 수 도 있다



6.3 Ranger 로그인후에 바로 로그아웃된다

DEVIEW 2019

상황

Ranger 서버 이중화 목적으로 2대에 설치하고 로드 밸런서를 사용했다

로드 밸런서는 round-robin 방식

원인

Ranger Web-UI는 로그인 세션 정보를 서버 간에 공유를 하지 않는다

로그인은 1번 서버에서 했는데, 다음 요청은 2번 서버에 요청하게 되어서 로그인이 튕김

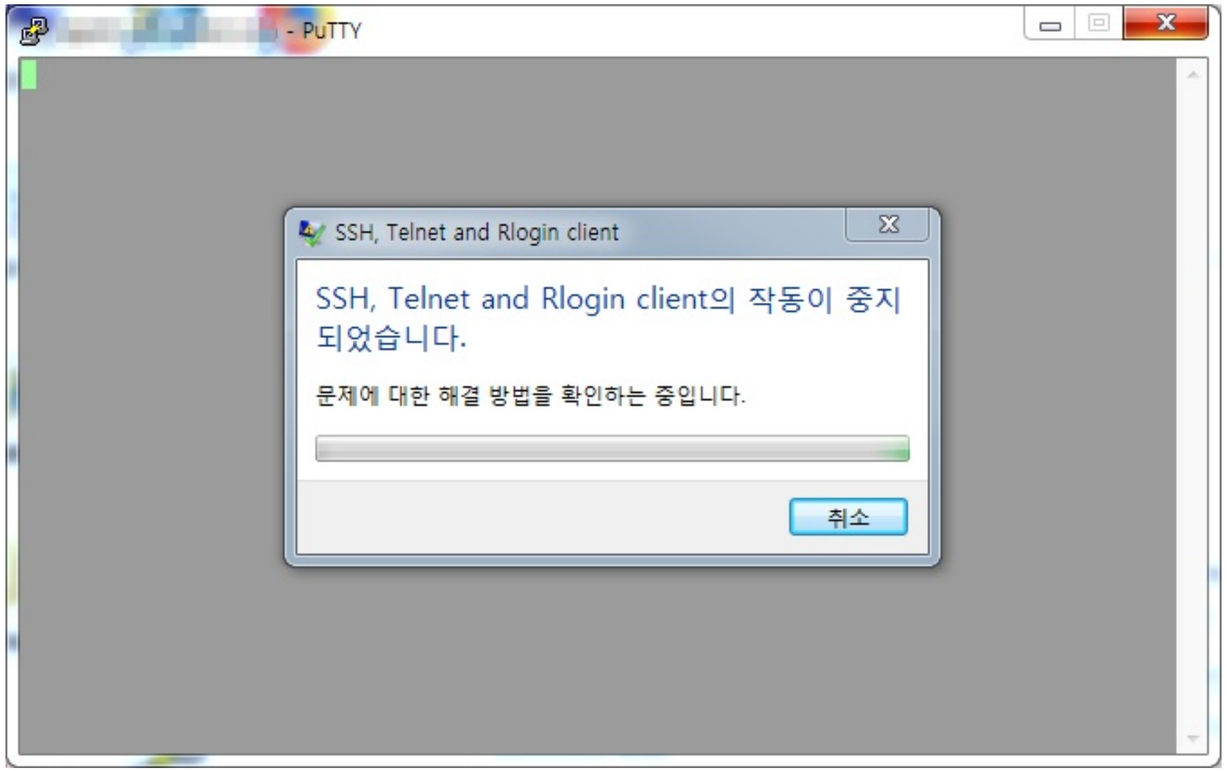
해결

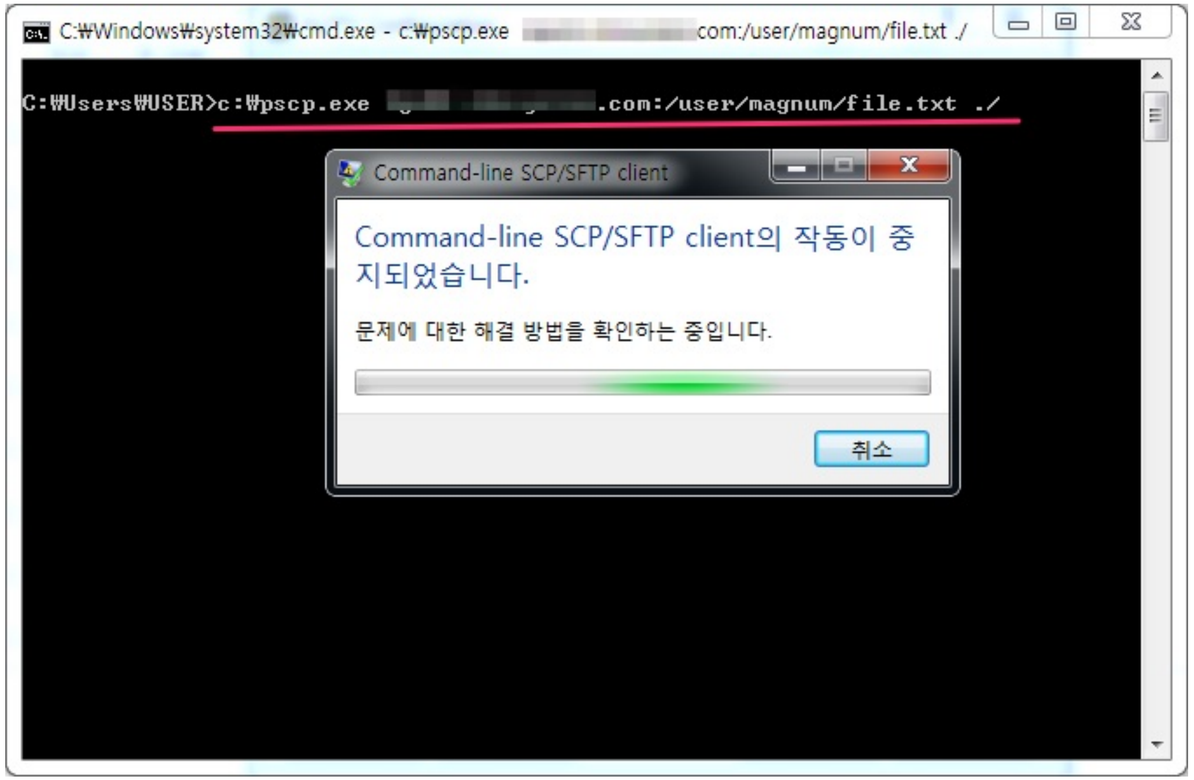
sticky session 이 지원되는 로드 밸런서를 사용한다(e.g. nginx ip_hash)

