Item 279
git_comments:

1. ## Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements. See the NOTICE file distributed with this work for additional information regarding copyright ownership. The ASF licenses this file to you under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at http://www.apache.org/licenses/LICENSE-2.0 Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License. ##
2. Capture last command status
3. !/usr/bin/env bash ## Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements. See the NOTICE file distributed with this work for additional information regarding copyright ownership. The ASF licenses this file to you under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at http://www.apache.org/licenses/LICENSE-2.0 Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License. ##
4. conda activate does stuff with unbound variables :(
5. Secure Phoenix setup
6. (at least) one other phoenix test triggers the caching of this field before the KDC is up which causes principal parsing to fail.
7. Ensure the dirs we need are created/empty
8. Render a Heimdal compatible krb5.conf Currently kinit will only try tcp if the KDC is defined as kdc = tcp/hostname:port
9. It appears that we cannot generate a krb5.conf that is compatible with both MIT Kerberos and Heimdal Kerberos that works with MiniKdc. MiniKdc forces a choice between either UDP or or TCP for the KDC port. If we could have MiniKdc support both UDP and TCP, then we might be able to converge on a single krb5.conf for both MIT and Heimdal. With the below Heimdal configuration, MIT kerberos will fail on a DNS lookup to the hostname "tcp/localhost" instead of pulling off the "tcp/" prefix.
10. * * Setup and start kerberos, hbase
11. * Licensed to the Apache Software Foundation (ASF) under one or more * contributor license agreements. See the NOTICE file distributed with * this work for additional information regarding copyright ownership. * The ASF licenses this file to you under the Apache License, Version 2.0 * (the "License"); you may not use this file except in compliance with * the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, software * distributed under the License is distributed on an "AS IS" BASIS, * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. * See the License for the specific language governing permissions and * limitations under the License.
12. Set configuration for HBase
13. Then fork a thread with PQS in it.
14. Launch PQS, doing in the Kerberos login instead of letting PQS do it itself (which would break the HBase/HDFS logins also running in the same test case).
15. Create a number of unprivileged users
16. Enable token access for HDFS blocks
17. Only use HTTPS (required because we aren't using "secure" ports)
18. Generate SSL certs
19. Required so that PQS can impersonate the end-users to HBase
20. Default realm for MiniKDC
21. * * Setup the security configuration for hdfs.
22. * * This integration test stands up a secured PQS and runs Python code against it. See supporting * files in phoenix-queryserver/src/it/bin.
23. Make sure the ConnectionInfo doesn't try to pull a default Configuration
24. Start ZK by hand
25. Remove our custom ConfigurationFactory for future tests
26. dump stdout and stderr
27. * * Verifies that there is a python Executable on the PATH
28. Set principal+keytab configuration for HDFS
29. Start a MiniKDC

30. Start HDFS
31. Use LocalHBaseCluster to avoid HBaseTestingUtility from doing something wrong NB. I'm not actually sure what HTU does incorrect, but this was pulled from some test classes in HBase itself. I couldn't get HTU to work myself (2017/07/06)
32. Create a service principal and spnego principal in one keytab NB. Due to some apparent limitations between HDFS and HBase in the same JVM, trying to use separate identies for HBase and HDFS results in a GSS initiate error. The quick solution is to just use a single "service" principal instead of "hbase" and "hdfs" (or "dn" and "nn") per usual.
33. This assumes the test is being run from phoenix/phoenix-queryserver
34. **comment:** Not managed by miniKDC so we have to clean up
    label: code-design
35. Magic flag to tell hdfs to not fail on using ports above 1024
36. Get the PQS ident for PQS to use
37. Bind on localhost for spnego to have a chance at working
38. Clear the cached singletons so we can inject our own.
39. !/bin/bash
40. krb5_newrealm returns non-0 return code as it is running in a container, ignore it for this command only
41. python 2.7 introduced a NullHandler which we want to use, but to support older versions, we implement our own if needed.
42. Unable to attempt mutual authentication when mutual auth is required, raise an exception so the user doesn't use an untrusted response.
43. If this is set pass along the struct to Kerberos
44. if the cert signature algorithm is either md5 or sha1 then use sha256 otherwise use the signature algorithm
45. GSS Failure, return existing response
46. Flags used by kerberos module.
47. contexts still need to be stored by host, but hostname_override allows use of an arbitrary hostname for the kerberos exchange (eg, in cases of aliased hosts, internal vs external, CNAMEs w/ name-based HTTP hosting)
48. ensure we raised this for translation to KerberosExchangeError by comparing errno to result, re-raise if not
49. Still receiving 401 responses after attempting to handle them. Authentication has failed. Return the 401 response.
50. if we have a previous response from the server, use it to continue the auth process, otherwise use an empty value
51. Check if we have already tried to get the CBT data value
52. https://tools.ietf.org/html/rfc5929#section-4.1
53. Rewind the file position indicator of the body to where it was to resend the request.
54. Set the CBT values populated after the first response
55. add Authorization header before we receive a 401 by the 401 handler
56. There's no need to re-compile this EVERY time it is called. Compile it once and you won't have the performance hit of the compilation.
57. Regardless of the result, set tried to True so we don't waste time next time
58. 401 Unauthorized. Handle it, and if it still comes back as 401, that means authentication failed.
59. Authentication successful
60. Mutual authentication failure when mutual auth is wanted, raise an exception so the user doesn't use an untrusted response.
61. Only the latest version of pykerberos has this method available
62. If we haven't tried, try getting it now
63. Using older version set to None
64. **comment:** Consume the content so we can reuse the connection for the next request.
    label: code-design
65. In the case of HTTPKerberosAuth being reused and the body of the previous request was a file-like object, pos has the file position of the previous body. Ensure it's set to None.
66. Different types of mutual authentication: with mutual_authentication set to REQUIRED, all responses will be authenticated with the exception of errors. Errors will have their contents and headers stripped. If a non-error response cannot be authenticated, a MutualAuthenticationError exception will be raised. with mutual_authentication set to OPTIONAL, mutual authentication will be attempted if supported, and if supported and failed, a MutualAuthenticationError exception will be raised. Responses which do not support mutual authentication will be returned directly to the user. with mutual_authentication set to DISABLED, mutual authentication will not be attempted, even if supported.
67. !/usr/bin/env python coding: utf-8
68. **comment:** Skip the test if not set
    label: test
69. kerberos.authGSSCLientResponse() is called with the kerberos context which was initially returned by authGSSClientInit and had been mutated by a call by authGSSClientStep. It returns a string.
70. On Windows
71. Get a 401 from server, authenticate, and get a 200 back.

72. Get a 401 from server, authenticate, and get another 401 back. Ensure there is no infinite recursion.
73. kerberos.authGSSClientStep() is called with the kerberos context object returned by authGSSClientInit and the negotiate auth token provided in the http response's www-authenticate header. It returns 0 or 1 on success. 0 Indicates that authentication is progressing but not complete.
74. !/usr/bin/env python -*- coding: utf-8 -*-
75. re-test with error response sanitizing disabled
76. kerberos.authClientInit() is called with the service name (HTTP@FQDN) and returns 1 and a kerberos context object on success. Returns -1 on failure.
77. Note: we're not using the @mock.patch decorator: > My only word of warning is that in the past, the patch decorator hides > tests when using the standard unittest library. > -- sigmavirus24 in https://github.com/requests/requests-kerberos/issues/1
78. Manually edited from test_ecdsa_sha512 to change the OID to '1.2.840.10045.4.3.5'
79. Copyright 2015 Lukas Lalinsky Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at http://www.apache.org/licenses/LICENSE-2.0
80. from requests_gssapi import HTTPSPNEGOAuth, OPTIONAL
81. response = requests.request('post', self.url.geturl(), data=body, stream=True, headers=headers, auth=HTTPSPNEGOAuth(mutual_authentication=OPTIONAL))
82. Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements. See the NOTICE file distributed with this work for additional information regarding copyright ownership. The ASF licenses this file to you under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at http://www.apache.org/licenses/LICENSE-2.0 Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.
83. Not our code
84. Copyright 2015 Lukas Lalinsky Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at http://www.apache.org/licenses/LICENSE-2.0
85. from requests_gssapi import HTTPSPNEGOAuth, OPTIONAL
86. response = requests.request('post', self.url.geturl(), data=body, stream=True, headers=headers, auth=HTTPSPNEGOAuth(mutual_authentication=OPTIONAL))
87. Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements. See the NOTICE file distributed with this work for additional information regarding copyright ownership. The ASF licenses this file to you under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at http://www.apache.org/licenses/LICENSE-2.0 Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

git_commits:

1. **summary:** PHOENIX-4688 Support SPNEGO for python driver via requests-kerberos
   message: PHOENIX-4688 Support SPNEGO for python driver via requests-kerberos Includes updated L&N for requests-kerberos. Tries to detect when the host system doesn't have necessary dependencies to run the test Closes #344 Signed-off-by: Josh Elser <elserj@apache.org>

github_issues:

github_issues_comments:

github_pulls:

1. **title:** PHOENIX-4688 Kerberize python phoenixdb -- Phoenix 5
   body: Lets rip out httplib and replace with requests and use requests kerberos #### Notes - Will not work out of the box, since requires virtualenv on anaconda - Excluded the IT from list, needs to be invoked manually `mvn -Dit.test=org.apache.phoenix.end2end.SecureQueryServerPhoenixDBIT verify` #### Standalone install This is currently a manual procedure with a few steps due to the current version not being in a public pipy and also the fact that we had to fork requests-kerberos 1. Create a python virtual environment by calling `conda create` or `virtual-env` 2. . activate the virtual environment 3. pip install -e phoenix/python/requests-kerberos 4. pip install -e phoenix/python/phoenixdb-module 5. You need kerberos credentials run kinit 6. Run a python program to access a DB similar to one below 7. Profit #### Example script ````python import phoenixdb import phoenixdb.cursor import sys if __name__ == '__main__': pqs_port = sys.argv[1] database_url = 'http://localhost:' + str(pqs_port) + '/' print "CREATING PQS CONNECTION" conn = phoenixdb.connect(database_url, autocommit=True,

auth="SPNEGO") cursor = conn.cursor() cursor.execute("CREATE TABLE users (id INTEGER PRIMARY KEY, username VARCHAR)") cursor.execute("UPSERT INTO users VALUES (?, ?)", (1, 'admin')) cursor.execute("UPSERT INTO users VALUES (?, ?)", (2, 'user')) cursor.execute("SELECT * FROM users") print cursor.fetchall() ```` Replaces #307

2. **title:** PHOENIX-4688 Kerberize python phoenixdb -- Phoenix 5
   body: Lets rip out httplib and replace with requests and use requests kerberos #### Notes - Will not work out of the box, since requires virtualenv on anaconda - Excluded the IT from list, needs to be invoked manually `mvn -Dit.test=org.apache.phoenix.end2end.SecureQueryServerPhoenixDBIT verify` #### Standalone install This is currently a manual procedure with a few steps due to the current version not being in a public pipy and also the fact that we had to fork requests-kerberos 1. Create a python virtual environment by calling `conda create` or `virtual-env` 2. . activate the virtual environment 3. pip install -e phoenix/python/requests-kerberos 4. pip install -e phoenix/python/phoenixdb-module 5. You need kerberos credentials run kinit 6. Run a python program to access a DB similar to one below 7. Profit #### Example script ````python import phoenixdb import phoenixdb.cursor import sys if __name__ == '__main__': pqs_port = sys.argv[1] database_url = 'http://localhost:' + str(pqs_port) + '/' print "CREATING PQS CONNECTION" conn = phoenixdb.connect(database_url, autocommit=True, auth="SPNEGO") cursor = conn.cursor() cursor.execute("CREATE TABLE users (id INTEGER PRIMARY KEY, username VARCHAR)") cursor.execute("UPSERT INTO users VALUES (?, ?)", (1, 'admin')) cursor.execute("UPSERT INTO users VALUES (?, ?)", (2, 'user')) cursor.execute("SELECT * FROM users") print cursor.fetchall() ```` Replaces #307
3. **title:** PHOENIX-4688 Kerberize python phoenixdb -- Phoenix 5
   body: Lets rip out httplib and replace with requests and use requests kerberos #### Notes - Will not work out of the box, since requires virtualenv on anaconda - Excluded the IT from list, needs to be invoked manually `mvn -Dit.test=org.apache.phoenix.end2end.SecureQueryServerPhoenixDBIT verify` #### Standalone install This is currently a manual procedure with a few steps due to the current version not being in a public pipy and also the fact that we had to fork requests-kerberos 1. Create a python virtual environment by calling `conda create` or `virtual-env` 2. . activate the virtual environment 3. pip install -e phoenix/python/requests-kerberos 4. pip install -e phoenix/python/phoenixdb-module 5. You need kerberos credentials run kinit 6. Run a python program to access a DB similar to one below 7. Profit #### Example script ````python import phoenixdb import phoenixdb.cursor import sys if __name__ == '__main__': pqs_port = sys.argv[1] database_url = 'http://localhost:' + str(pqs_port) + '/' print "CREATING PQS CONNECTION" conn = phoenixdb.connect(database_url, autocommit=True, auth="SPNEGO") cursor = conn.cursor() cursor.execute("CREATE TABLE users (id INTEGER PRIMARY KEY, username VARCHAR)") cursor.execute("UPSERT INTO users VALUES (?, ?)", (1, 'admin')) cursor.execute("UPSERT INTO users VALUES (?, ?)", (2, 'user')) cursor.execute("SELECT * FROM users") print cursor.fetchall() ```` Replaces #307
4. **title:** PHOENIX-4688 Kerberize python phoenixdb -- Phoenix 5
   body: Lets rip out httplib and replace with requests and use requests kerberos #### Notes - Will not work out of the box, since requires virtualenv on anaconda - Excluded the IT from list, needs to be invoked manually `mvn -Dit.test=org.apache.phoenix.end2end.SecureQueryServerPhoenixDBIT verify` #### Standalone install This is currently a manual procedure with a few steps due to the current version not being in a public pipy and also the fact that we had to fork requests-kerberos 1. Create a python virtual environment by calling `conda create` or `virtual-env` 2. . activate the virtual environment 3. pip install -e phoenix/python/requests-kerberos 4. pip install -e phoenix/python/phoenixdb-module 5. You need kerberos credentials run kinit 6. Run a python program to access a DB similar to one below 7. Profit #### Example script ````python import phoenixdb import phoenixdb.cursor import sys if __name__ == '__main__': pqs_port = sys.argv[1] database_url = 'http://localhost:' + str(pqs_port) + '/' print "CREATING PQS CONNECTION" conn = phoenixdb.connect(database_url, autocommit=True, auth="SPNEGO") cursor = conn.cursor() cursor.execute("CREATE TABLE users (id INTEGER PRIMARY KEY, username VARCHAR)") cursor.execute("UPSERT INTO users VALUES (?, ?)", (1, 'admin')) cursor.execute("UPSERT INTO users VALUES (?, ?)", (2, 'user')) cursor.execute("SELECT * FROM users") print cursor.fetchall() ```` Replaces #307
5. **title:** PHOENIX-4688 Kerberize python phoenixdb -- Phoenix 5
   body: Lets rip out httplib and replace with requests and use requests kerberos #### Notes - Will not work out of the box, since requires virtualenv on anaconda - Excluded the IT from list, needs to be invoked manually `mvn -Dit.test=org.apache.phoenix.end2end.SecureQueryServerPhoenixDBIT verify` #### Standalone install This is currently a manual procedure with a few steps due to the current version not being in a public pipy and also the fact that we had to fork requests-kerberos 1. Create a python virtual environment by calling `conda create` or `virtual-env` 2. . activate the virtual environment 3. pip install -e phoenix/python/requests-kerberos 4. pip install -e phoenix/python/phoenixdb-module 5. You need kerberos credentials run kinit 6. Run a python program to access a DB similar to one below 7. Profit #### Example script ````python import phoenixdb import phoenixdb.cursor import sys if __name__ == '__main__': pqs_port = sys.argv[1] database_url = 'http://localhost:' + str(pqs_port) + '/' print "CREATING PQS CONNECTION" conn = phoenixdb.connect(database_url, autocommit=True, auth="SPNEGO") cursor = conn.cursor() cursor.execute("CREATE TABLE users (id INTEGER PRIMARY KEY, username VARCHAR)") cursor.execute("UPSERT INTO users VALUES (?, ?)", (1, 'admin'))

cursor.execute("UPSERT INTO users VALUES (?, ?)", (2, 'user')) cursor.execute("SELECT * FROM users") print cursor.fetchall() ```` Replaces #307