6.4 PuTTY 가 동작하지 않는다 (1/2)

DEVIEW 2019

MIT Kerberos for Windows를 설치하면 Windows PuTTY로 ssh 접속이 되지 않는다



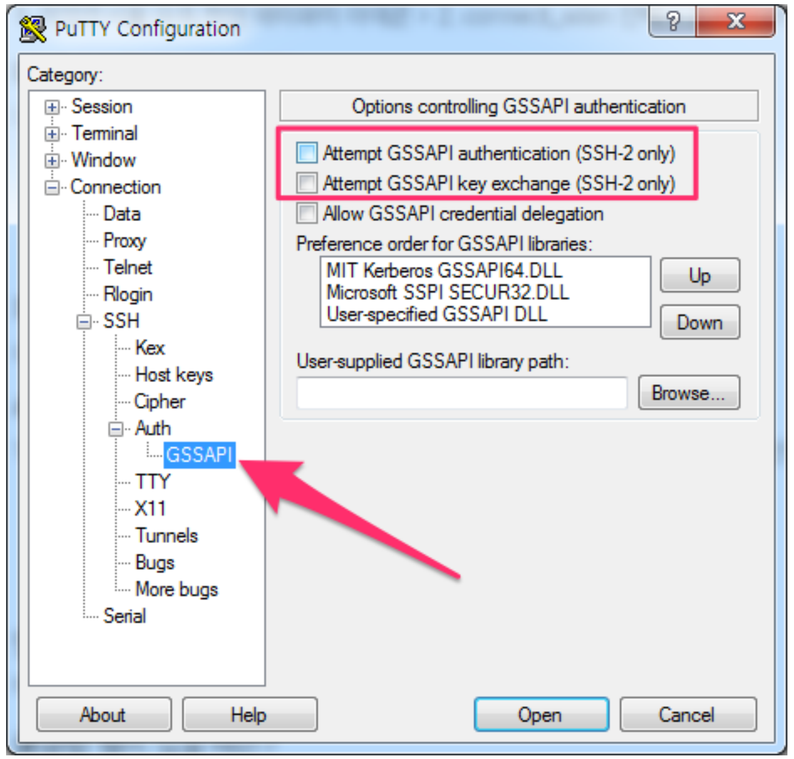


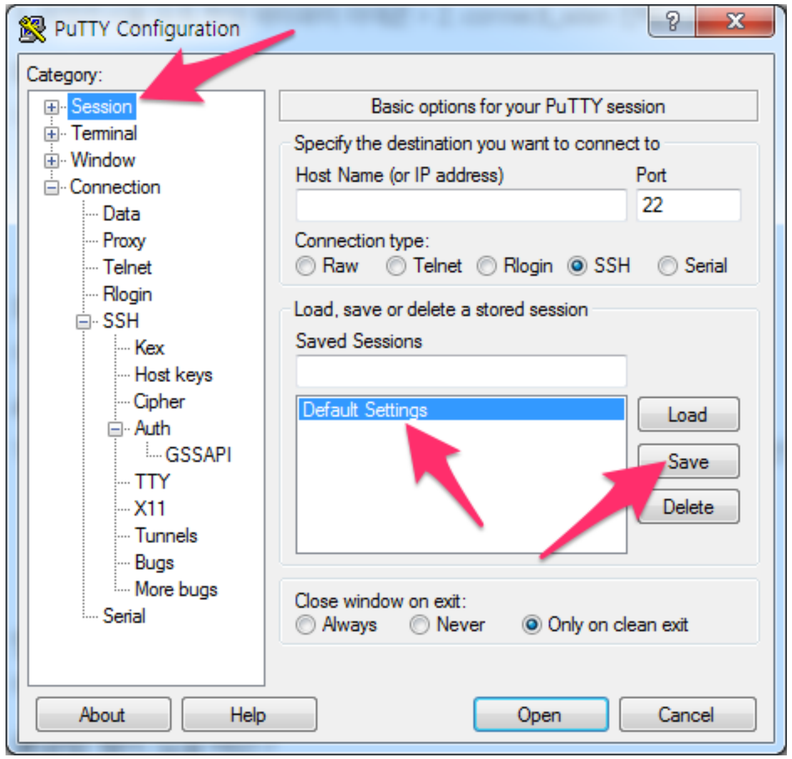
6.4 PuTTY 가 동작하지 않는다 (2/2)

DEVIEW 2019

Attempt GSSAPI authentication, Attempt GSSAPI key exchange 체크박스 해제한다

Default Settings에 저장하면 pscp.exe도 정상적으로 사용할 수 있다





6.5 YARN LOG를 확인할 수 없다 (1/2)

DEVIEW 2019

현상

MapReduce/Spark/Tez Job 런칭이 느리고(작업은 실패하지 않는다)

작업 완료 후에 yarn log를 확인할 수 없다

상황

HDFS Federation 구성했고

디폴트 파일 시스템(fs.defaultFS)은 hdfs://a

yarn log는 hdfs://b/app-logs 에 저장한다

YARN LOG 네임노드 분리 관련: DEVIEW 2017 멀티테넌트 하둡 클러스터 운영 경험기

6.5 YARN LOG를 확인할 수 없다 (2/2)

DEVIEW 2019

원인

yarn log를 저장하는 hdfs://b 네임노드의 Delegation token을 발급받지 않고 작업을 제출함
AppMaster가 할당된 노드매니저에서 yarn log 저장을 위한 디렉터리 생성 실패
2분동안 수차례 재시도하고 AppMaster 런칭됨 (런칭이 느려짐)
작업 완료 후 hdfs://b 네임노드 접속 실패로 yarn log 업로드 실패함 (yarn log 확인할 수 없게 됨)

해결

mapreduce.job.hdfs-servers=hdfs://b
spark.yarn.access.hadoopFileSystems=hdfs://b
tez.job.fs-servers=hdfs://b (tez-0.9.2 이상에서만 사용가능)

6.6 커버러스 디버깅

인증 관련 문제가 있으면 아래 환경변수를 세팅하면 디버깅 로그를 확인할 수 있다

```
export KRB5_TRACE=/dev/stderr
export JAVA_TOOL_OPTIONS="-Dsun.security.krb5.debug=true"
export HADOOP_LOGLEVEL=DEBUG
```

DEVIEW
2019

7. Open Source Contribution

오픈소스 컨트리뷰션

DEVIEW
2019

apache hadoop 3.1.2 기반의 프로덕션 클러스터를 운영
운영 중에 발견한 버그를 직접 수정하거나 apache jira 의 패치 적용

HADOOP-16441 if use -Dbundle.openssl=true, bundled with unnecessary libk5crypto.*
HDFS-14434 Ignore user.name query parameter in secure WebHDFS
YARN-9197 NPE in service AM when failed to launch container
YARN-9307 node_partitions constraint does not work
YARN-9521 RM failed to start due to system services
YARN-9583 Failed job which is submitted unknown queue is showed all users
YARN-9633 Support doas parameter at rest api of yarn-service
YARN-9647 Docker launch fails when local-dirs or log-dirs is unhealthy.
YARN-9691 canceling upgrade does not work if upgrade failed container is existing
YARN-9703 Failed to cancel yarn service upgrade when canceling multiple times
YARN-9719 Failed to restart yarn-service if it doesn’t exist in RM
YARN-9731 In ATS v1.5, all jobs are visible to all users without view-acl
YARN-9732 yarn.system-metrics-publisher.enabled=false is not honored by RM
YARN-9790 Failed to set default-application-lifetime if maximum-application-lifetime is less than or equal to zero

직접 공헌하지 않았지만 적용하면 좋을 패치

HDFS-14323. Distcp fails in Hadoop 3.x when 2.x source webhdfs url has special characters in hdfs file path

MAPREDUCE-7069 Add ability to specify user environment variables individually

YARN-8693 Add signalToContainer REST API for RMWebServices

YARN-8761 Service AM support for decommissioning component instances

YARN-9685 NPE when rendering the info table of leaf queue in non-accessible partitions

Q&A

DEVIEW
2019

Big Data&AI Platform

Data Suite

AI Suite

DataStore

DataProc

DataLog

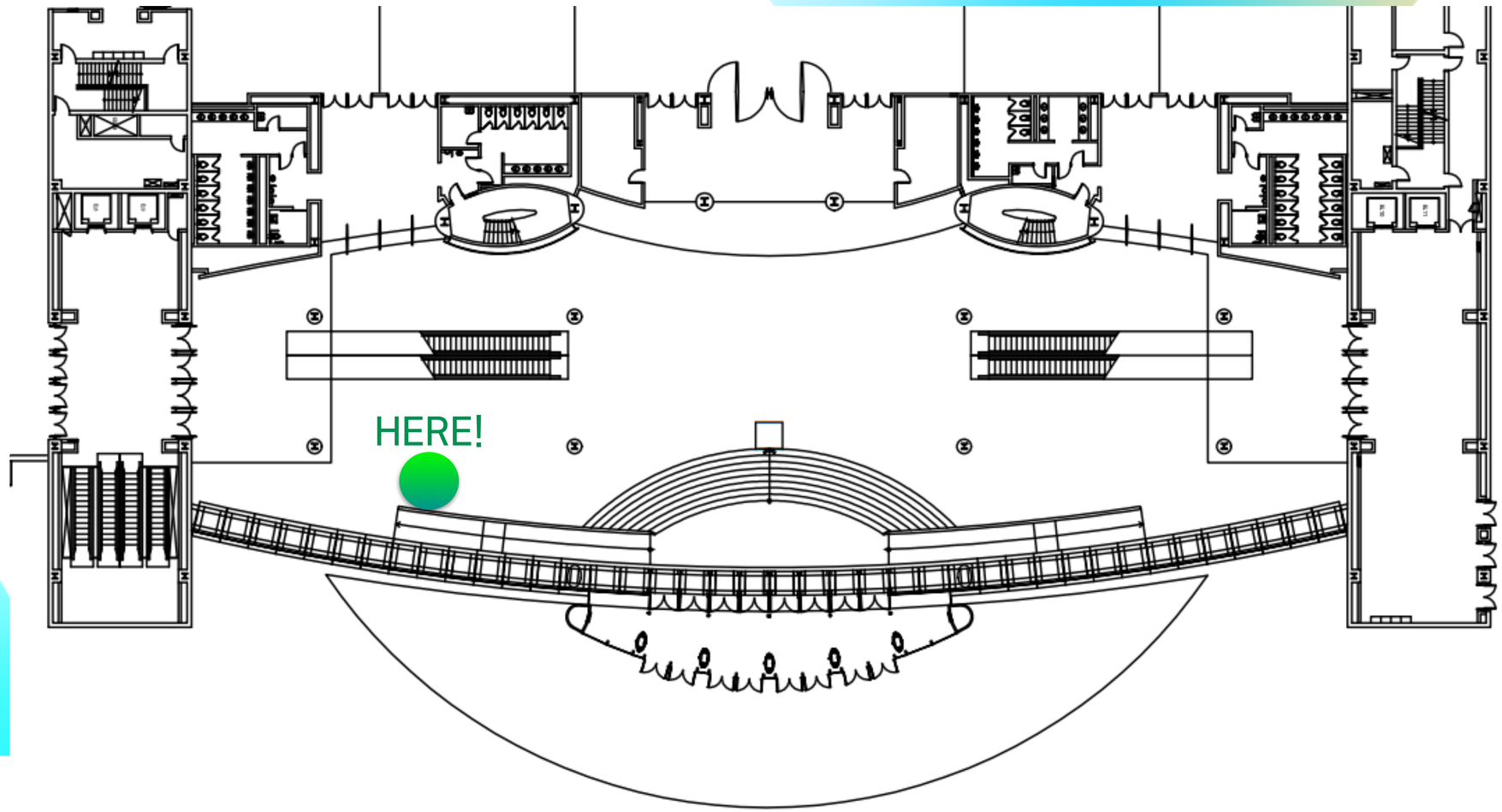
AiFeatures

AiTraining

AiServing

NAVER

DEVIEW
2019



DEVIEW
2019

Thank You

Appendix

A. Kerberos KDC 설치

MIT Kerberos 공식홈페이지의 문서가 매우 충실하게 잘 되어 있다
- https://web.mit.edu/kerberos/krb5-latest/doc/

다음장의 설정과 절차를 참고해서 설치과정을 진행하면 쉽게 KDC 를 설치할 있다
- https://web.mit.edu/kerberos/krb5-latest/doc/admin/install_kdc.html

다음장의 설치과정설명은 Master KDC 설치이다
운영용 클러스터에서는 MIT Kerberos 문서를 참고해서 Slave KDC 도 설치해야한다

KDC 설치 절차

DEVIEW
2019

```
# centos7 기준. root 권한으로 진행한다.
# 패키지 설치
yum -y install krb5-server krb5-libs krb5-workstation

# 커버러스 설정작성 (다음장 참고)
# /etc/krb5.conf
# /var/kerberos/krb5kdc/kdc.conf
# /var/kerberos/krb5kdc/kadm5.acl

# DB 초기화. 비밀번호를 잘 기억해둔다
kdb5_util create -s -P ${DB_MASTER_PASSWORD}

# Admin Principal 생성. 비밀번호를 잘 기억해둔다.
kadmin.local -q "addprinc -pw $ADMIN_PASSWORD admin/admin@EXAMPLE.COM"

# 데몬 시작
systemctl start krb5kdc
systemctl start kadmin
```

/etc/krb5.conf

DEVIEW
2019

```
[libdefaults]
  dns_lookup_realm = false
  dns_uri_lookup = false
  ticket_lifetime = 24h
  renew_lifetime = 7d
  forwardable = true
  rdns = false
  default_realm = EXAMPLE.COM
  default_ccache_name = /tmp/krb5cc_%{uid}

[realms]
  EXMAPLE.COM = {
    kdc = kdc1.host.com
    kdc = kdc2.host.com
    admin_server = kdc1.host.com
  }

[domain_realm]
```