6. **title:** PHOENIX-4688 Kerberize python phoenixdb -- Phoenix 5
   body: Lets rip out httplib and replace with requests and use requests kerberos #### Notes - Will not work out of the box, since requires virtualenv on anaconda - Excluded the IT from list, needs to be invoked manually `mvn -Dit.test=org.apache.phoenix.end2end.SecureQueryServerPhoenixDBIT verify` #### Standalone install This is currently a manual procedure with a few steps due to the current version not being in a public pipy and also the fact that we had to fork requests-kerberos 1. Create a python virtual environment by calling `conda create` or `virtual-env` 2. . activate the virtual environment 3. pip install -e phoenix/python/requests-kerberos 4. pip install -e phoenix/python/phoenixdb-module 5. You need kerberos credentials run kinit 6. Run a python program to access a DB similar to one below 7. Profit #### Example script ````python import phoenixdb import phoenixdb.cursor import sys if __name__ == '__main__': pqs_port = sys.argv[1] database_url = 'http://localhost:' + str(pqs_port) + '/' print "CREATING PQS CONNECTION" conn = phoenixdb.connect(database_url, autocommit=True, auth="SPNEGO") cursor = conn.cursor() cursor.execute("CREATE TABLE users (id INTEGER PRIMARY KEY, username VARCHAR)") cursor.execute("UPSERT INTO users VALUES (?, ?)", (1, 'admin')) cursor.execute("UPSERT INTO users VALUES (?, ?)", (2, 'user')) cursor.execute("SELECT * FROM users") print cursor.fetchall() ```` Replaces #307
7. **title:** PHOENIX-4688 Kerberize python phoenixdb -- Phoenix 5
   body: Lets rip out httplib and replace with requests and use requests kerberos #### Notes - Will not work out of the box, since requires virtualenv on anaconda - Excluded the IT from list, needs to be invoked manually `mvn -Dit.test=org.apache.phoenix.end2end.SecureQueryServerPhoenixDBIT verify` #### Standalone install This is currently a manual procedure with a few steps due to the current version not being in a public pipy and also the fact that we had to fork requests-kerberos 1. Create a python virtual environment by calling `conda create` or `virtual-env` 2. . activate the virtual environment 3. pip install -e phoenix/python/requests-kerberos 4. pip install -e phoenix/python/phoenixdb-module 5. You need kerberos credentials run kinit 6. Run a python program to access a DB similar to one below 7. Profit #### Example script ````python import phoenixdb import phoenixdb.cursor import sys if __name__ == '__main__': pqs_port = sys.argv[1] database_url = 'http://localhost:' + str(pqs_port) + '/' print "CREATING PQS CONNECTION" conn = phoenixdb.connect(database_url, autocommit=True, auth="SPNEGO") cursor = conn.cursor() cursor.execute("CREATE TABLE users (id INTEGER PRIMARY KEY, username VARCHAR)") cursor.execute("UPSERT INTO users VALUES (?, ?)", (1, 'admin')) cursor.execute("UPSERT INTO users VALUES (?, ?)", (2, 'user')) cursor.execute("SELECT * FROM users") print cursor.fetchall() ```` Replaces #307
8. **title:** PHOENIX-4688 Kerberize python phoenixdb -- Phoenix 5
   body: Lets rip out httplib and replace with requests and use requests kerberos #### Notes - Will not work out of the box, since requires virtualenv on anaconda - Excluded the IT from list, needs to be invoked manually `mvn -Dit.test=org.apache.phoenix.end2end.SecureQueryServerPhoenixDBIT verify` #### Standalone install This is currently a manual procedure with a few steps due to the current version not being in a public pipy and also the fact that we had to fork requests-kerberos 1. Create a python virtual environment by calling `conda create` or `virtual-env` 2. . activate the virtual environment 3. pip install -e phoenix/python/requests-kerberos 4. pip install -e phoenix/python/phoenixdb-module 5. You need kerberos credentials run kinit 6. Run a python program to access a DB similar to one below 7. Profit #### Example script ````python import phoenixdb import phoenixdb.cursor import sys if __name__ == '__main__': pqs_port = sys.argv[1] database_url = 'http://localhost:' + str(pqs_port) + '/' print "CREATING PQS CONNECTION" conn = phoenixdb.connect(database_url, autocommit=True, auth="SPNEGO") cursor = conn.cursor() cursor.execute("CREATE TABLE users (id INTEGER PRIMARY KEY, username VARCHAR)") cursor.execute("UPSERT INTO users VALUES (?, ?)", (1, 'admin')) cursor.execute("UPSERT INTO users VALUES (?, ?)", (2, 'user')) cursor.execute("SELECT * FROM users") print cursor.fetchall() ```` Replaces #307
9. **title:** PHOENIX-4688 Kerberize python phoenixdb -- Phoenix 5
   body: Lets rip out httplib and replace with requests and use requests kerberos #### Notes - Will not work out of the box, since requires virtualenv on anaconda - Excluded the IT from list, needs to be invoked manually `mvn -Dit.test=org.apache.phoenix.end2end.SecureQueryServerPhoenixDBIT verify` #### Standalone install This is currently a manual procedure with a few steps due to the current version not being in a public pipy and also the fact that we had to fork requests-kerberos 1. Create a python virtual environment by calling `conda create` or `virtual-env` 2. . activate the virtual environment 3. pip install -e phoenix/python/requests-kerberos 4. pip install -e phoenix/python/phoenixdb-module 5. You need kerberos credentials run kinit 6. Run a python program to access a DB similar to one below 7. Profit #### Example script ````python import phoenixdb import phoenixdb.cursor import sys if __name__ == '__main__': pqs_port = sys.argv[1] database_url = 'http://localhost:' + str(pqs_port) + '/' print "CREATING PQS CONNECTION" conn = phoenixdb.connect(database_url, autocommit=True, auth="SPNEGO") cursor = conn.cursor() cursor.execute("CREATE TABLE users (id INTEGER PRIMARY KEY, username VARCHAR)") cursor.execute("UPSERT INTO users VALUES (?, ?)", (1, 'admin')) cursor.execute("UPSERT INTO users VALUES (?, ?)", (2, 'user')) cursor.execute("SELECT * FROM users") print cursor.fetchall() ```` Replaces #307
10. **title:** PHOENIX-4688 Kerberize python phoenixdb -- Phoenix 5

body: Lets rip out httplib and replace with requests and use requests kerberos #### Notes - Will not work out of the box, since requires virtualenv on anaconda - Excluded the IT from list, needs to be invoked manually `mvn -Dit.test=org.apache.phoenix.end2end.SecureQueryServerPhoenixDBIT verify` #### Standalone install This is currently a manual procedure with a few steps due to the current version not being in a public pipy and also the fact that we had to fork requests-kerberos 1. Create a python virtual environment by calling `conda create` or `virtual-env` 2. . activate the virtual environment 3. pip install -e phoenix/python/requests-kerberos 4. pip install -e phoenix/python/phoenixdb-module 5. You need kerberos credentials run kinit 6. Run a python program to access a DB similar to one below 7. Profit #### Example script ````python import phoenixdb import phoenixdb.cursor import sys if __name__ == '__main__': pqs_port = sys.argv[1] database_url = 'http://localhost:' + str(pqs_port) + '/' print "CREATING PQS CONNECTION" conn = phoenixdb.connect(database_url, autocommit=True, auth="SPNEGO") cursor = conn.cursor() cursor.execute("CREATE TABLE users (id INTEGER PRIMARY KEY, username VARCHAR)") cursor.execute("UPSERT INTO users VALUES (?, ?)", (1, 'admin')) cursor.execute("UPSERT INTO users VALUES (?, ?)", (2, 'user')) cursor.execute("SELECT * FROM users") print cursor.fetchall() ```` Replaces #307

11. **title:** PHOENIX-4688 Kerberize python phoenixdb -- Phoenix 5
    body: Lets rip out httplib and replace with requests and use requests kerberos #### Notes - Will not work out of the box, since requires virtualenv on anaconda - Excluded the IT from list, needs to be invoked manually `mvn -Dit.test=org.apache.phoenix.end2end.SecureQueryServerPhoenixDBIT verify` #### Standalone install This is currently a manual procedure with a few steps due to the current version not being in a public pipy and also the fact that we had to fork requests-kerberos 1. Create a python virtual environment by calling `conda create` or `virtual-env` 2. . activate the virtual environment 3. pip install -e phoenix/python/requests-kerberos 4. pip install -e phoenix/python/phoenixdb-module 5. You need kerberos credentials run kinit 6. Run a python program to access a DB similar to one below 7. Profit #### Example script ````python import phoenixdb import phoenixdb.cursor import sys if __name__ == '__main__': pqs_port = sys.argv[1] database_url = 'http://localhost:' + str(pqs_port) + '/' print "CREATING PQS CONNECTION" conn = phoenixdb.connect(database_url, autocommit=True, auth="SPNEGO") cursor = conn.cursor() cursor.execute("CREATE TABLE users (id INTEGER PRIMARY KEY, username VARCHAR)") cursor.execute("UPSERT INTO users VALUES (?, ?)", (1, 'admin')) cursor.execute("UPSERT INTO users VALUES (?, ?)", (2, 'user')) cursor.execute("SELECT * FROM users") print cursor.fetchall() ```` Replaces #307
12. **title:** PHOENIX-4688 Kerberize python phoenixdb -- Phoenix 5
    body: Lets rip out httplib and replace with requests and use requests kerberos #### Notes - Will not work out of the box, since requires virtualenv on anaconda - Excluded the IT from list, needs to be invoked manually `mvn -Dit.test=org.apache.phoenix.end2end.SecureQueryServerPhoenixDBIT verify` #### Standalone install This is currently a manual procedure with a few steps due to the current version not being in a public pipy and also the fact that we had to fork requests-kerberos 1. Create a python virtual environment by calling `conda create` or `virtual-env` 2. . activate the virtual environment 3. pip install -e phoenix/python/requests-kerberos 4. pip install -e phoenix/python/phoenixdb-module 5. You need kerberos credentials run kinit 6. Run a python program to access a DB similar to one below 7. Profit #### Example script ````python import phoenixdb import phoenixdb.cursor import sys if __name__ == '__main__': pqs_port = sys.argv[1] database_url = 'http://localhost:' + str(pqs_port) + '/' print "CREATING PQS CONNECTION" conn = phoenixdb.connect(database_url, autocommit=True, auth="SPNEGO") cursor = conn.cursor() cursor.execute("CREATE TABLE users (id INTEGER PRIMARY KEY, username VARCHAR)") cursor.execute("UPSERT INTO users VALUES (?, ?)", (1, 'admin')) cursor.execute("UPSERT INTO users VALUES (?, ?)", (2, 'user')) cursor.execute("SELECT * FROM users") print cursor.fetchall() ```` Replaces #307
13. **title:** PHOENIX-4688 Kerberize python phoenixdb -- Phoenix 5
    body: Lets rip out httplib and replace with requests and use requests kerberos #### Notes - Will not work out of the box, since requires virtualenv on anaconda - Excluded the IT from list, needs to be invoked manually `mvn -Dit.test=org.apache.phoenix.end2end.SecureQueryServerPhoenixDBIT verify` #### Standalone install This is currently a manual procedure with a few steps due to the current version not being in a public pipy and also the fact that we had to fork requests-kerberos 1. Create a python virtual environment by calling `conda create` or `virtual-env` 2. . activate the virtual environment 3. pip install -e phoenix/python/requests-kerberos 4. pip install -e phoenix/python/phoenixdb-module 5. You need kerberos credentials run kinit 6. Run a python program to access a DB similar to one below 7. Profit #### Example script ````python import phoenixdb import phoenixdb.cursor import sys if __name__ == '__main__': pqs_port = sys.argv[1] database_url = 'http://localhost:' + str(pqs_port) + '/' print "CREATING PQS CONNECTION" conn = phoenixdb.connect(database_url, autocommit=True, auth="SPNEGO") cursor = conn.cursor() cursor.execute("CREATE TABLE users (id INTEGER PRIMARY KEY, username VARCHAR)") cursor.execute("UPSERT INTO users VALUES (?, ?)", (1, 'admin')) cursor.execute("UPSERT INTO users VALUES (?, ?)", (2, 'user')) cursor.execute("SELECT * FROM users") print cursor.fetchall() ```` Replaces #307
14. **title:** PHOENIX-4688 Kerberize python phoenixdb -- Phoenix 5
    body: Lets rip out httplib and replace with requests and use requests kerberos #### Notes - Will not work out of the box, since requires virtualenv on anaconda - Excluded the IT from list, needs to be invoked manually `mvn -Dit.test=org.apache.phoenix.end2end.SecureQueryServerPhoenixDBIT verify` #### Standalone install This is

currently a manual procedure with a few steps due to the current version not being in a public pipy and also the fact that we had to fork requests-kerberos 1. Create a python virtual environment by calling `conda create` or `virtual-env` 2. . activate the virtual environment 3. pip install -e phoenix/python/requests-kerberos 4. pip install -e phoenix/python/phoenixdb-module 5. You need kerberos credentials run kinit 6. Run a python program to access a DB similar to one below 7. Profit #### Example script ````python import phoenixdb import phoenixdb.cursor import sys if __name__ == '__main__': pqs_port = sys.argv[1] database_url = 'http://localhost:' + str(pqs_port) + '/' print "CREATING PQS CONNECTION" conn = phoenixdb.connect(database_url, autocommit=True, auth="SPNEGO") cursor = conn.cursor() cursor.execute("CREATE TABLE users (id INTEGER PRIMARY KEY, username VARCHAR)") cursor.execute("UPSERT INTO users VALUES (?, ?)", (1, 'admin')) cursor.execute("UPSERT INTO users VALUES (?, ?)", (2, 'user')) cursor.execute("SELECT * FROM users") print cursor.fetchall() ```` Replaces #307

github_pulls_comments:

1. This looks awesome, @pu239ppy. Testing it out.
2. > This looks awesome, @pu239ppy. Testing it out. Hi @joshelser , did you finally managed to test it out successfully?
3. @akhmadMizkat this solution currently does not work, please see latest commentary on PHOENIX-4688 for description. There are existing solutions to remediate this, however I was hoping not to distibute a patched version of requests-kerberos
4. @pu239ppy thank you for the clarification. Hope they can accept your PR on the requests-kerberos. Really looking forward for that.
5. > Hi @joshelser , did you finally managed to test it out successfully? No, sadly. I didn't get the last variant working. Pulling down this version and trying again.
6. ``` 2018-09-27 14:05:06,308 ERROR [main] end2end.SecureQueryServerPhoenixDBIT(358): + python /Users/jelser/projects/phoenix.git/phoenix-queryserver/./src/it/bin/test_phoenixdb.py 50023 2018-09-27 14:05:06,309 ERROR [main] end2end.SecureQueryServerPhoenixDBIT(358): /Users/jelser/projects/phoenix.git/phoenix-queryserver/./src/it/bin/test_phoenixdb.sh: line 60: 7582 Illegal instruction: 4 python $PYTHON_SCRIPT $PQS_PORT ``` We're back to this junk again. @pu239ppy what version of Python are you running for which this is working?
7. Switched to Python 3.6.5 and now hitting https://github.com/02strich/pykerberos/issues/37
8. Ok, with python 3.6.5, downgrading to pykerberos to 1.1.14 (in requests-kerberos), and updating test_phoenixdb.py for python3 print syntax, it works!!!
9. Oops, forgot one more thing. I changed the integration test that had heimdal syntax because, even though I have OSX, I have MIT Kerberos installed. Although, now that I think about it, do we need to generate a specific krb5.conf based on OS at all? The embedded KDC is provided by Kerby from Apache Directory -- it's not Heimdal or MIT Kerberos. That can be cleaned up completely. @pu239ppy let me know. I can send a pull request to your branch if you're ready to just be done with these changes. I want to poke around some more at Python versions that work locally.
10. Awesome guys. This is a very good improvement for the phoenixdb library.
11. Python version has to be >= 3.6.5? @joshelser @pu239ppy
12. Hey @Reidddddd -- likely other version of Python will work too, we just only tested using 3.6.5

github_pulls_reviews:

jira_issues:

1. **summary:** Add kerberos authentication to python-phoenixdb
   description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
2. **summary:** Add kerberos authentication to python-phoenixdb
   description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
3. **summary:** Add kerberos authentication to python-phoenixdb
   description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
   label: code-design
4. **summary:** Add kerberos authentication to python-phoenixdb
   description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
5. **summary:** Add kerberos authentication to python-phoenixdb
   description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
6. **summary:** Add kerberos authentication to python-phoenixdb

description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.

7. **summary:** Add kerberos authentication to python-phoenixdb
   description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
8. **summary:** Add kerberos authentication to python-phoenixdb
   description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
9. **summary:** Add kerberos authentication to python-phoenixdb
   description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
10. **summary:** Add kerberos authentication to python-phoenixdb
    description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
11. **summary:** Add kerberos authentication to python-phoenixdb
    description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
12. **summary:** Add kerberos authentication to python-phoenixdb
    description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
13. **summary:** Add kerberos authentication to python-phoenixdb
    description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
14. **summary:** Add kerberos authentication to python-phoenixdb
    description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
15. **summary:** Add kerberos authentication to python-phoenixdb
    description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
16. **summary:** Add kerberos authentication to python-phoenixdb
    description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
17. **summary:** Add kerberos authentication to python-phoenixdb
    description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
18. **summary:** Add kerberos authentication to python-phoenixdb
    description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
19. **summary:** Add kerberos authentication to python-phoenixdb
    description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
    label: code-design
20. **summary:** Add kerberos authentication to python-phoenixdb
    description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
    label: code-design
21. **summary:** Add kerberos authentication to python-phoenixdb
    description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
22. **summary:** Add kerberos authentication to python-phoenixdb
    description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
    label: code-design
23. **summary:** Add kerberos authentication to python-phoenixdb
    description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
24. **summary:** Add kerberos authentication to python-phoenixdb
    description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
25. **summary:** Add kerberos authentication to python-phoenixdb
    description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
26. **summary:** Add kerberos authentication to python-phoenixdb

description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
label: code-design

27. **summary:** Add kerberos authentication to python-phoenixdb
    description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
28. **summary:** Add kerberos authentication to python-phoenixdb
    description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
29. **summary:** Add kerberos authentication to python-phoenixdb
    description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
30. **summary:** Add kerberos authentication to python-phoenixdb
    description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
31. **summary:** Add kerberos authentication to python-phoenixdb
    description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
32. **summary:** Add kerberos authentication to python-phoenixdb
    description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
33. **summary:** Add kerberos authentication to python-phoenixdb
    description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
34. **summary:** Add kerberos authentication to python-phoenixdb
    description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
35. **summary:** Add kerberos authentication to python-phoenixdb
    description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
36. **summary:** Add kerberos authentication to python-phoenixdb
    description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
37. **summary:** Add kerberos authentication to python-phoenixdb
    description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
38. **summary:** Add kerberos authentication to python-phoenixdb
    description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
39. **summary:** Add kerberos authentication to python-phoenixdb
    description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
40. **summary:** Add kerberos authentication to python-phoenixdb
    description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
41. **summary:** Add kerberos authentication to python-phoenixdb
    description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
42. **summary:** Add kerberos authentication to python-phoenixdb
    description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
    label: code-design
43. **summary:** Add kerberos authentication to python-phoenixdb
    description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
44. **summary:** Add kerberos authentication to python-phoenixdb
    description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
45. **summary:** Add kerberos authentication to python-phoenixdb
    description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
46. **summary:** Add kerberos authentication to python-phoenixdb

description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.

47. **summary:** Add kerberos authentication to python-phoenixdb
    description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
48. **summary:** Add kerberos authentication to python-phoenixdb
    description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
49. **summary:** Add kerberos authentication to python-phoenixdb
    description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
50. **summary:** Add kerberos authentication to python-phoenixdb
    description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
51. **summary:** Add kerberos authentication to python-phoenixdb
    description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
52. **summary:** Add kerberos authentication to python-phoenixdb
    description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
53. **summary:** Add kerberos authentication to python-phoenixdb
    description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
54. **summary:** Add kerberos authentication to python-phoenixdb
    description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
55. **summary:** Add kerberos authentication to python-phoenixdb
    description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
56. **summary:** Add kerberos authentication to python-phoenixdb
    description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
    label: code-design
57. **summary:** Add kerberos authentication to python-phoenixdb
    description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
58. **summary:** Add kerberos authentication to python-phoenixdb
    description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
    label: code-design
59. **summary:** Add kerberos authentication to python-phoenixdb
    description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
60. **summary:** Add kerberos authentication to python-phoenixdb
    description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
61. **summary:** Add kerberos authentication to python-phoenixdb
    description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
62. **summary:** Add kerberos authentication to python-phoenixdb
    description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
63. **summary:** Add kerberos authentication to python-phoenixdb
    description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
    label: code-design
64. **summary:** Add kerberos authentication to python-phoenixdb
    description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
65. **summary:** Add kerberos authentication to python-phoenixdb
    description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
66. **summary:** Add kerberos authentication to python-phoenixdb

description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.

67. **summary:** Add kerberos authentication to python-phoenixdb
    description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
68. **summary:** Add kerberos authentication to python-phoenixdb
    description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
69. **summary:** Add kerberos authentication to python-phoenixdb
    description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
    label: test
70. **summary:** Add kerberos authentication to python-phoenixdb
    description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
71. **summary:** Add kerberos authentication to python-phoenixdb
    description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
72. **summary:** Add kerberos authentication to python-phoenixdb
    description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
73. **summary:** Add kerberos authentication to python-phoenixdb
    description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
74. **summary:** Add kerberos authentication to python-phoenixdb
    description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
75. **summary:** Add kerberos authentication to python-phoenixdb
    description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
76. **summary:** Add kerberos authentication to python-phoenixdb
    description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
77. **summary:** Add kerberos authentication to python-phoenixdb
    description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
78. **summary:** Add kerberos authentication to python-phoenixdb
    description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
79. **summary:** Add kerberos authentication to python-phoenixdb
    description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
80. **summary:** Add kerberos authentication to python-phoenixdb
    description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
81. **summary:** Add kerberos authentication to python-phoenixdb
    description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
82. **summary:** Add kerberos authentication to python-phoenixdb
    description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.
83. **summary:** Add kerberos authentication to python-phoenixdb
    description: In its current state python-phoenixdv does not support support kerberos authentication. Using a modern python http library such as requests or urllib it would be simple (if not trivial) to add this support.

jira_issues_comments:

1. I currently have some code in progress at [https://github.com/pu239ppy/python-phoenixdb] There are two issues with it 1. I started working on this before I became aware that apache/phoenix has assumed ownership of python-phoenixdb from Lucas 2. In its current state the code relies on the fact that [https://github.com/requests/requests-kerberos/pull/115] will be merged, as an alternative to this there is [https://github.com/requests/requests-kerberos/pull/89] a much more ambitious PR, but one that is already a year with no merge. An alternative exists as urllib, however the same person who filed #89 for requests-kerberos has also filed [https://github.com/willthames/urllib_kerberos/pull/1] around the same time and this had also gone un merged.

2. Since it is security related, I think this is a critical improvement.
3. **body:** Agreed, Reid. This one is important to make this library that much more useful. {quote}2. In its current state the code relies on the fact that [https://github.com/requests/requests-kerberos/pull/115] will be merged, as an alternative to this there is [https://github.com/requests/requests-kerberos/pull/89] a much more ambitious PR, but one that is already a year with no merge. {quote} If we can't get traction from upstream (not sure if we just need to pester, or what), can we bundle these locally for the interim? I don't think there are any licensing concerns, but I'm not familiar with "standard ways" to do this for Python projects (akin to hygiene for shading dependencies in JARs for Java).
   label: code-design
4. We can bundle a fork and run "our_requests-kerberos/setup.py install" this will work for phoenix installs. However this is not an MO we want to support as we push pythondb-phoenix to pypi.python.org
5. {quote}However this is not an MO we want to support as we push pythondb-phoenix to pypi.python.org {quote} Understood and agree. I'm still not sure how we do that for ASF projects. Need to do more research to figure out how we properly deploy binaries as a part of the release cycle. For now, I'm just shooting for "source-releases" and accepting the burden on users.
6. Another potential issue with this approach is that some user may at some point override our version of requests-kerberos with a newer one. This can be avoided if a local phoenix install decides to have its own python environment
7. Still trying to get in touch with maintainers. Got a few redirects I am following up on
8. It looks like we may have to for requests-kerberos as no one has gotten back to me. Any suggestions as to how we should go about this? At this point I don't know if we want to own this for python community at large and tis probably precludes us from doing a public fork and a pypi module. This is perhaps something we can name requests-kerberos-phoenix and will be installed into the specified runtime/virtualenv at install time.
9. {quote}This is perhaps something we can name requests-kerberos-phoenix and will be installed into the specified runtime/virtualenv at install time. {quote} This sounds like the easiest route to me.
10. I need some advise on structural changes. We can do one of the following things # in the python directory [https://github.com/apache/phoenix/tree/master/python] create a subdirectory for python-phoenixdb and one for our patched version of python requests-kerberos.   # Make a private fork of requests kerberos on github as in github/pu239ppy or github/jelser and use that Once the fork is made will make sure that every instance of requests_kerberos and requests-kerberos is renamed into phoenix-requests_kerberos and phoenix-requests-kerberos. The readme file will explicitly state that this is a fork and is only here until requests kerberos get back to us about merging
11. One more update, someone did finally look at [https://github.com/requests/requests-kerberos/pull/89,] however deemed it too complicated to be merged and asked for someone else to step up. I asked in turn to consider my PR [https://github.com/requests/requests-kerberos/pull/89|https://github.com/requests/requests-kerberos/pull/89,], which essentially accomplishes the same thing but with less code change
12. I am rearranging the python directory to add mode modules https://github.com/pu239ppy/phoenix/tree/PHOENIX-4688
13. Added my changes on top
14. I have yet to look into renaming the forked requests-kerberos modules we should see for now if it works in a python virtual environment and then make this decision. I am hoping that the maintainers will eventually get back to me on this
15. {quote}I am hoping that the maintainers will eventually get back to me on this {quote} Ditto ;) {quote}we should see for now if it works in a python virtual environment {quote} I wanted to know what you thought the "workflow" would be for a user to build the python driver. Install the custom kerberos-requests library (maybe in a virtualenv or to their local python installation) and then do the normal phoenixdb build on its own?
16. Currently this would be a manual where a user would be forced to spin up a virtual environment go to first install requests-kerberos by going to requests-kerberos-fork-module and running setup.py install. Following that going to phoenixdb-module and running setup.py install. Using a virtual environment would ensure that our kerberos-requests for does not conflict with any previously installed kerberos-requests module (I suppose I should also bump the version number to ensure that, alternatively do a full rename). If a user is careful then a virtual environment is not necessarily needed. In addition we do a rename and push both modules into pipy, so a simple pip install would do what is needed. Is there any ASF governance regarding the latter option?
17. Rather then renaming a package I decided to change the version string -__version__ = '0.13.0.dev0' +__version__ = '0.13.0.dev0-phoenixdb'
18. installing from source directly ensures that only extra packages are downloaded pip install file:///Users/lbronshtein/DEV/phoenix/python/requests-kerberos >>> import requests_kerberos >>> requests_kerberos.__version__ '0.13.0.dev0-phoenixdb'   So we can just install requests-kerberos locally and the phoenixdb
19. **body:** I am closing in on the integration test. Currently have the following issues * Unable to connect to miniKDC and perform a KINIT even after exporting KRB5_CONFIG to be whatever the miniKDC dumped 2018-06-27 08:51:26,641 INFO [main] end2end.SecureQueryServerPhoenixDBIT(312): [libdefaults] 2018-06-27 08:51:26,641 INFO [main] end2end.SecureQueryServerPhoenixDBIT(312): default_realm = EXAMPLE.COM