/var/kerberos/krb5kdc/kdc.conf

DEVIEW 2019

```
[dbmodules]
 EXAMPLE.COM = {
  disable_last_success = true
 }

[kdcdefaults]
 kdc_ports = 88
 kdc_tcp_ports = 88

[realms]
 EXAMPLE.COM = {
  max_life = 24h
  max_renewable_life = 7d
  acl_file = /var/kerberos/krb5kdc/kadm5.acl
  dict_file = /usr/share/dict/words
  supported_enctypes = aes256-cts-hmac-sha1-96:normal aes128-cts-hmac-sha1-96:normal
  default_principal_flags = +preauth
 }

[logging]
  default = FILE:/var/log/krb5libs.log
  kdc = FILE:/var/log/krb5kdc.log
  admin_server = FILE:/var/log/kadmind.log
  debug = false
```

/var/kerberos/krb5kdc/kadm5.acl

DEVIEW
2019

```
*/admin@EXAMPLE.COM      *
```

B. OpenLDAP 설치

LDAP 용어 정리
http://electron-swamp.blogspot.com/2014/04/initializing-openldap-database-with.html

OpenLDAP 설치 (Single Master)
https://www.itzgeek.com/how-tos/linux/centos-how-tos/step-step-openldap-server-configuration-centos-7-rhel-7.html

OpenLDAP 설치 (Muliti Master)
https://www.itzgeek.com/how-tos/linux/centos-how-tos/configure-openldap-multi-master-replication-linux.html

OpenLDAP 설치 (Master-Slave Replication)
https://www.itzgeek.com/how-tos/linux/configure-openldap-master-slave-replication.html

다음장의 설치과정은 Single Master 이다.
운영용 클러스터에서는 위 링크의 문서를 참고해서 Multi Master or Master-Slave Replication 으로 구성한다.

OpenLDAP 설치절차 (1/4)

DEVIEW 2019

```
# centos7 기준. root 권한으로 진행한다.
# 패키지 설치
yum -y install openldap compat-openldap openldap-clients openldap-servers openldap-servers-sql openldap-devel
systemctl start slapd
systemctl enable slapd

# 아래 정보를 기준으로 LDAP이 세팅된다. 추후 LDAP 연동할때 아래 정보를 사용한다. 잘 기억해둔다
SUFFIX="dc=example,dc=com"  # 자신의 환경에 맞게 변경한다. e.g. dc=deview,dc=naver,dc=com
ROOT_DN="cn=root"
ROOT_PW="root_password"
USERS_OU="ou=users"
GROUPS_OU="ou=groups"

# 로깅 설정
echo "local4.* /var/log/ldap.log" | tee /etc/rsyslog.d/99-ldap.conf
systemctl restart rsyslog
temp_file=`mktemp`
echo "/var/log/ldap.log" | cat - /etc/logrotate.d/syslog | tee $temp_file
chmod --reference /etc/logrotate.d/syslog $temp_file
mv $temp_file /etc/logrotate.d/syslog

# root 비밀번호 생성
HASH_ROOT_PW=`slappasswd -h "{SSHA}" -s "$ROOT_PW"`
```

OpenLDAP 설치절차 (2/4)

DEVIEW 2019

```
# openldap 설정
ldif=config_openldap.ldif
cat << EOF | tee $ldif
dn: olcDatabase={2}hdb,cn=config
changetype: modify
replace: olcSuffix
olcSuffix: $SUFFIX
dn: olcDatabase={2}hdb,cn=config
changetype: modify
replace: olcRootDN
olcRootDN: $ROOT_DN
dn: olcDatabase={2}hdb,cn=config
changetype: modify
replace: olcRootPW
olcRootPW: $HASH_ROOT_PW
EOF

ldapmodify -Y EXTERNAL -H ldapi:/// -f $ldif

# ACL Monitor
ldif=acl_monitor.ldif
cat << EOF | tee $ldif
dn: olcDatabase={1}monitor,cn=config
changetype: modify
replace: olcAccess
olcAccess: {0}to * by dn.base="gidNumber=0+uidNumber=0,cn=peercred,cn=external, cn=auth" read by dn.base="$ROOT_DN" read by * none
EOF

ldapmodify -Y EXTERNAL  -H ldapi:/// -f $ldif
```

OpenLDAP 설치절차 (3/4)

DEVIEW 2019

```
# 데이타베이스 설정
ldif=config_database.ldif
cp /usr/share/openldap-servers/DB_CONFIG.example /var/lib/ldap/DB_CONFIG
chown ldap:ldap /var/lib/ldap/DB_CONFIG

ldapadd -Y EXTERNAL -H ldapi:/// -f /etc/openldap/schema/cosine.ldif
ldapadd -Y EXTERNAL -H ldapi:/// -f /etc/openldap/schema/nis.ldif
ldapadd -Y EXTERNAL -H ldapi:/// -f /etc/openldap/schema/inetorgperson.ldif

short_dc=`echo $SUFFIX | awk -F"," '{print $1}' | awk -F"=" '{print $2}'`
short_root=`echo $ROOT_DN | awk -F"," '{print $1}' | awk -F"=" '{print $2}'`
short_users_ou=`echo $USERS_OU | awk -F"," '{print $1}' | awk -F"=" '{print $2}'`
short_groups_ou=`echo $GROUPS_OU | awk -F"," '{print $1}' | awk -F"=" '{print $2}'`

cat << EOF | tee $ldif
dn: $SUFFIX
dc: $short_dc
objectClass: top
objectClass: domain
dn: $ROOT_DN
objectClass: organizationalRole
cn: $short_root
description: LDAP Manager
dn: $USERS_OU
objectClass: organizationalUnit
ou: $short_users_ou
dn: $GROUPS_OU
objectClass: organizationalUnit
ou: $short_groups_ou
EOF

ldapadd -H ldapi:/// -x -w "$ROOT_PW" -D "$ROOT_DN" -f $ldif
```

OpenLDAP 설치절차 (4/4)

DEVIEW 2019

```
# ACL 설정
ldif=config_acl.ldif
cat << EOF | tee $ldif
dn: olcDatabase={2}hdb,cn=config
changetype: modify
replace: olcAccess
olcAccess: to attrs=userPassword,shadowLastChange
  by self write
  by anonymous auth
  by dn.base="$ROOT_DN" write
  by * none
olcAccess: to *
  by self read
  by dn.base="$ROOT_DN" write
  by * read
EOF

ldapmodify -Y EXTERNAL -H ldapi:/// -f $ldif
```