2018-06-27 08:51:26,641 INFO [main] end2end.SecureQueryServerPhoenixDBIT(312): udp_preference_limit = 1 2018-06-27 08:51:26,641 INFO [main] end2end.SecureQueryServerPhoenixDBIT(312): 2018-06-27 08:51:26,641 INFO [main] end2end.SecureQueryServerPhoenixDBIT(312): [realms] 2018-06-27 08:51:26,642 INFO [main] end2end.SecureQueryServerPhoenixDBIT(312): EXAMPLE.COM = { 2018-06-27 08:51:26,642 INFO [main] end2end.SecureQueryServerPhoenixDBIT(312): kdc = localhost:54339 2018-06-27 08:51:26,642 INFO [main] end2end.SecureQueryServerPhoenixDBIT(312): } 2018-06-27 08:51:26,642 INFO [main] end2end.SecureQueryServerPhoenixDBIT(312): RUNNING KINIT 2018-06-27 08:51:26,650 ERROR [main] end2end.SecureQueryServerPhoenixDBIT(316): + kinit -kt /Users/lbronshtein/DEV/phoenix/phoenix-queryserver/target/SecureQueryServerPhoenixDBIT/keytabs/user1.keytab user1@EXAMPLE.COM 2018-06-27 08:51:26,650 ERROR [main] end2end.SecureQueryServerPhoenixDBIT(316): kinit: krb5_get_init_creds: unable to reach any KDC in realm EXAMPLE.COM, tried 1 KDC 2018-06-27 08:51:26,650 ERROR [main] end2end.SecureQueryServerPhoenixDBIT(316): + cleanup * There are a few places in the IT and the script it launches that hardcode paths so I need some pointers on figuring out paths relative to the IT test or perhaps another work around
label: code-design

20. **body:** Enabling KRB5_DEBUG it looks like only UDP is being tried 2018-06-27 10:28:22,731 INFO [main] end2end.SecureQueryServerPhoenixDBIT(308): 2018-06-27T10:28:22 set-error: -1765328242: Reached end of credential caches 2018-06-27 10:28:22,731 INFO [main] end2end.SecureQueryServerPhoenixDBIT(308): 2018-06-27T10:28:22 set-error: -1765328243: Principal user1@EXAMPLE.COM not found in any credential cache 2018-06-27 10:28:22,731 INFO [main] end2end.SecureQueryServerPhoenixDBIT(308): 2018-06-27T10:28:22 set-error: -1765328234: Encryption type des-cbc-md5-deprecated not supported 2018-06-27 10:28:22,731 INFO [main] end2end.SecureQueryServerPhoenixDBIT(308): 2018-06-27T10:28:22 Adding PA mech: ENCRYPTED_CHALLENGE 2018-06-27 10:28:22,731 INFO [main] end2end.SecureQueryServerPhoenixDBIT(308): 2018-06-27T10:28:22 Adding PA mech: ENCRYPTED_TIMESTAMP 2018-06-27 10:28:22,731 INFO [main] end2end.SecureQueryServerPhoenixDBIT(308): 2018-06-27T10:28:22 krb5_get_init_creds: loop 1 2018-06-27 10:28:22,731 INFO [main] end2end.SecureQueryServerPhoenixDBIT(308): 2018-06-27T10:28:22 KDC sent 0 patypes 2018-06-27 10:28:22,731 INFO [main] end2end.SecureQueryServerPhoenixDBIT(308): 2018-06-27T10:28:22 fast disabled, not doing any fast wrapping 2018-06-27 10:28:22,731 INFO [main] end2end.SecureQueryServerPhoenixDBIT(308): 2018-06-27T10:28:22 Trying to find service kdc for realm EXAMPLE.COM flags 0 2018-06-27 10:28:22,731 INFO [main] end2end.SecureQueryServerPhoenixDBIT(308): 2018-06-27T10:28:22 configuration file for realm EXAMPLE.COM found 2018-06-27 10:28:22,731 INFO [main] end2end.SecureQueryServerPhoenixDBIT(308): 2018-06-27T10:28:22 submissing new requests to new host 2018-06-27 10:28:22,731 INFO [main] end2end.SecureQueryServerPhoenixDBIT(308): 2018-06-27T10:28:22 host_create: setting hostname localhost 2018-06-27 10:28:22,731 INFO [main] end2end.SecureQueryServerPhoenixDBIT(308): 2018-06-27T10:28:22 connecting to host: udp ::1:56481 (localhost) tid: 00000001 2018-06-27 10:28:22,732 INFO [main] end2end.SecureQueryServerPhoenixDBIT(308): 2018-06-27T10:28:22 host_create: setting hostname localhost 2018-06-27 10:28:22,732 INFO [main] end2end.SecureQueryServerPhoenixDBIT(308): 2018-06-27T10:28:22 Queuing host in future (in 3s), its the 2 address on the same name: udp 127.0.0.1:56481 (localhost) tid: 00000002 2018-06-27 10:28:22,732 INFO [main] end2end.SecureQueryServerPhoenixDBIT(308): 2018-06-27T10:28:22 writing packet: udp ::1:56481 (localhost) tid: 00000001 2018-06-27 10:28:22,732 INFO [main] end2end.SecureQueryServerPhoenixDBIT(308): 2018-06-27T10:28:22 reading packet: udp ::1:56481 (localhost) tid: 00000001 2018-06-27 10:28:22,732 INFO [main] end2end.SecureQueryServerPhoenixDBIT(308): 2018-06-27T10:28:22 failed to get nbytes from socket, no bytes there?: udp ::1:56481 (localhost) tid: 00000001 2018-06-27 10:28:22,732 INFO [main] end2end.SecureQueryServerPhoenixDBIT(308): 2018-06-27T10:28:22 host disconnected: udp ::1:56481 (localhost) tid: 00000001 2018-06-27 10:28:22,732 INFO [main] end2end.SecureQueryServerPhoenixDBIT(308): 2018-06-27T10:28:22 connecting to host: udp 127.0.0.1:56481 (localhost) tid: 00000002 2018-06-27 10:28:22,732 INFO [main] end2end.SecureQueryServerPhoenixDBIT(308): 2018-06-27T10:28:22 writing packet: udp 127.0.0.1:56481 (localhost) tid: 00000002 2018-06-27 10:28:22,732 INFO [main] end2end.SecureQueryServerPhoenixDBIT(308): 2018-06-27T10:28:22 reading packet: udp 127.0.0.1:56481 (localhost) tid: 00000002 2018-06-27 10:28:22,732 INFO [main] end2end.SecureQueryServerPhoenixDBIT(308): 2018-06-27T10:28:22 failed to get nbytes from socket, no bytes there?: udp 127.0.0.1:56481 (localhost) tid: 00000002 2018-06-27 10:28:22,732 INFO [main] end2end.SecureQueryServerPhoenixDBIT(308): 2018-06-27T10:28:22 host disconnected: udp 127.0.0.1:56481 (localhost) tid: 00000002 2018-06-27 10:28:22,732 INFO [main] end2end.SecureQueryServerPhoenixDBIT(308): 2018-06-27T10:28:22 no more hosts to send/recv packets to/from and no more hosts -> failure 2018-06-27 10:28:22,732 INFO [main] end2end.SecureQueryServerPhoenixDBIT(308): 2018-06-27T10:28:22 Configuration exists for realm EXAMPLE.COM, wont go to DNS 2018-06-27 10:28:22,732 INFO [main] end2end.SecureQueryServerPhoenixDBIT(308): 2018-06-27T10:28:22 out of hosts, waiting for replies 2018-06-27 10:28:22,732 INFO [main] end2end.SecureQueryServerPhoenixDBIT(308): 2018-06-27T10:28:22 no more hosts to send/recv packets to/from trying to pulling more hosts 2018-06-27 10:28:22,732 INFO [main] end2end.SecureQueryServerPhoenixDBIT(308): 2018-06-27T10:28:22 set-error: -1765328228: unable to reach

any KDC in realm EXAMPLE.COM, tried 1 KDC 2018-06-27 10:28:22,733 INFO [main] end2end.SecureQueryServerPhoenixDBIT(308): 2018-06-27T10:28:22 krb5_sendto_context EXAMPLE.COM done: -1765328228 hosts 1 packets 2 wc: 0.099881 nr: 0.001117 kh: 0.004248 tid: 00000002
label: code-design
21. Having to render a custom KDC entry kdc = tcp/localhost:56819
22. **body:** Test is working 2018-06-27 12:52:15,048 INFO [main] end2end.SecureQueryServerPhoenixDBIT(310): CREATING PQS CONNECTION 2018-06-27 12:52:15,048 INFO [main] end2end.SecureQueryServerPhoenixDBIT(310): [[1, u'admin'], [2, u'user']] now just need to clean it up. * -How do I pass down proxy settings? Or should I assume no proxy?- Inherited from callers shell * -The heimdal kerberos utilities apple ships does not support krb5.conf format miniKDC ships (minor variations)- Render a custom krb5.conf if MAC * Currently a shell script is needed to launch. I tried taking out as much as possible but I still need to kinit, also it needs to source activate script * -Do we care if it only works on Linux?- It will work on MAC OS too * -currently only works with Anaconda, would rather see it with virtualenv,- though neither one comes pre installed on stock MAC OS
label: code-design
23. This will require a very large patch, if anyone would like to start reviewing this, I opened https://github.com/apache/phoenix/pull/307
24. Looking at this today. {quote}Currently a shell script is needed to launch. I tried taking out as much as possible but I still need to kinit, also it needs to source activate script {quote} Can you point me to an example of this? Would like to understand it, but don't see a problem at first blush.
25. Create and activate environment here [https://github.com/apache/phoenix/pull/307/files#diff-0d0a748959965a7cfdc725f33414d1c0R30]   KINIT here [https://github.com/apache/phoenix/pull/307/files#diff-0d0a748959965a7cfdc725f33414d1c0R50]   There are minor improvements from the first take where the python script and krb5.conf both started out as heredocs inside the shell script. I attempted to pull kinit into JAVA as well and pass the environment around but # I failed to make this work # executing various shell commands from java adds a lot of bloat # I don't even know how I would source a script and then pass the resulting shell modifications onto the next one in java Having encountered #3, I gave up on further shell script pruning/elimination   Just realized that that I never transitioned from conda to virtualenv. I am going to attempt to support both, but for now there is probably enough to look at.
26. Github user joshelser commented on a diff in the pull request: https://github.com/apache/phoenix/pull/307#discussion_r201076404 --- Diff: phoenix-queryserver/src/it/bin/test_phoenixdb.sh --- @@ -0,0 +1,59 @@ +#/usr/bin/env bash + +set -u +set -x +set -e + +function cleanup { + set +e + set +u + kdestroy --- End diff -- If we use a custom directory for the `kinit`, then this just becomes removing that custom directory.
27. Github user joshelser commented on a diff in the pull request: https://github.com/apache/phoenix/pull/307#discussion_r201076875 --- Diff: phoenix-queryserver/src/it/java/org/apache/phoenix/end2end/SecureQueryServerPhoenixDBIT.java --- @@ -0,0 +1,423 @@ +/* + * Licensed to the Apache Software Foundation (ASF) under one or more + * contributor license agreements. See the NOTICE file distributed with + * this work for additional information regarding copyright ownership. + * The ASF licenses this file to you under the Apache License, Version 2.0 + * (the "License"); you may not use this file except in compliance with + * the License. You may obtain a copy of the License at + * + * http://www.apache.org/licenses/LICENSE-2.0 + * + * Unless required by applicable law or agreed to in writing, software + * distributed under the License is distributed on an "AS IS" BASIS, + * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. + * See the License for the specific language governing permissions and + * limitations under the License. + */ +package org.apache.phoenix.end2end; + +import com.google.common.base.Preconditions; +import com.google.common.collect.Maps; +import org.apache.commons.io.FileUtils; +import org.apache.commons.logging.Log; +import org.apache.commons.logging.LogFactory; +import org.apache.hadoop.conf.Configuration; +import org.apache.hadoop.fs.Path; +import org.apache.hadoop.hbase.HBaseTestingUtility; +import org.apache.hadoop.hbase.HConstants; +import org.apache.hadoop.hbase.LocalHBaseCluster; +import org.apache.hadoop.hbase.coprocessor.CoprocessorHost; +import org.apache.hadoop.hbase.http.ssl.KeyStoreTestUtil; +import org.apache.hadoop.hbase.security.HBaseKerberosUtils; +import org.apache.hadoop.hbase.security.token.TokenProvider; +import org.apache.hadoop.hbase.util.FSUtils; +import org.apache.hadoop.hdfs.DFSConfigKeys; +import org.apache.hadoop.http.HttpConfig; +import org.apache.hadoop.minikdc.MiniKdc; +import org.apache.hadoop.security.UserGroupInformation; +import org.apache.hadoop.security.authentication.util.KerberosName; +import org.apache.phoenix.query.ConfigurationFactory; +import org.apache.phoenix.query.QueryServices; +import org.apache.phoenix.queryserver.client.ThinClientUtil; +import org.apache.phoenix.queryserver.server.QueryServer; +import org.apache.phoenix.util.InstanceResolver; +import org.junit.AfterClass; +import org.junit.BeforeClass; +import org.junit.Test; +import org.junit.experimental.categories.Category; + +import java.io.*; +import java.lang.reflect.Field; +import java.security.PrivilegedAction; +import java.security.PrivilegedExceptionAction; +import java.sql.DriverManager;

```
+import java.sql.ResultSet; +import java.sql.Statement; +import java.util.ArrayList; +import java.util.List;
+import java.util.Map.Entry; +import java.util.concurrent.ExecutorService; +import
java.util.concurrent.Executors; +import java.util.concurrent.TimeUnit; + +import java.nio.file.Paths; +import
java.util.Map; + +import static org.junit.Assert.*; + +@Category(NeedsOwnMiniClusterTest.class) +public class
SecureQueryServerPhoenixDBIT { + private static final Log LOG =
LogFactory.getLog(SecureQueryServerPhoenixDBIT.class); + + private static final File TEMP_DIR = new
File(getTempDirForClass()); + private static final File KEYTAB_DIR = new File(TEMP_DIR, "keytabs"); +
private static final List<File> USER_KEYTAB_FILES = new ArrayList<>(); + + private static final String
SPNEGO_PRINCIPAL = "HTTP/localhost"; + private static final String PQS_PRINCIPAL =
"phoenixqs/localhost"; + private static final String SERVICE_PRINCIPAL = "securecluster/localhost"; + private
static File KEYTAB; + + private static MiniKdc KDC; + private static HBaseTestingUtility UTIL = new
HBaseTestingUtility(); + private static LocalHBaseCluster HBASE_CLUSTER; + private static int
NUM_CREATED_USERS; + + private static ExecutorService PQS_EXECUTOR; + private static QueryServer
PQS; + private static int PQS_PORT; + private static String PQS_URL; + + private static String
getTempDirForClass() { + StringBuilder sb = new StringBuilder(32); +
sb.append(System.getProperty("user.dir")).append(File.separator); + sb.append("target").append(File.separator); +
sb.append(SecureQueryServerPhoenixDBIT.class.getSimpleName()); + return sb.toString(); + } + + private static
void updateDefaultRealm() throws Exception { + // (at least) one other phoenix test triggers the caching of this
field before the KDC is up + // which causes principal parsing to fail. + Field f =
KerberosName.class.getDeclaredField("defaultRealm"); + f.setAccessible(true); + // Default realm for MiniKDC +
f.set(null, "EXAMPLE.COM"); + } + + private static void createUsers(int numUsers) throws Exception { +
assertNotNull("KDC is null, was setup method called?", KDC); + NUM_CREATED_USERS = numUsers; + for
(int i = 1; i <= numUsers; i++) { + String principal = "user" + i; + File keytabFile = new File(KEYTAB_DIR,
principal + ".keytab"); + KDC.createPrincipal(keytabFile, principal); + USER_KEYTAB_FILES.add(keytabFile);
+ } + } + + private static Entry<String,File> getUser(int offset) { + Preconditions.checkArgument(offset > 0 &&
offset <= NUM_CREATED_USERS); + return Maps.immutableEntry("user" + offset,
USER_KEYTAB_FILES.get(offset - 1)); + } + + /** + * Setup the security configuration for hdfs. + */ + private
static void setHdfsSecuredConfiguration(Configuration conf) throws Exception { + // Set principal+keytab
configuration for HDFS + conf.set(DFSConfigKeys.DFS_NAMENODE_KERBEROS_PRINCIPAL_KEY,
SERVICE_PRINCIPAL + "@" + KDC.getRealm()); +
conf.set(DFSConfigKeys.DFS_NAMENODE_KEYTAB_FILE_KEY, KEYTAB.getAbsolutePath()); +
conf.set(DFSConfigKeys.DFS_DATANODE_KERBEROS_PRINCIPAL_KEY, SERVICE_PRINCIPAL + "@" +
KDC.getRealm()); + conf.set(DFSConfigKeys.DFS_DATANODE_KEYTAB_FILE_KEY,
KEYTAB.getAbsolutePath()); +
conf.set(DFSConfigKeys.DFS_WEB_AUTHENTICATION_KERBEROS_PRINCIPAL_KEY,
SPNEGO_PRINCIPAL + "@" + KDC.getRealm()); + // Enable token access for HDFS blocks +
conf.setBoolean(DFSConfigKeys.DFS_BLOCK_ACCESS_TOKEN_ENABLE_KEY, true); + // Only use HTTPS
(required because we aren't using "secure" ports) + conf.set(DFSConfigKeys.DFS_HTTP_POLICY_KEY,
HttpConfig.Policy.HTTPS_ONLY.name()); + // Bind on localhost for spnego to have a chance at working +
conf.set(DFSConfigKeys.DFS_NAMENODE_HTTPS_ADDRESS_KEY, "localhost:0"); +
conf.set(DFSConfigKeys.DFS_DATANODE_HTTPS_ADDRESS_KEY, "localhost:0"); + + // Generate SSL certs
+ File keystoresDir = new File(UTIL.getDataTestDir("keystore").toUri().getPath()); + keystoresDir.mkdirs(); +
String sslConfDir = KeyStoreTestUtil.getClasspathDir(SecureQueryServerPhoenixDBIT.class); +
KeyStoreTestUtil.setupSSLConfig(keystoresDir.getAbsolutePath(), sslConfDir, conf, false); + + // Magic flag to
tell hdfs to not fail on using ports above 1024 + conf.setBoolean("ignore.secure.ports.for.testing", true); + } + +
private static void ensureIsEmptyDirectory(File f) throws IOException { + if (f.exists()) { + if (f.isDirectory()) { +
FileUtils.deleteDirectory(f); + } else { + assertTrue("Failed to delete keytab directory", f.delete()); + } + } +
assertTrue("Failed to create keytab directory", f.mkdirs()); + } + + /** + * Setup and start kerberos, hbase + */ +
@BeforeClass + public static void setUp() throws Exception { + final Configuration conf =
UTIL.getConfiguration(); + // Ensure the dirs we need are created/empty + ensureIsEmptyDirectory(TEMP_DIR);
+ ensureIsEmptyDirectory(KEYTAB_DIR); + KEYTAB = new File(KEYTAB_DIR, "test.keytab"); + // Start a
MiniKDC + KDC = UTIL.setupMiniKdc(KEYTAB); + // Create a service principal and spnego principal in one
keytab + // NB. Due to some apparent limitations between HDFS and HBase in the same JVM, trying to + // use
separate identies for HBase and HDFS results in a GSS initiate error. The quick + // solution is to just use a single
"service" principal instead of "hbase" and "hdfs" + // (or "dn" and "nn") per usual. +
KDC.createPrincipal(KEYTAB, SPNEGO_PRINCIPAL, PQS_PRINCIPAL, SERVICE_PRINCIPAL); + // Start
ZK by hand + UTIL.startMiniZKCluster(); + + // Create a number of unprivileged users + createUsers(3); + + //
Set configuration for HBase + HBaseKerberosUtils.setPrincipalForTesting(SERVICE_PRINCIPAL + "@" +
KDC.getRealm()); + HBaseKerberosUtils.setSecuredConfiguration(conf); + setHdfsSecuredConfiguration(conf);
+ UserGroupInformation.setConfiguration(conf); + conf.setInt(HConstants.MASTER_PORT, 0); +
conf.setInt(HConstants.MASTER_INFO_PORT, 0); + conf.setInt(HConstants.REGIONSERVER_PORT, 0); +
conf.setInt(HConstants.REGIONSERVER_INFO_PORT, 0); +
conf.setStrings(CoprocessorHost.REGION_COPROCESSOR_CONF_KEY, + TokenProvider.class.getName()); +
```

+ // Secure Phoenix setup + conf.set("phoenix.queryserver.kerberos.http.principal", SPNEGO_PRINCIPAL + "@" + KDC.getRealm()); + conf.set("phoenix.queryserver.http.keytab.file", KEYTAB.getAbsolutePath()); + conf.set("phoenix.queryserver.kerberos.principal", PQS_PRINCIPAL + "@" + KDC.getRealm()); + conf.set("phoenix.queryserver.keytab.file", KEYTAB.getAbsolutePath()); + conf.setBoolean(QueryServices.QUERY_SERVER_DISABLE_KERBEROS_LOGIN, true); + conf.setInt(QueryServices.QUERY_SERVER_HTTP_PORT_ATTRIB, 0); + // Required so that PQS can impersonate the end-users to HBase + conf.set("hadoop.proxyuser.phoenixqs.groups", "*"); + conf.set("hadoop.proxyuser.phoenixqs.hosts", "*"); + + // Clear the cached singletons so we can inject our own. + InstanceResolver.clearSingletons(); + // Make sure the ConnectionInfo doesn't try to pull a default Configuration + InstanceResolver.getSingleton(ConfigurationFactory.class, new ConfigurationFactory() { + @Override + public Configuration getConfiguration() { + return conf; + } + @Override + public Configuration getConfiguration(Configuration confToClone) { + Configuration copy = new Configuration(conf); + copy.addResource(confToClone); + return copy; + } + }); + updateDefaultRealm(); + + // Start HDFS + UTIL.startMiniDFSCluster(1); + // Use LocalHBaseCluster to avoid HBaseTestingUtility from doing something wrong + // NB. I'm not actually sure what HTU does incorrect, but this was pulled from some test + // classes in HBase itself. I couldn't get HTU to work myself (2017/07/06) + Path rootdir = UTIL.getDataTestDirOnTestFS(SecureQueryServerPhoenixDBIT.class.getSimpleName()); + FSUtils.setRootDir(conf, rootdir); + HBASE_CLUSTER = new LocalHBaseCluster(conf, 1); + HBASE_CLUSTER.startup(); + + // Then fork a thread with PQS in it. + startQueryServer(); + } + + private static void startQueryServer() throws Exception { + PQS = new QueryServer(new String[0], UTIL.getConfiguration()); + // Get the PQS ident for PQS to use + final UserGroupInformation ugi = UserGroupInformation.loginUserFromKeytabAndReturnUGI(PQS_PRINCIPAL, KEYTAB.getAbsolutePath()); + PQS_EXECUTOR = Executors.newSingleThreadExecutor(); + // Launch PQS, doing in the Kerberos login instead of letting PQS do it itself (which would + // break the HBase/HDFS logins also running in the same test case). + PQS_EXECUTOR.submit(new Runnable() { + @Override public void run() { + ugi.doAs(new PrivilegedAction<Void>() { + @Override public Void run() { + PQS.run(); + return null; + } + }); + } + }); + PQS.awaitRunning(); + PQS_PORT = PQS.getPort(); + PQS_URL = ThinClientUtil.getConnectionUrl("localhost", PQS_PORT) + ";authentication=SPNEGO"; + } + + @AfterClass + public static void stopKdc() throws Exception { + // Remove our custom ConfigurationFactory for future tests + InstanceResolver.clearSingletons(); + if (PQS_EXECUTOR != null) { + PQS.stop(); + PQS_EXECUTOR.shutdown(); + if (!PQS_EXECUTOR.awaitTermination(5, TimeUnit.SECONDS)) { + LOG.info("PQS didn't exit in 5 seconds, proceeding anyways."); + } + } + if (HBASE_CLUSTER != null) { + HBASE_CLUSTER.shutdown(); + HBASE_CLUSTER.join(); + } + if (UTIL != null) { + UTIL.shutdownMiniZKCluster(); + } + if (KDC != null) { + KDC.stop(); + } + } + + @Test + public void testBasicReadWrite() throws Exception { + final Entry<String,File> user1 = getUser(1); + String currentDirectory; + File file = new File("."); + currentDirectory = file.getAbsolutePath(); + LOG.debug("Current working directory : "+currentDirectory); + LOG.debug("PQS_PORT:" + PQS_PORT); + LOG.debug("PQS_URL: " + PQS_URL); + ArrayList<String> cmdList = new ArrayList<>(); + // This assumes the test is being run from phoenix/phoenix-queryserver + cmdList.add(Paths.get(currentDirectory, "src", "it", "bin", "test_phoenixdb.sh").toString()); + cmdList.add(Paths.get(currentDirectory, "..", "python").toString()); + cmdList.add(user1.getKey() + "@" + KDC.getRealm()); + cmdList.add(user1.getValue().getAbsolutePath()); + String osName = System.getProperty("os.name").toLowerCase(); + LOG.info("OS is " + osName); + File krb5ConfFile = null; + if (osName.indexOf("mac") >= 0) { + int kdcPort = KDC.getPort(); + LOG.info("MINIKDC PORT " + kdcPort); + // Render a Heimdal compatible krb5.conf + // Currently kinit will only try tcp if the KDC is defined as + // kdc = tcp/hostname:port + StringBuilder krb5conf = new StringBuilder(); + krb5conf.append("[libdefaults]\n"); + krb5conf.append(" default_realm = EXAMPLE.COM\n"); + krb5conf.append(" udp_preference_limit = 1\n"); + krb5conf.append("\n"); + krb5conf.append("[realms]\n"); + krb5conf.append(" EXAMPLE.COM = {\n"); + krb5conf.append(" kdc = tcp/localhost:"); + krb5conf.append(kdcPort); + krb5conf.append("\n"); + krb5conf.append(" }\n"); + + LOG.info("Writing Heimdal style krb5.conf"); + LOG.info(krb5conf.toString()); + krb5ConfFile = File.createTempFile("krb5.conf", null); + FileOutputStream fos = new FileOutputStream(krb5ConfFile); + fos.write(krb5conf.toString().getBytes()); + fos.close(); + LOG.info("krb5.conf written to " + krb5ConfFile.getAbsolutePath()); + cmdList.add(krb5ConfFile.getAbsolutePath()); + } else { + cmdList.add(System.getProperty("java.security.krb5.conf")); + LOG.info("Using miniKDC provided krb5.conf " + KDC.getKrb5conf().getAbsolutePath()); + } + + cmdList.add(Integer.toString(PQS_PORT)); + cmdList.add(Paths.get(currentDirectory, "src", "it", "bin", "test_phoenixdb.py").toString()); + + // kinit in some random credcache --- End diff -- Delete this stuff?

28. Github user joshelser commented on a diff in the pull request: https://github.com/apache/phoenix/pull/307#discussion_r201077800 --- Diff: python/phoenixdb-module/phoenixdb/__init__.py --- @@ -1,11 +1,10 @@ -# Licensed to the Apache Software Foundation (ASF) under one or more -# contributor license agreements. See the NOTICE file distributed with -# this work for additional information regarding copyright ownership. -# The ASF licenses this file to You under the Apache License, Version 2.0 -# (the "License"); you may not use this file except in compliance with -# the License. You

may obtain a copy of the License at +# Copyright 2015 Lukas Lalinsky --- End diff -- Any reason for the re-add of this? We don't need this after the IP Clearance process, I think. NOTICE file should be sufficient.