OpenLDAP 새로운 그룹추가

DEVIEW 2019

```
# openldap 서버외부의 원격지에서 실행해도 된다
SUFFIX="dc=example,dc=com"
ROOT_DN="cn=root"
ROOT_PW="root_password"
USERS_OU="ou=users"
GROUPS_OU="ou=groups"
LDAP_URI="ldap://ldap.host.com"
LDAP_USER_SEARCH_BASE=${USERS_OU},${SUFFIX}
LDAP_GROUP_SEARCH_BASE=${GROUPS_OU},${SUFFIX}
# 이상 공통 코드
#########################################################

# 추가할 그룹명
GROUP=hadoop-admins
# GID_NUMBER 는 유니크해야한다. 그룹을 추가할때마다 다른 그룹의 GID_NUMBER 와 중복되지 않게 잘 관리해야한다
GID_NUMBER=20001
DESC="그룹 설명"

LDIF=add_group.ldif
cat << EOF | tee $LDIF
dn: cn=$GROUP,$LDAP_GROUP_SEARCH_BASE
objectClass: top
objectClass: posixGroup
gidNumber: $GID
description: $DESC
EOF

ldapadd -w $ROOT_PW -H $LDAP_URI -D "${ROOT_DN},${SUFFIX}" -f $LDIF
```

OpenLDAP 새로운 계정추가

DEVIEW 2019

```
# openldap 서버외부의 원격지에서 실행해도 된다
SUFFIX="dc=example,dc=com"
ROOT_DN="cn=root"
ROOT_PW="root_password"
USERS_OU="ou=users"
GROUPS_OU="ou=groups"
LDAP_URI="ldap://ldap.host.com"
LDAP_USER_SEARCH_BASE=${USERS_OU},${SUFFIX}
LDAP_GROUP_SEARCH_BASE=${GROUPS_OU},${SUFFIX}
# 이상 공통 코드
#########################################################

# 추가할 계정명
USER=foo
# 사용자 비밀번호
PASSWORD=deview1234
# UID_NUMBER 는 유니크해야한다.
# 계정을 추가할때 다른 계정의 UID_NUMBER 와 중복되지 않게 잘 관리해야한다
UID_NUMBER=20001
# 계정이 속할 그룹의 GID_NUMBER
GID_NUMBER=20001
DESC="계정 설명”

LDIF=add_user.ldif
HASH_PASSWORD=`/usr/sbin/slappasswd -s $PASSWORD -h {SSHA}`

cat << EOF | tee $LDIF
dn: uid=$USER,$LDAP_USER_SEARCH_BASE
objectClass: top
objectClass: account
objectClass: posixAccount
cn: $USER
uid: $USER
uidNumber: $UID_NUMBER
gidNumber: $GID_NUMBER
homeDirectory: /home/$USER
loginShell: /bin/bash
gecos: $DESC
userPassword: $HASH_PASSWORD
EOF

ldapadd -w $ROOT_PW -H $LDAP_URI -D "${ROOT_DN},${SUFFIX}" -f $LDIF
```

OpenLDAP 그룹멤버 추가

DEVIEW 2019

```
# openldap 서버외부의 원격지에서 실행해도 된다
# foo 계정을 만들때 그룹을 hadoop-admins 로 지정했다. foo 계정을 users 그룹에 추가하려고 할때 아래절차로 진행한다.users 그룹은 미리 추가해놓는다
SUFFIX="dc=example,dc=com"
ROOT_DN="cn=root"
ROOT_PW="root_password"
USERS_OU="ou=users"
GROUPS_OU="ou=groups"
LDAP_URI="ldap://ldap.host.com"
LDAP_USER_SEARCH_BASE=${USERS_OU},${SUFFIX}
LDAP_GROUP_SEARCH_BASE=${GROUPS_OU},${SUFFIX}
# 이상 공통 코드
#########################################################

GROUP=users
# users 그룹에 추가할 계정명
USER=foo

LDIF=usermod_add_user_to_group.ldif
cat << EOF | tee $LDIF
dn: cn=$GROUP,$LDAP_GROUP_SEARCH_BASE
changetype: modify
add: memberUid
memberUid: $USER
EOF

ldapmodify -w $ROOT_PW -H $LDAP_URI -D "${ROOT_DN},${SUFFIX}" -f $LDIF
```

OpenLDAP 그룹멤버 삭제

DEVIEW 2019

```
# openldap 서버외부의 원격지에서 실행해도 된다
# foo 계정은 hadoop-admins, users 그룹에 속해 있다. users 그룹에서 foo 계정을 제거한다
SUFFIX="dc=example,dc=com"
ROOT_DN="cn=root"
ROOT_PW="root_password"
USERS_OU="ou=users"
GROUPS_OU="ou=groups"
LDAP_URI="ldap://ldap.host.com"
LDAP_USER_SEARCH_BASE=${USERS_OU},${SUFFIX}
LDAP_GROUP_SEARCH_BASE=${GROUPS_OU},${SUFFIX}
# 이상 공통 코드
#########################################################

GROUP=users
# users 그룹에서 제거할 계정명
USER=foo

LDIF=usermod_del_user_in_group.ldif
cat << EOF | tee $LDIF
dn: cn=$GROUP,$LDAP_GROUP_SEARCH_BASE
changetype: modify
delete: memberUid
memberUid: $USER
EOF

ldapmodify -w $ROOT_PW -H $LDAP_URI -D "${ROOT_DN},${SUFFIX}" -f $LDIF
```

OpenLDAP 그룹/계정 삭제

DEVIEW 2019

```
# openldap 서버외부의 원격지에서 실행해도 된다
# 그룹, 계정 삭제한다.
SUFFIX="dc=example,dc=com"
ROOT_DN="cn=root"
ROOT_PW="root_password"
USERS_OU="ou=users"
GROUPS_OU="ou=groups"
LDAP_URI="ldap://ldap.host.com"
LDAP_USER_SEARCH_BASE=${USERS_OU},${SUFFIX}
LDAP_GROUP_SEARCH_BASE=${GROUPS_OU},${SUFFIX}
# 이상 공통 코드
#########################################################

# 삭제할 그룹명
GROUP=users
ldapdelete -w $ROOT_PW -H $LDAP_URI -D "${ROOT_DN},${SUFFIX}" cn=$GROUP,$LDAP_GROUP_SEARCH_BASE

# 삭제할 계정
USER=foo
ldapdelete -w $ROOT_PW -H $LDAP_URI -D "${ROOT_DN},${SUFFIX}" uid=$USER,$LDAP_USER_SEARCH_BASE
```