29. Github user joshelser commented on a diff in the pull request: https://github.com/apache/phoenix/pull/307#discussion_r201078376 --- Diff: python/requests-kerberos/LICENSE --- @@ -0,0 +1,15 @@ +ISC License --- End diff -- Just calling out that this is allowed: ISC is a Category-A license per https://www.apache.org/legal/resolved.html
30. Github user joshelser commented on a diff in the pull request: https://github.com/apache/phoenix/pull/307#discussion_r201075953 --- Diff: phoenix-queryserver/src/it/bin/test_phoenixdb.sh --- @@ -0,0 +1,59 @@ +#/usr/bin/env bash + +set -u +set -x +set -e + +function cleanup { + set +e + set +u + kdestroy + pushd ${PY_ENV_PATH}/bin + . deactivate "" + popd + rm -rf $PY_ENV_PATH +} + +trap cleanup EXIT + +echo "LAUNCHING SCRIPT" + +LOCAL_PY=$1 +PRINC=$2 +KEYTAB_LOC=$3 +KRB5_CFG_FILE=$4 +PQS_PORT=$5 +PYTHON_SCRIPT=$6 + +PY_ENV_PATH=$(mktemp -d) + +conda create -y -p $PY_ENV_PATH || virtualenv $PY_ENV_PATH + +pushd ${PY_ENV_PATH}/bin + +# conda activate does stuff with unbound variables :(+set +u +. activate "" + +popd + +set -u +echo "INSTALLING COMPONENTS" +pip install -e file:///${LOCAL_PY}/requests-kerberos +pip install -e file:///${LOCAL_PY}/phoenixdb-module + +export KRB5_CONFIG=$KRB5_CFG_FILE +cat $KRB5_CONFIG +export KRB5_TRACE=/dev/stdout + +#echo "RUNNING KINIT" +kinit -kt $KEYTAB_LOC $PRINC --- End diff -- Can we kinit to a custom location? e.g. the `-c` option. Then, later, we just set the variable `KRB5CCNAME` in the shell ENV. This would help prevent us from bashing the user's ticket (if they already have one).
31. Github user joshelser commented on a diff in the pull request: https://github.com/apache/phoenix/pull/307#discussion_r201077202 --- Diff: phoenix-queryserver/src/it/java/org/apache/phoenix/end2end/SecureQueryServerPhoenixDBIT.java --- @@ -0,0 +1,423 @@ +/* + * Licensed to the Apache Software Foundation (ASF) under one or more + * contributor license agreements. See the NOTICE file distributed with + * this work for additional information regarding copyright ownership. + * The ASF licenses this file to you under the Apache License, Version 2.0 + * (the "License"); you may not use this file except in compliance with + * the License. You may obtain a copy of the License at + * + * http://www.apache.org/licenses/LICENSE-2.0 + * + * Unless required by applicable law or agreed to in writing, software + * distributed under the License is distributed on an "AS IS" BASIS, + * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. + * See the License for the specific language governing permissions and + * limitations under the License. + */ +package org.apache.phoenix.end2end; + +import com.google.common.base.Preconditions; +import com.google.common.collect.Maps; +import org.apache.commons.io.FileUtils; +import org.apache.commons.logging.Log; +import org.apache.commons.logging.LogFactory; +import org.apache.hadoop.conf.Configuration; +import org.apache.hadoop.fs.Path; +import org.apache.hadoop.hbase.HBaseTestingUtility; +import org.apache.hadoop.hbase.HConstants; +import org.apache.hadoop.hbase.LocalHBaseCluster; +import org.apache.hadoop.hbase.coprocessor.CoprocessorHost; +import org.apache.hadoop.hbase.http.ssl.KeyStoreTestUtil; +import org.apache.hadoop.hbase.security.HBaseKerberosUtils; +import org.apache.hadoop.hbase.security.token.TokenProvider; +import org.apache.hadoop.hbase.util.FSUtils; +import org.apache.hadoop.hdfs.DFSConfigKeys; +import org.apache.hadoop.http.HttpConfig; +import org.apache.hadoop.minikdc.MiniKdc; +import org.apache.hadoop.security.UserGroupInformation; +import org.apache.hadoop.security.authentication.util.KerberosName; +import org.apache.phoenix.query.ConfigurationFactory; +import org.apache.phoenix.query.QueryServices; +import org.apache.phoenix.queryserver.client.ThinClientUtil; +import org.apache.phoenix.queryserver.server.QueryServer; +import org.apache.phoenix.util.InstanceResolver; +import org.junit.AfterClass; +import org.junit.BeforeClass; +import org.junit.Test; +import org.junit.experimental.categories.Category; + +import java.io.*; +import java.lang.reflect.Field; +import java.security.PrivilegedAction; +import java.security.PrivilegedExceptionAction; +import java.sql.DriverManager; +import java.sql.ResultSet; +import java.sql.Statement; +import java.util.ArrayList; +import java.util.List; +import java.util.Map.Entry; +import java.util.concurrent.ExecutorService; +import java.util.concurrent.Executors; +import java.util.concurrent.TimeUnit; + +import java.nio.file.Paths; +import java.util.Map; + +import static org.junit.Assert.*; + +@Category(NeedsOwnMiniClusterTest.class) +public class SecureQueryServerPhoenixDBIT { + private static final Log LOG = LogFactory.getLog(SecureQueryServerPhoenixDBIT.class); + + private static final File TEMP_DIR = new File(getTempDirForClass()); + private static final File KEYTAB_DIR = new File(TEMP_DIR, "keytabs"); + private static final List<File> USER_KEYTAB_FILES = new ArrayList<>(); + + private static final String SPNEGO_PRINCIPAL = "HTTP/localhost"; + private static final String PQS_PRINCIPAL = "phoenixqs/localhost"; + private static final String SERVICE_PRINCIPAL = "securecluster/localhost"; + private static File KEYTAB; + + private static MiniKdc KDC; + private static HBaseTestingUtility UTIL = new HBaseTestingUtility(); + private static LocalHBaseCluster HBASE_CLUSTER; + private static int NUM_CREATED_USERS; + + private static ExecutorService PQS_EXECUTOR; + private static QueryServer PQS; + private static int PQS_PORT; + private static String PQS_URL; + + private static String

```
getTempDirForClass() { + StringBuilder sb = new StringBuilder(32); +
sb.append(System.getProperty("user.dir")).append(File.separator); + sb.append("target").append(File.separator); +
sb.append(SecureQueryServerPhoenixDBIT.class.getSimpleName()); + return sb.toString(); + } + + private static
void updateDefaultRealm() throws Exception { + // (at least) one other phoenix test triggers the caching of this
field before the KDC is up + // which causes principal parsing to fail. + Field f =
KerberosName.class.getDeclaredField("defaultRealm"); + f.setAccessible(true); + // Default realm for MiniKDC +
f.set(null, "EXAMPLE.COM"); + } + + private static void createUsers(int numUsers) throws Exception { +
assertNotNull("KDC is null, was setup method called?", KDC); + NUM_CREATED_USERS = numUsers; + for
(int i = 1; i <= numUsers; i++) { + String principal = "user" + i; + File keytabFile = new File(KEYTAB_DIR,
principal + ".keytab"); + KDC.createPrincipal(keytabFile, principal); + USER_KEYTAB_FILES.add(keytabFile);
+ } + } + + private static Entry<String,File> getUser(int offset) { + Preconditions.checkArgument(offset > 0 &&
offset <= NUM_CREATED_USERS); + return Maps.immutableEntry("user" + offset,
USER_KEYTAB_FILES.get(offset - 1)); + } + + /** + * Setup the security configuration for hdfs. + */ + private
static void setHdfsSecuredConfiguration(Configuration conf) throws Exception { + // Set principal+keytab
configuration for HDFS + conf.set(DFSConfigKeys.DFS_NAMENODE_KERBEROS_PRINCIPAL_KEY,
SERVICE_PRINCIPAL + "@" + KDC.getRealm()); +
conf.set(DFSConfigKeys.DFS_NAMENODE_KEYTAB_FILE_KEY, KEYTAB.getAbsolutePath()); +
conf.set(DFSConfigKeys.DFS_DATANODE_KERBEROS_PRINCIPAL_KEY, SERVICE_PRINCIPAL + "@" +
KDC.getRealm()); + conf.set(DFSConfigKeys.DFS_DATANODE_KEYTAB_FILE_KEY,
KEYTAB.getAbsolutePath()); +
conf.set(DFSConfigKeys.DFS_WEB_AUTHENTICATION_KERBEROS_PRINCIPAL_KEY,
SPNEGO_PRINCIPAL + "@" + KDC.getRealm()); + // Enable token access for HDFS blocks +
conf.setBoolean(DFSConfigKeys.DFS_BLOCK_ACCESS_TOKEN_ENABLE_KEY, true); + // Only use HTTPS
(required because we aren't using "secure" ports) + conf.set(DFSConfigKeys.DFS_HTTP_POLICY_KEY,
HttpConfig.Policy.HTTPS_ONLY.name()); + // Bind on localhost for spnego to have a chance at working +
conf.set(DFSConfigKeys.DFS_NAMENODE_HTTPS_ADDRESS_KEY, "localhost:0"); +
conf.set(DFSConfigKeys.DFS_DATANODE_HTTPS_ADDRESS_KEY, "localhost:0"); + + // Generate SSL certs
+ File keystoresDir = new File(UTIL.getDataTestDir("keystore").toUri().getPath()); + keystoresDir.mkdirs(); +
String sslConfDir = KeyStoreTestUtil.getClasspathDir(SecureQueryServerPhoenixDBIT.class); +
KeyStoreTestUtil.setupSSLConfig(keystoresDir.getAbsolutePath(), sslConfDir, conf, false); + + // Magic flag to
tell hdfs to not fail on using ports above 1024 + conf.setBoolean("ignore.secure.ports.for.testing", true); + } + +
private static void ensureIsEmptyDirectory(File f) throws IOException { + if (f.exists()) { + if (f.isDirectory()) { +
FileUtils.deleteDirectory(f); + } else { + assertTrue("Failed to delete keytab directory", f.delete()); + } + } +
assertTrue("Failed to create keytab directory", f.mkdirs()); + } + + /** + * Setup and start kerberos, hbase + */ +
@BeforeClass + public static void setUp() throws Exception { + final Configuration conf =
UTIL.getConfiguration(); + // Ensure the dirs we need are created/empty + ensureIsEmptyDirectory(TEMP_DIR);
+ ensureIsEmptyDirectory(KEYTAB_DIR); + KEYTAB = new File(KEYTAB_DIR, "test.keytab"); + // Start a
MiniKDC + KDC = UTIL.setupMiniKdc(KEYTAB); + // Create a service principal and spnego principal in one
keytab + // NB. Due to some apparent limitations between HDFS and HBase in the same JVM, trying to + // use
separate identies for HBase and HDFS results in a GSS initiate error. The quick + // solution is to just use a single
"service" principal instead of "hbase" and "hdfs" + // (or "dn" and "nn") per usual. +
KDC.createPrincipal(KEYTAB, SPNEGO_PRINCIPAL, PQS_PRINCIPAL, SERVICE_PRINCIPAL); + // Start
ZK by hand + UTIL.startMiniZKCluster(); + + // Create a number of unprivileged users + createUsers(3); + + //
Set configuration for HBase + HBaseKerberosUtils.setPrincipalForTesting(SERVICE_PRINCIPAL + "@" +
KDC.getRealm()); + HBaseKerberosUtils.setSecuredConfiguration(conf); + setHdfsSecuredConfiguration(conf);
+ UserGroupInformation.setConfiguration(conf); + conf.setInt(HConstants.MASTER_PORT, 0); +
conf.setInt(HConstants.MASTER_INFO_PORT, 0); + conf.setInt(HConstants.REGIONSERVER_PORT, 0); +
conf.setInt(HConstants.REGIONSERVER_INFO_PORT, 0); +
conf.setStrings(CoprocessorHost.REGION_COPROCESSOR_CONF_KEY, + TokenProvider.class.getName()); +
+ // Secure Phoenix setup + conf.set("phoenix.queryserver.kerberos.http.principal", SPNEGO_PRINCIPAL + "@"
+ KDC.getRealm()); + conf.set("phoenix.queryserver.http.keytab.file", KEYTAB.getAbsolutePath()); +
conf.set("phoenix.queryserver.kerberos.principal", PQS_PRINCIPAL + "@" + KDC.getRealm()); +
conf.set("phoenix.queryserver.keytab.file", KEYTAB.getAbsolutePath()); +
conf.setBoolean(QueryServices.QUERY_SERVER_DISABLE_KERBEROS_LOGIN, true); +
conf.setInt(QueryServices.QUERY_SERVER_HTTP_PORT_ATTRIB, 0); + // Required so that PQS can
impersonate the end-users to HBase + conf.set("hadoop.proxyuser.phoenixqs.groups", "*"); +
conf.set("hadoop.proxyuser.phoenixqs.hosts", "*"); + + // Clear the cached singletons so we can inject our own. +
InstanceResolver.clearSingletons(); + // Make sure the ConnectionInfo doesn't try to pull a default Configuration +
InstanceResolver.getSingleton(ConfigurationFactory.class, new ConfigurationFactory() { + @Override + public
Configuration getConfiguration() { + return conf; + } + @Override + public Configuration
getConfiguration(Configuration confToClone) { + Configuration copy = new Configuration(conf); +
copy.addResource(confToClone); + return copy; + } + }); + updateDefaultRealm(); + + // Start HDFS +
UTIL.startMiniDFSCluster(1); + // Use LocalHBaseCluster to avoid HBaseTestingUtility from doing something
```

wrong + // NB. I'm not actually sure what HTU does incorrect, but this was pulled from some test + // classes in HBase itself. I couldn't get HTU to work myself (2017/07/06) + Path rootdir = UTIL.getDataTestDirOnTestFS(SecureQueryServerPhoenixDBIT.class.getSimpleName()); + FSUtils.setRootDir(conf, rootdir); + HBASE_CLUSTER = new LocalHBaseCluster(conf, 1); + HBASE_CLUSTER.startup(); + + // Then fork a thread with PQS in it. + startQueryServer(); + } + + private static void startQueryServer() throws Exception { + PQS = new QueryServer(new String[0], UTIL.getConfiguration()); + // Get the PQS ident for PQS to use + final UserGroupInformation ugi = UserGroupInformation.loginUserFromKeytabAndReturnUGI(PQS_PRINCIPAL, KEYTAB.getAbsolutePath()); + PQS_EXECUTOR = Executors.newSingleThreadExecutor(); + // Launch PQS, doing in the Kerberos login instead of letting PQS do it itself (which would + // break the HBase/HDFS logins also running in the same test case). + PQS_EXECUTOR.submit(new Runnable() { + @Override public void run() { + ugi.doAs(new PrivilegedAction<Void>() { + @Override public Void run() { + PQS.run(); + return null; + } + }); + } + }); + PQS.awaitRunning(); + PQS_PORT = PQS.getPort(); + PQS_URL = ThinClientUtil.getConnectionUrl("localhost", PQS_PORT) + ";authentication=SPNEGO"; + } + + @AfterClass + public static void stopKdc() throws Exception { + // Remove our custom ConfigurationFactory for future tests + InstanceResolver.clearSingletons(); + if (PQS_EXECUTOR != null) { + PQS.stop(); + PQS_EXECUTOR.shutdown(); + if (!PQS_EXECUTOR.awaitTermination(5, TimeUnit.SECONDS)) { + LOG.info("PQS didn't exit in 5 seconds, proceeding anyways."); + } + } + if (HBASE_CLUSTER != null) { + HBASE_CLUSTER.shutdown(); + HBASE_CLUSTER.join(); + } + if (UTIL != null) { + UTIL.shutdownMiniZKCluster(); + } + if (KDC != null) { + KDC.stop(); + } + } + + @Test + public void testBasicReadWrite() throws Exception { + final Entry<String,File> user1 = getUser(1); + String currentDirectory; + File file = new File("."); + currentDirectory = file.getAbsolutePath(); + LOG.debug("Current working directory : "+currentDirectory); + LOG.debug("PQS_PORT:" + PQS_PORT); + LOG.debug("PQS_URL: " + PQS_URL); + ArrayList<String> cmdList = new ArrayList<>(); + // This assumes the test is being run from phoenix/phoenix-queryserver + cmdList.add(Paths.get(currentDirectory, "src", "it", "bin", "test_phoenixdb.sh").toString()); + cmdList.add(Paths.get(currentDirectory, "..", "python").toString()); + cmdList.add(user1.getKey() + "@" + KDC.getRealm()); + cmdList.add(user1.getValue().getAbsolutePath()); + String osName = System.getProperty("os.name").toLowerCase(); + LOG.info("OS is " + osName); + File krb5ConfFile = null; + if (osName.indexOf("mac") >= 0) { + int kdcPort = KDC.getPort(); + LOG.info("MINIKDC PORT " + kdcPort); + // Render a Heimdal compatible krb5.conf + // Currently kinit will only try tcp if the KDC is defined as + // kdc = tcp/hostname:port --- End diff -- Nice! I learned something here :)

32. Github user joshelser commented on a diff in the pull request: https://github.com/apache/phoenix/pull/307#discussion_r201077355 --- Diff: phoenix-queryserver/src/it/java/org/apache/phoenix/end2end/SecureQueryServerPhoenixDBIT.java --- @@ -0,0 +1,423 @@ +/* + * Licensed to the Apache Software Foundation (ASF) under one or more + * contributor license agreements. See the NOTICE file distributed with + * this work for additional information regarding copyright ownership. + * The ASF licenses this file to you under the Apache License, Version 2.0 + * (the "License"); you may not use this file except in compliance with + * the License. You may obtain a copy of the License at + * + * http://www.apache.org/licenses/LICENSE-2.0 + * + * Unless required by applicable law or agreed to in writing, software + * distributed under the License is distributed on an "AS IS" BASIS, + * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. + * See the License for the specific language governing permissions and + * limitations under the License. + */ +package org.apache.phoenix.end2end; + +import com.google.common.base.Preconditions; +import com.google.common.collect.Maps; +import org.apache.commons.io.FileUtils; +import org.apache.commons.logging.Log; +import org.apache.commons.logging.LogFactory; +import org.apache.hadoop.conf.Configuration; +import org.apache.hadoop.fs.Path; +import org.apache.hadoop.hbase.HBaseTestingUtility; +import org.apache.hadoop.hbase.HConstants; +import org.apache.hadoop.hbase.LocalHBaseCluster; +import org.apache.hadoop.hbase.coprocessor.CoprocessorHost; +import org.apache.hadoop.hbase.http.ssl.KeyStoreTestUtil; +import org.apache.hadoop.hbase.security.HBaseKerberosUtils; +import org.apache.hadoop.hbase.security.token.TokenProvider; +import org.apache.hadoop.hbase.util.FSUtils; +import org.apache.hadoop.hdfs.DFSConfigKeys; +import org.apache.hadoop.http.HttpConfig; +import org.apache.hadoop.minikdc.MiniKdc; +import org.apache.hadoop.security.UserGroupInformation; +import org.apache.hadoop.security.authentication.util.KerberosName; +import org.apache.phoenix.query.ConfigurationFactory; +import org.apache.phoenix.query.QueryServices; +import org.apache.phoenix.queryserver.client.ThinClientUtil; +import org.apache.phoenix.queryserver.server.QueryServer; +import org.apache.phoenix.util.InstanceResolver; +import org.junit.AfterClass; +import org.junit.BeforeClass; +import org.junit.Test; +import org.junit.experimental.categories.Category; + +import java.io.*; +import java.lang.reflect.Field; +import java.security.PrivilegedAction; +import java.security.PrivilegedExceptionAction; +import java.sql.DriverManager; +import java.sql.ResultSet; +import java.sql.Statement; +import java.util.ArrayList; +import java.util.List; +import java.util.Map.Entry; +import java.util.concurrent.ExecutorService; +import java.util.concurrent.Executors; +import java.util.concurrent.TimeUnit; + +import java.nio.file.Paths; +import

java.util.Map; + +import static org.junit.Assert.*; + +@Category(NeedsOwnMiniClusterTest.class) +public class SecureQueryServerPhoenixDBIT { + private static final Log LOG = LogFactory.getLog(SecureQueryServerPhoenixDBIT.class); + + private static final File TEMP_DIR = new File(getTempDirForClass()); + private static final File KEYTAB_DIR = new File(TEMP_DIR, "keytabs"); + private static final List<File> USER_KEYTAB_FILES = new ArrayList<>(); + + private static final String SPNEGO_PRINCIPAL = "HTTP/localhost"; + private static final String PQS_PRINCIPAL = "phoenixqs/localhost"; + private static final String SERVICE_PRINCIPAL = "securecluster/localhost"; + private static File KEYTAB; + + private static MiniKdc KDC; + private static HBaseTestingUtility UTIL = new HBaseTestingUtility(); + private static LocalHBaseCluster HBASE_CLUSTER; + private static int NUM_CREATED_USERS; + + private static ExecutorService PQS_EXECUTOR; + private static QueryServer PQS; + private static int PQS_PORT; + private static String PQS_URL; + + private static String getTempDirForClass() { + StringBuilder sb = new StringBuilder(32); + sb.append(System.getProperty("user.dir")).append(File.separator); + sb.append("target").append(File.separator); + sb.append(SecureQueryServerPhoenixDBIT.class.getSimpleName()); + return sb.toString(); + } + + private static void updateDefaultRealm() throws Exception { + // (at least) one other phoenix test triggers the caching of this field before the KDC is up + // which causes principal parsing to fail. + Field f = KerberosName.class.getDeclaredField("defaultRealm"); + f.setAccessible(true); + // Default realm for MiniKDC + f.set(null, "EXAMPLE.COM"); + } + + private static void createUsers(int numUsers) throws Exception { + assertNotNull("KDC is null, was setup method called?", KDC); + NUM_CREATED_USERS = numUsers; + for (int i = 1; i <= numUsers; i++) { + String principal = "user" + i; + File keytabFile = new File(KEYTAB_DIR, principal + ".keytab"); + KDC.createPrincipal(keytabFile, principal); + USER_KEYTAB_FILES.add(keytabFile); + } + } + + private static Entry<String,File> getUser(int offset) { + Preconditions.checkArgument(offset > 0 && offset <= NUM_CREATED_USERS); + return Maps.immutableEntry("user" + offset, USER_KEYTAB_FILES.get(offset - 1)); + } + + /** + * Setup the security configuration for hdfs. + */ + private static void setHdfsSecuredConfiguration(Configuration conf) throws Exception { + // Set principal+keytab configuration for HDFS + conf.set(DFSConfigKeys.DFS_NAMENODE_KERBEROS_PRINCIPAL_KEY, SERVICE_PRINCIPAL + "@" + KDC.getRealm()); + conf.set(DFSConfigKeys.DFS_NAMENODE_KEYTAB_FILE_KEY, KEYTAB.getAbsolutePath()); + conf.set(DFSConfigKeys.DFS_DATANODE_KERBEROS_PRINCIPAL_KEY, SERVICE_PRINCIPAL + "@" + KDC.getRealm()); + conf.set(DFSConfigKeys.DFS_DATANODE_KEYTAB_FILE_KEY, KEYTAB.getAbsolutePath()); + conf.set(DFSConfigKeys.DFS_WEB_AUTHENTICATION_KERBEROS_PRINCIPAL_KEY, SPNEGO_PRINCIPAL + "@" + KDC.getRealm()); + // Enable token access for HDFS blocks + conf.setBoolean(DFSConfigKeys.DFS_BLOCK_ACCESS_TOKEN_ENABLE_KEY, true); + // Only use HTTPS (required because we aren't using "secure" ports) + conf.set(DFSConfigKeys.DFS_HTTP_POLICY_KEY, HttpConfig.Policy.HTTPS_ONLY.name()); + // Bind on localhost for spnego to have a chance at working + conf.set(DFSConfigKeys.DFS_NAMENODE_HTTPS_ADDRESS_KEY, "localhost:0"); + conf.set(DFSConfigKeys.DFS_DATANODE_HTTPS_ADDRESS_KEY, "localhost:0"); + + // Generate SSL certs + File keystoresDir = new File(UTIL.getDataTestDir("keystore").toUri().getPath()); + keystoresDir.mkdirs(); + String sslConfDir = KeyStoreTestUtil.getClasspathDir(SecureQueryServerPhoenixDBIT.class); + KeyStoreTestUtil.setupSSLConfig(keystoresDir.getAbsolutePath(), sslConfDir, conf, false); + + // Magic flag to tell hdfs to not fail on using ports above 1024 + conf.setBoolean("ignore.secure.ports.for.testing", true); + } + + private static void ensureIsEmptyDirectory(File f) throws IOException { + if (f.exists()) { + if (f.isDirectory()) { + FileUtils.deleteDirectory(f); + } else { + assertTrue("Failed to delete keytab directory", f.delete()); + } + } + assertTrue("Failed to create keytab directory", f.mkdirs()); + } + + /** + * Setup and start kerberos, hbase + */ + @BeforeClass + public static void setUp() throws Exception { + final Configuration conf = UTIL.getConfiguration(); + // Ensure the dirs we need are created/empty + ensureIsEmptyDirectory(TEMP_DIR); + ensureIsEmptyDirectory(KEYTAB_DIR); + KEYTAB = new File(KEYTAB_DIR, "test.keytab"); + // Start a MiniKDC + KDC = UTIL.setupMiniKdc(KEYTAB); + // Create a service principal and spnego principal in one keytab + // NB. Due to some apparent limitations between HDFS and HBase in the same JVM, trying to + // use separate identies for HBase and HDFS results in a GSS initiate error. The quick + // solution is to just use a single "service" principal instead of "hbase" and "hdfs" + // (or "dn" and "nn") per usual. + KDC.createPrincipal(KEYTAB, SPNEGO_PRINCIPAL, PQS_PRINCIPAL, SERVICE_PRINCIPAL); + // Start ZK by hand + UTIL.startMiniZKCluster(); + + // Create a number of unprivileged users + createUsers(3); + + // Set configuration for HBase + HBaseKerberosUtils.setPrincipalForTesting(SERVICE_PRINCIPAL + "@" + KDC.getRealm()); + HBaseKerberosUtils.setSecuredConfiguration(conf); + setHdfsSecuredConfiguration(conf); + UserGroupInformation.setConfiguration(conf); + conf.setInt(HConstants.MASTER_PORT, 0); + conf.setInt(HConstants.MASTER_INFO_PORT, 0); + conf.setInt(HConstants.REGIONSERVER_PORT, 0); + conf.setInt(HConstants.REGIONSERVER_INFO_PORT, 0); + conf.setStrings(CoprocessorHost.REGION_COPROCESSOR_CONF_KEY, + TokenProvider.class.getName()); + + // Secure Phoenix setup + conf.set("phoenix.queryserver.kerberos.http.principal", SPNEGO_PRINCIPAL + "@" + KDC.getRealm()); + conf.set("phoenix.queryserver.http.keytab.file", KEYTAB.getAbsolutePath()); + conf.set("phoenix.queryserver.kerberos.principal", PQS_PRINCIPAL + "@" + KDC.getRealm()); +

```
conf.set("phoenix.queryserver.keytab.file", KEYTAB.getAbsolutePath()); +
conf.setBoolean(QueryServices.QUERY_SERVER_DISABLE_KERBEROS_LOGIN, true); +
conf.setInt(QueryServices.QUERY_SERVER_HTTP_PORT_ATTRIB, 0); + // Required so that PQS can
impersonate the end-users to HBase + conf.set("hadoop.proxyuser.phoenixqs.groups", "*"); +
conf.set("hadoop.proxyuser.phoenixqs.hosts", "*"); + + // Clear the cached singletons so we can inject our own. +
InstanceResolver.clearSingletons(); + // Make sure the ConnectionInfo doesn't try to pull a default Configuration +
InstanceResolver.getSingleton(ConfigurationFactory.class, new ConfigurationFactory() { + @Override + public
Configuration getConfiguration() { + return conf; + } + @Override + public Configuration
getConfiguration(Configuration confToClone) { + Configuration copy = new Configuration(conf); +
copy.addResource(confToClone); + return copy; + } + }); + updateDefaultRealm(); + + // Start HDFS +
UTIL.startMiniDFSCluster(1); + // Use LocalHBaseCluster to avoid HBaseTestingUtility from doing something
wrong + // NB. I'm not actually sure what HTU does incorrect, but this was pulled from some test + // classes in
HBase itself. I couldn't get HTU to work myself (2017/07/06) + Path rootdir =
UTIL.getDataTestDirOnTestFS(SecureQueryServerPhoenixDBIT.class.getSimpleName()); +
FSUtils.setRootDir(conf, rootdir); + HBASE_CLUSTER = new LocalHBaseCluster(conf, 1); +
HBASE_CLUSTER.startup(); + + // Then fork a thread with PQS in it. + startQueryServer(); + } + + private static
void startQueryServer() throws Exception { + PQS = new QueryServer(new String[0], UTIL.getConfiguration());
+ // Get the PQS ident for PQS to use + final UserGroupInformation ugi =
UserGroupInformation.loginUserFromKeytabAndReturnUGI(PQS_PRINCIPAL, KEYTAB.getAbsolutePath()); +
PQS_EXECUTOR = Executors.newSingleThreadExecutor(); + // Launch PQS, doing in the Kerberos login instead
of letting PQS do it itself (which would + // break the HBase/HDFS logins also running in the same test case). +
PQS_EXECUTOR.submit(new Runnable() { + @Override public void run() { + ugi.doAs(new
PrivilegedAction<Void>() { + @Override public Void run() { + PQS.run(); + return null; + } + }); + } + }); +
PQS.awaitRunning(); + PQS_PORT = PQS.getPort(); + PQS_URL =
ThinClientUtil.getConnectionUrl("localhost", PQS_PORT) + ";authentication=SPNEGO"; + } + + @AfterClass +
public static void stopKdc() throws Exception { + // Remove our custom ConfigurationFactory for future tests +
InstanceResolver.clearSingletons(); + if (PQS_EXECUTOR != null) { + PQS.stop(); +
PQS_EXECUTOR.shutdown(); + if (!PQS_EXECUTOR.awaitTermination(5, TimeUnit.SECONDS)) { +
LOG.info("PQS didn't exit in 5 seconds, proceeding anyways."); + } + } + if (HBASE_CLUSTER != null) { +
HBASE_CLUSTER.shutdown(); + HBASE_CLUSTER.join(); + } + if (UTIL != null) { +
UTIL.shutdownMiniZKCluster(); + } + if (KDC != null) { + KDC.stop(); + } + } + + @Test + public void
testBasicReadWrite() throws Exception { + final Entry<String,File> user1 = getUser(1); + String currentDirectory;
+ File file = new File("."); + currentDirectory = file.getAbsolutePath(); + LOG.debug("Current working directory :
"+currentDirectory); + LOG.debug("PQS_PORT:" + PQS_PORT); + LOG.debug("PQS_URL: " + PQS_URL); +
ArrayList<String> cmdList = new ArrayList<>(); + // This assumes the test is being run from phoenix/phoenix-
queryserver + cmdList.add(Paths.get(currentDirectory, "src", "it", "bin", "test_phoenixdb.sh").toString()); +
cmdList.add(Paths.get(currentDirectory, "..", "python").toString()); + cmdList.add(user1.getKey() + "@" +
KDC.getRealm()); + cmdList.add(user1.getValue().getAbsolutePath()); + String osName =
System.getProperty("os.name").toLowerCase(); + LOG.info("OS is " + osName); + File krb5ConfFile = null; + if
(osName.indexOf("mac") >= 0 ) { + int kdcPort = KDC.getPort(); + LOG.info("MINIKDC PORT " + kdcPort); +
// Render a Heimdal compatible krb5.conf + // Currently kinit will only try tcp if the KDC is defined as + // kdc =
tcp/hostname:port + StringBuilder krb5conf = new StringBuilder(); + krb5conf.append("[libdefaults]\n"); +
krb5conf.append(" default_realm = EXAMPLE.COM\n"); + krb5conf.append(" udp_preference_limit = 1\n"); +
krb5conf.append("\n"); + krb5conf.append("[realms]\n"); + krb5conf.append(" EXAMPLE.COM = {\n"); +
krb5conf.append(" kdc = tcp/localhost:"); + krb5conf.append(kdcPort); + krb5conf.append("\n"); +
krb5conf.append(" }\n"); + + LOG.info("Writing Heimdal style krb5.conf"); + LOG.info(krb5conf.toString()); +
krb5ConfFile = File.createTempFile("krb5.conf", null); + FileOutputStream fos = new
FileOutputStream(krb5ConfFile); + fos.write(krb5conf.toString().getBytes()); + fos.close(); +
LOG.info("krb5.conf written to " + krb5ConfFile.getAbsolutePath()); +
cmdList.add(krb5ConfFile.getAbsolutePath()); + } else { +
cmdList.add(System.getProperty("java.security.krb5.conf")); + LOG.info("Using miniKDC provided krb5.conf " +
KDC.getKrb5conf().getAbsolutePath()); + } + + cmdList.add(Integer.toString(PQS_PORT)); +
cmdList.add(Paths.get(currentDirectory, "src", "it", "bin", "test_phoenixdb.py").toString()); + + // kinit in some
random credcache + /*String KRB5CCNAME; + if (osName.indexOf("mac") >= 0 ) + KRB5CCNAME =
kinit(user1.getKey(), user1.getValue(), krb5ConfFile); + else + KRB5CCNAME = kinit(user1.getKey(),
user1.getValue(), KDC.getKrb5conf());*/ + + + ProcessBuilder runPython = new ProcessBuilder(cmdList); +
Map<String, String> runPythonEnv = runPython.environment(); + //runPythonEnv.put("KRB5CCNAME",
KRB5CCNAME); + Process runPythonProcess = runPython.start(); + BufferedReader processOutput = new
BufferedReader(new InputStreamReader(runPythonProcess.getInputStream())); + BufferedReader processError =
new BufferedReader(new InputStreamReader(runPythonProcess.getErrorStream())); + int exitCode =
runPythonProcess.waitFor(); + + // dump stdout and stderr + while (processOutput.ready()) { +
LOG.info(processOutput.readLine()); + } + while (processError.ready()) { + LOG.error(processError.readLine());
+ } + + // Not managed by miniKDC so we have to clean up + if (krb5ConfFile != null) + krb5ConfFile.delete(); +
```