C. LDAP 연동

리눅스 노드 LDAP 연동

DEVIEW 2019

```
# centos7 기준. 모든 작업은 root 로 진행한다
# 클러스터내의 모든 노드(네임노드, 리소스매니저 포함)에서 실행해야한다.
# LDAP 클라이언트 관련 패키지 설치
yum install -y sssd sssd-client sssd-ldap openldap-clients oddjob-mkhomedir

# LDAP 계정으로 노드에 ssh 접속가능하게 할 경우 (e.g. 노드매니저)
authconfig --enablesssd --enablesssdauth --enablemkhomedir --update
# LDAP 계정으로 노드에 ssh 접속 불가능하게 할 경우 (e.g. 네임노드, 리소스매니저)
authconfig --enablesssd --disablesssdauth --enablemkhomedir --update

# /etc/sssd/sssd.conf 작성

# 데몬 재시작
dbus-daemon --system
systemctl start sssd
systemctl start oddjobd
```

/etc/sssd/sssd.conf

```
[domain/LDAP]
#ref: man sssd.conf
id_provider = ldap
auth_provider = ldap
cache_credentials = true
enumerate = true

#ref: man sssd-ldap
ldap_schema = rfc2307

# LDAP 정보를 변경한다.
ldap_uri = ldap://ldap1.host.com,ldap://ldap2.host.com
ldap_search_base = dc=example,dc=com
ldap_user_search_base = ou=users,dc=example,dc=com
ldap_group_search_base = ou=groups,dc=example,dc=com

ldap_user_object_class = posixAccount
ldap_group_object_class = posixGroup
ldap_user_gecos = gecos
ldap_tls_reqcert = allow
ldap_id_use_start_tls = false
ldap_pwd_policy = none
ldap_chpass_update_last_change = false

[sssd]
config_file_version = 2
services = nss, pam
domains = LDAP

[nss]
[pam]
[sudo]
[autofs]
[ssh]
[pac]
```

Ambari LDAP 연동

DEVIEW 2019

1. /sbin/ambari-server setup-ldap 명령으로 LDAP 설정하고 ambari-server 재시작

```
# /sbin/ambari-server setup-ldap
Using python  /usr/bin/python
Currently 'no auth method' is configured, do you wish to use LDAP instead [y/n] (y)?
Enter Ambari Admin login: admin
Enter Ambari Admin password:

Fetching LDAP configuration from DB. No configuration.
Please select the type of LDAP you want to use [AD/IPA/Generic](Generic):
Primary LDAP Host (ldap.ambari.apache.org): ldap.host.com
Primary LDAP Port (389):
Secondary LDAP Host <Optional>:
Secondary LDAP Port <Optional>:
Use SSL [true/false] (false):
User object class (posixUser): posixAccount
User ID attribute (uid):
Group object class (posixGroup):
Group name attribute (cn):
Group member attribute (memberUid):
Distinguished name attribute (dn): ou=users,dc=example,dc=com
Search Base (dc=ambari,dc=apache,dc=org): dc=example,dc=com
Referral method [follow/ignore] (follow):
Bind anonymously [true/false] (false): true
Handling behavior for username collisions [convert/skip] for LDAP sync (skip):
Force lower-case user names [true/false]:true
Results from LDAP are paginated when requested [true/false]:true

Save settings [y/n] (y)? y
Saving LDAP properties...
Saving LDAP properties finished
Ambari Server 'setup-ldap' completed successfully.
```

2. hadoop-admins 그룹에 속한 사용자만 동기화.

운영자만 Ambari 에 로그인할 수 있게 한다.

자동으로 동기화되지 않는다. 주기적으로 아래명령으로 동기화해야한다.

```
echo hadoop-admins > groups_file.txt
/sbin/ambari-server sync-ldap -v --groups=groups_file.txt \
                                 --ldap-sync-admin-name=$ADMIN_USER \
                                 --ldap-sync-admin-password=$ADMIN_PASSWORD
```

Ranger LDAP 연동 (1/3)

Ranger 는 클러스터 운영자만 사용하지 않고 일반 사용자들도 사용하게 된다. 보통은 아래같은 패턴으로 사용하게 된다.

1. 사용자X 가 사용자Y 의 데이터인 /user/y/some_data 에 읽기 권한을 요청한다.
2. 사용자Y 는 Ranger 에서 해당 hdfs 경로에 대해서 사용자X 에게 읽기 권한을 부여한다.

사용자들은 LDAP 계정으로 Ranger 로그인이 가능해지고(사용자Y 가 로그인) Ranger 에서 권한을 부여할 사용자들을 검색(사용자X 를 찾아야함) 할 수 도 있다.
Ranger 구성요소중에 RANGER USERSYNC 가 주기적으로 LDAP 에서 사용자정보를 읽어와서 Ranger 가 사용하는 DB 에 저장을 한다.

아래 두 가지 케이스에 대해 LDAP과 연동을 위한 각각의 설정이 필요하다.
1. Ranger Web-UI 로그인
2. Ranger USERSYNC 에서 주기적인 동기화

Ranger LDAP 연동 (2/3)

DEVIEW 2019

Ambari Web-UI 에서 RANGER -> CONFIGS -> RANGER USER INFO 으로 진입하고 아래같이 설정한다

Sync Source: LDAP/AP

COMMON CONFIGS

- LDAP URL: ldap://ldap.host.com
- Bind User: cn=root,dc=example,dc=com
- Bind User Password: LDAP 설치할때 입력한 root 비밀번호
- Incremetal Sync: Yes
- Enable LDAP STARTTLS: No

GROUP CONFIGS

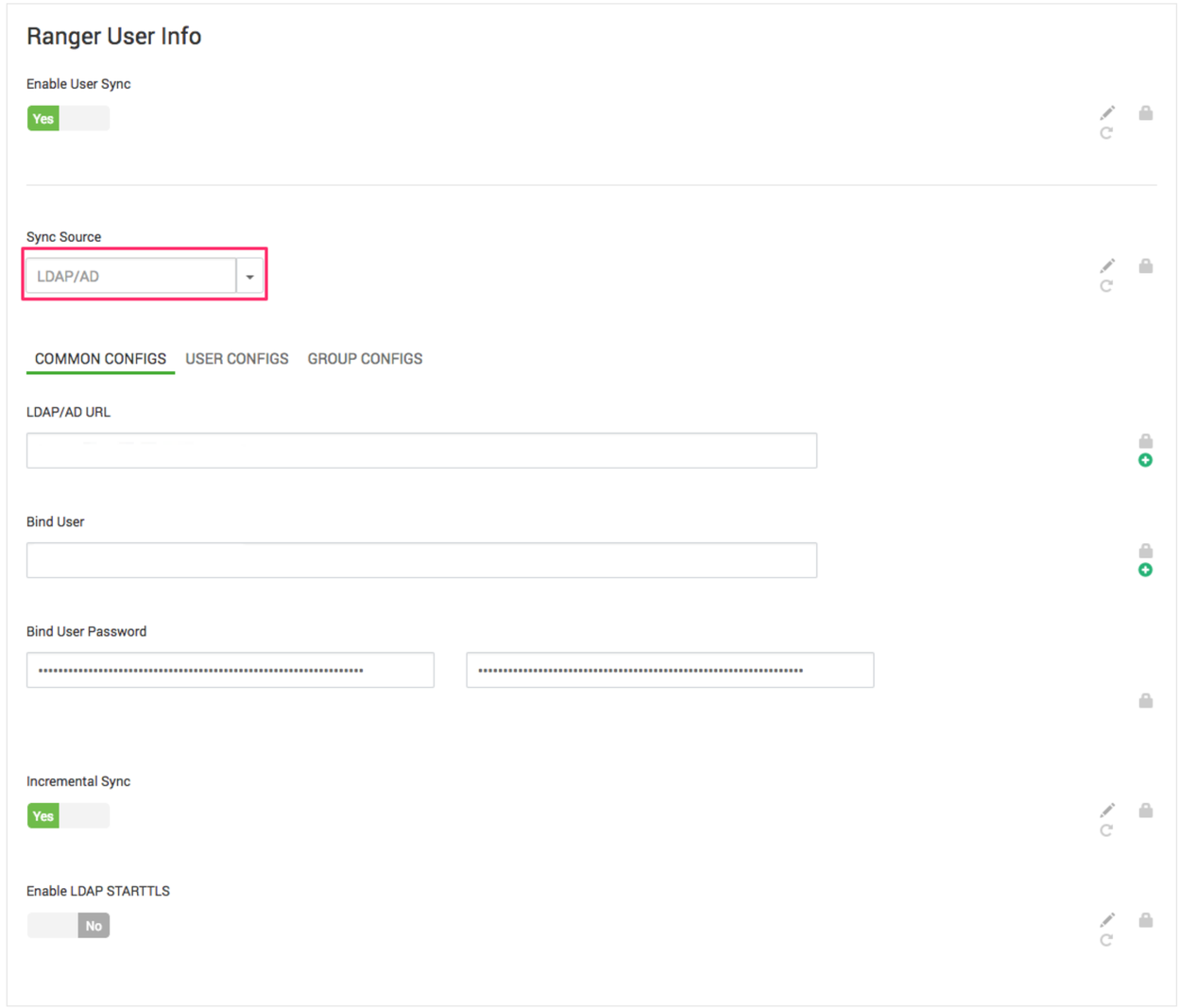
- Enable Group Sync: yes
- Group Member Attribute : memberUid
- Group Name Attribute: cn
- Group Object Class: posixGroup
- Group Search Base: ou=groups,dc=example,dc=com
- Group Search Filter: 빈값
- Enable Group Search First: Yes
- Sync Nested Groups: No

USER CONFIGS

- Username Attribute: uid
- User Object Class: posixAccount
- User Search Base: ou=users,dc=example,dc=com
- User Search Filter: 빈값
- User Search Scope: sub
- User Group Name Attribute gidNumber
- Group User Map Sync: Yes
- Enable User Search: Yes

ranger-ugsync-site.xml (Advanced ranger-ugsync-site)

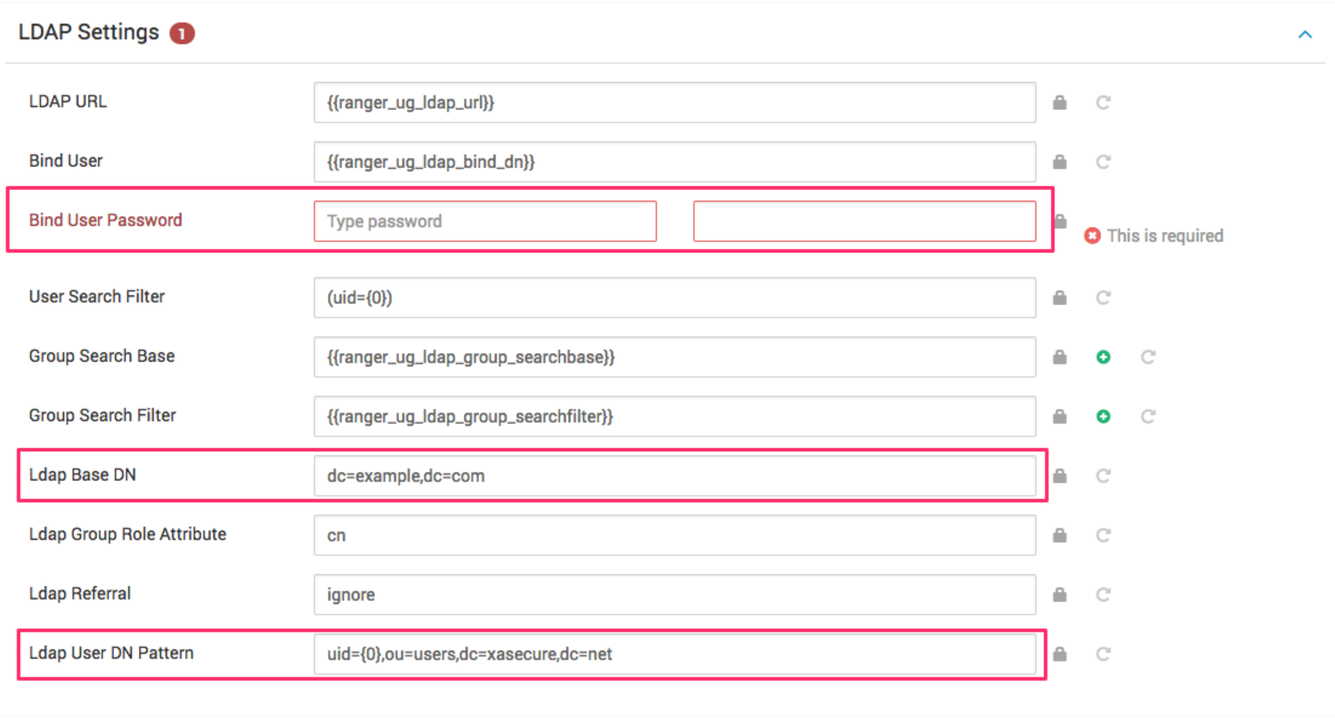
- ranger.usersync.ldap.searchBase=dc=example,dc=com



Ranger LDAP 연동 (3/3)

DEVIEW 2019

Bind User Password: LDAP 설치할때 입력한 root 비밀번호
Ldap Base DN: dc=example,dc=com
Ldap User DN Pattern: uid={0},ou=users,dc=example,dc=com



D. 하둡 설정 변경

DEVIEW
2019

core-site.xml

DEVIEW
2019

```
ipc.server.listen.queue.size=3000
hadoop.security.service.user.name.key.pattern=*
hadoop.http.cross-origin.allowed-origins=*
```

hdfs-site.xml


```
dfs.namenode.handler.count=200
dfs.namenode.service.handler.count=100
dfs.namenode.lifeline.handler.ratio=0.20
dfs.namenode.service.rpc-address.<ns>.<nnid>=namenode.host:8022      # HA 설정에 맞게 네임서비스와 네임노드ID 를 명시해야한다
dfs.namenode.lifeline.rpc-address.<ns>.<nnid>=namenode.host:8024     # HA 설정에 맞게 네임서비스와 네임노드ID 를 명시해야한다
dfs.blockreport.split.threshold=100000
dfs.blockreport.initialDelay=2000
dfs.blocksize=268435456
dfs.datanode.du.reserved=429496729600
dfs.datanode.handler.count=100
dfs.image.transfer.bandwidthPerSec=31457280
dfs.namenode.accesstime.precision=2592000000
dfs.namenode.checkpoint.txns=9000000
dfs.namenode.checkpoint.period=3600
dfs.namenode.quota.init-threads=8
fs.permissions.umask-mode=077
ipc.8020.callqueue.impl=org.apache.hadoop.ipc.FairCallQueue
dfs.journalnode.kerberos.principal.pattern=*
dfs.datanode.kerberos.principal.pattern=*
dfs.namenode.kerberos.principal.pattern=*
```

yarn-site.xml


```
yarn.log-aggregation.retain-seconds=604800
yarn.log-aggregation.retain-check-interval-seconds=86400
yarn.node-labels.enabled=true
yarn.node-labels.fs-store.retry-policy-spec=100, 1
yarn.nodemanager.delete.debug-delay-sec=3600
yarn.nodemanager.disk-health-checker.max-disk-utilization-per-disk-percentage=100
yarn.nodemanager.disk-health-checker.min-free-space-per-disk-mb=1024
yarn.resourcemanager.am.max-attempts=4
yarn.resourcemanager.state-store.max-completed-applications=0
yarn.scheduler.minimum-allocation-mb=512
yarn.resourcemanager.principal.pattern=*
yarn.timeline-service.principal.pattern=*
```

Advanced hadoop-env

DEVIEW 2019

```
hdfs_user_nofile_limit=640000
hdfs_user_nproc_limit=640000
```

MapReduce Framework replication factor 조정

DEVIEW 2019

Mapreduce framework replication 50으로 상향조정

```
$ hdfs dfs -ls /hdp/apps/3.1.0.0-78/mapreduce
Found 2 items
-r--r--r--   3 hdfs hadoop     176342 2019-01-22 19:56 /hdp/apps/3.1.0.0-78/mapreduce/hadoop-streaming.jar
-r--r--r--   3 hdfs hadoop  308401145 2019-01-22 19:51 /hdp/apps/3.1.0.0-78/mapreduce/mapreduce.tar.gz

# setrep 50
$ hdfs dfs -setrep 50 /hdp/apps/3.1.0.0-78/mapreduce/mapreduce.tar.gz
Replication 50 set: /hdp/apps/3.1.0.0-78/mapreduce/mapreduce.tar.gz
$ hdfs dfs -setrep 50 /hdp/apps/3.1.0.0-78/mapreduce/hadoop-streaming.jar
Replication 50 set: /hdp/apps/3.1.0.0-78/mapreduce/hadoop-streaming.jar

# 확인
$ hdfs dfs -ls /hdp/apps/3.1.0.0-78/mapreduce
Found 2 items
-r--r--r--  50 hdfs hadoop     176342 2019-01-22 19:56 /hdp/apps/3.1.0.0-78/mapreduce/hadoop-streaming.jar
-r--r--r--  50 hdfs hadoop  308401145 2019-01-22 19:51 /hdp/apps/3.1.0.0-78/mapreduce/mapreduce.tar.gz
```

oozie-site.xml

DEVIEW
2019

```
oozie.processing.timezone=GMT+0900
```

E. SPNEGO 접속을 위한 브라우저 설정

macOS: 사파리

DEVIEW
2019

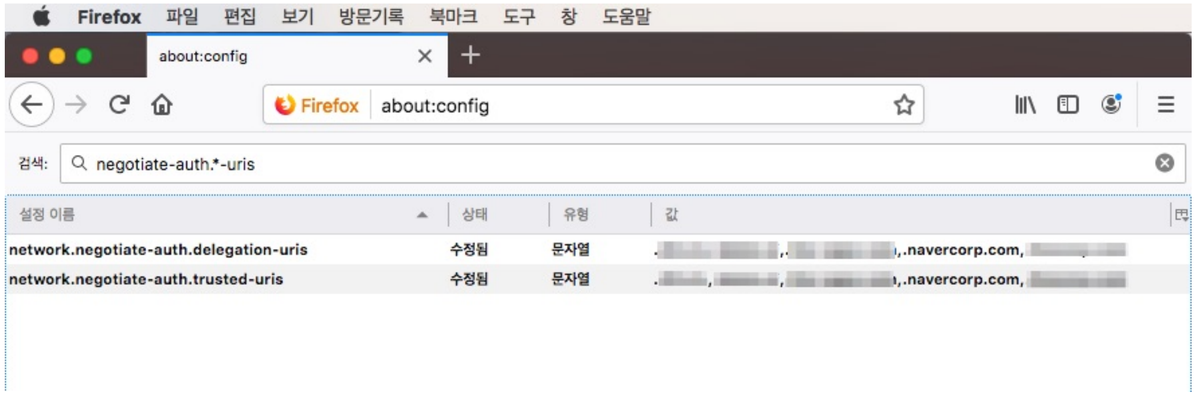
추가적인 설정없이 SPNEGO Web-UI 에 접속가능

macOS: 파이어폭스

DEVIEW 2019

주소창에 about:config 입력후 아래 설정에 하둡서버 도메인을 입력한다. 다수일경우 , 구분자로 나열한다

```
network.negotiate-auth.trusted-uris = .navercorp.com,.naver.com
network.negotiate-auth.delegation-uris = .navercorp.com,.naver.com
```



macOS: 구글 크롬, 네이버 웨일

DEVIEW
2019

셸에서 브라우저를 실행시켜야하고 --auth-server-whitelist 파라미터에 하둡서버 도메인목록을 , 구분자로 나열한다
브라우저에 설정을 저장하는 방법은 없다

크롬

```
/Applications/Google\ Chrome.app/Contents/MacOS/Google\ Chrome --auth-server-whitelist=".navercorp.com,.naver.com"
```

웨일

```
/Applications/Whale.app/Contents/MacOS/Whale --auth-server-whitelist=".navercorp.com,.naver.com"
```

Windows: 파이어폭스

주소창에 about:config 입력후 아래 설정을 모두 입력한다.
network.negotiate-auth.*-uris 에 하둡서버 도메인을 입력한다

```
network.negotiate-auth.trusted-uris = .navercorp.com,.naver.com
network.negotiate-auth.delegation-uris = .navercorp.com,.naver.com
network.negotiate-auth.using-native-gsslib = false
network.negotiate-auth.gsslib = C:\Program Files\MIT\Kerberos\bin\gssapi64.dll
network.auth.use-sspi = false
network.negotiate-auth.allow-non-fqdn = true
```