+ assertEquals("Subprocess exited with errors", 0, exitCode); + } + + byte[] copyBytes(byte[] src, int offset, int length) { + byte[] dest = new byte[length]; + System.arraycopy(src, offset, dest, 0, length); + return dest; + } + + String kinit(String principal, File keytab, File krb5ConfFile) throws IOException{ --- End diff -- This is unused now, right?

33. Github user joshelser commented on a diff in the pull request: https://github.com/apache/phoenix/pull/307#discussion_r201418146 --- Diff: phoenix-queryserver/pom.xml --- @@ -47,6 +47,11 @@ <plugin> <groupId>org.apache.maven.plugins</groupId> <artifactId>maven-failsafe-plugin</artifactId> + <configuration> + <excludes> + <exclude>**/SecureQueryServerPhoenixDBIT.java</exclude> --- End diff -- You not intending for this test to be executed during the normal build process?
34. Github user joshelser commented on the issue: https://github.com/apache/phoenix/pull/307 > How do we make sure requests-kerberos is built in a way that phoenixdb will grab it? (e.g. 0.13.0.dev0-phoenixdb) Ah, I see this is updated in python/requests-kerberos/requests_kerberos/__init__.py. We should update python/phoenixdb/requirements.txt to make sure we pull that 0.13.0.dev0-phoenixdb, right?
35. Github user joshelser commented on the issue: https://github.com/apache/phoenix/pull/307 Tried to run the test, but it failed with: ``` 2018-07-10 13:43:55,599 ERROR [main] end2end.SecureQueryServerPhoenixDBIT(359): + . deactivate '' 2018-07-10 13:43:55,599 ERROR [main] end2end.SecureQueryServerPhoenixDBIT(359): /Users/jelser/projects/phoenix.git/phoenix-queryserver/./src/it/bin/test_phoenixdb.sh: line 12: deactivate: No such file or directory ``` Might have been me hacking on things... digging more.
36. Github user pu239ppy commented on a diff in the pull request: https://github.com/apache/phoenix/pull/307#discussion_r201481976 --- Diff: phoenix-queryserver/src/it/bin/test_phoenixdb.sh --- @@ -0,0 +1,59 @@ +#/usr/bin/env bash + +set -u +set -x +set -e + +function cleanup { + set +e + set +u + kdestroy + pushd ${PY_ENV_PATH}/bin + . deactivate "" + popd + rm -rf $PY_ENV_PATH +} + +trap cleanup EXIT + +echo "LAUNCHING SCRIPT" + +LOCAL_PY=$1 +PRINC=$2 +KEYTAB_LOC=$3 +KRB5_CFG_FILE=$4 +PQS_PORT=$5 +PYTHON_SCRIPT=$6 + +PY_ENV_PATH=$(mktemp -d) + +conda create -y -p $PY_ENV_PATH || virtualenv $PY_ENV_PATH + +pushd ${PY_ENV_PATH}/bin + +# conda activate does stuff with unbound variables :(+set +u +. activate "" + +popd + +set -u +echo "INSTALLING COMPONENTS" +pip install -e file:///${LOCAL_PY}/requests-kerberos +pip install -e file:///${LOCAL_PY}/phoenixdb-module + +export KRB5_CONFIG=$KRB5_CFG_FILE +cat $KRB5_CONFIG +export KRB5_TRACE=/dev/stdout + +#echo "RUNNING KINIT" +kinit -kt $KEYTAB_LOC $PRINC --- End diff -- I tried something similar + File KRB5CCNAME = File.createTempFile("krb5ccname", null); + kinitEnv.put("KRB5CCNAME", KRB5CCNAME.getAbsolutePath()); This stalled, although looking at the code now it probably should have been a directory, which is why kinit stalled I can try this again
37. Github user pu239ppy commented on a diff in the pull request: https://github.com/apache/phoenix/pull/307#discussion_r201482215 --- Diff: phoenix-queryserver/src/it/bin/test_phoenixdb.sh --- @@ -0,0 +1,59 @@ +#/usr/bin/env bash + +set -u +set -x +set -e + +function cleanup { + set +e + set +u + kdestroy --- End diff -- Not to be overly pedantic, but you would want to still pass krb5ccname and just call kdestroy to make sure proper cleanup is done
38. Github user pu239ppy commented on a diff in the pull request: https://github.com/apache/phoenix/pull/307#discussion_r201482297 --- Diff: phoenix-queryserver/src/it/java/org/apache/phoenix/end2end/SecureQueryServerPhoenixDBIT.java --- @@ -0,0 +1,423 @@ +/* + * Licensed to the Apache Software Foundation (ASF) under one or more + * contributor license agreements. See the NOTICE file distributed with + * this work for additional information regarding copyright ownership. + * The ASF licenses this file to you under the Apache License, Version 2.0 + * (the "License"); you may not use this file except in compliance with + * the License. You may obtain a copy of the License at + * + * http://www.apache.org/licenses/LICENSE-2.0 + * + * Unless required by applicable law or agreed to in writing, software + * distributed under the License is distributed on an "AS IS" BASIS, + * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. + * See the License for the specific language governing permissions and + * limitations under the License. + */ +package org.apache.phoenix.end2end; + +import com.google.common.base.Preconditions; +import com.google.common.collect.Maps; +import org.apache.commons.io.FileUtils; +import org.apache.commons.logging.Log; +import org.apache.commons.logging.LogFactory; +import org.apache.hadoop.conf.Configuration; +import org.apache.hadoop.fs.Path; +import org.apache.hadoop.hbase.HBaseTestingUtility; +import org.apache.hadoop.hbase.HConstants; +import org.apache.hadoop.hbase.LocalHBaseCluster; +import org.apache.hadoop.hbase.coprocessor.CoprocessorHost; +import org.apache.hadoop.hbase.http.ssl.KeyStoreTestUtil; +import org.apache.hadoop.hbase.security.HBaseKerberosUtils; +import org.apache.hadoop.hbase.security.token.TokenProvider; +import org.apache.hadoop.hbase.util.FSUtils; +import org.apache.hadoop.hdfs.DFSConfigKeys; +import org.apache.hadoop.http.HttpConfig; +import org.apache.hadoop.minikdc.MiniKdc; +import org.apache.hadoop.security.UserGroupInformation; +import org.apache.hadoop.security.authentication.util.KerberosName; +import org.apache.phoenix.query.ConfigurationFactory; +import org.apache.phoenix.query.QueryServices; +import

```
org.apache.phoenix.queryserver.client.ThinClientUtil; +import
org.apache.phoenix.queryserver.server.QueryServer; +import org.apache.phoenix.util.InstanceResolver; +import
org.junit.AfterClass; +import org.junit.BeforeClass; +import org.junit.Test; +import
org.junit.experimental.categories.Category; + +import java.io.*; +import java.lang.reflect.Field; +import
java.security.PrivilegedAction; +import java.security.PrivilegedExceptionAction; +import java.sql.DriverManager;
+import java.sql.ResultSet; +import java.sql.Statement; +import java.util.ArrayList; +import java.util.List;
+import java.util.Map.Entry; +import java.util.concurrent.ExecutorService; +import
java.util.concurrent.Executors; +import java.util.concurrent.TimeUnit; + +import java.nio.file.Paths; +import
java.util.Map; + +import static org.junit.Assert.*; + +@Category(NeedsOwnMiniClusterTest.class) +public class
SecureQueryServerPhoenixDBIT { + private static final Log LOG =
LogFactory.getLog(SecureQueryServerPhoenixDBIT.class); + + private static final File TEMP_DIR = new
File(getTempDirForClass()); + private static final File KEYTAB_DIR = new File(TEMP_DIR, "keytabs"); +
private static final List<File> USER_KEYTAB_FILES = new ArrayList<>(); + + private static final String
SPNEGO_PRINCIPAL = "HTTP/localhost"; + private static final String PQS_PRINCIPAL =
"phoenixqs/localhost"; + private static final String SERVICE_PRINCIPAL = "securecluster/localhost"; + private
static File KEYTAB; + + private static MiniKdc KDC; + private static HBaseTestingUtility UTIL = new
HBaseTestingUtility(); + private static LocalHBaseCluster HBASE_CLUSTER; + private static int
NUM_CREATED_USERS; + + private static ExecutorService PQS_EXECUTOR; + private static QueryServer
PQS; + private static int PQS_PORT; + private static String PQS_URL; + + private static String
getTempDirForClass() { + StringBuilder sb = new StringBuilder(32); +
sb.append(System.getProperty("user.dir")).append(File.separator); + sb.append("target").append(File.separator); +
sb.append(SecureQueryServerPhoenixDBIT.class.getSimpleName()); + return sb.toString(); + } + + private static
void updateDefaultRealm() throws Exception { + // (at least) one other phoenix test triggers the caching of this
field before the KDC is up + // which causes principal parsing to fail. + Field f =
KerberosName.class.getDeclaredField("defaultRealm"); + f.setAccessible(true); + // Default realm for MiniKDC +
f.set(null, "EXAMPLE.COM"); + } + + private static void createUsers(int numUsers) throws Exception { +
assertNotNull("KDC is null, was setup method called?", KDC); + NUM_CREATED_USERS = numUsers; + for
(int i = 1; i <= numUsers; i++) { + String principal = "user" + i; + File keytabFile = new File(KEYTAB_DIR,
principal + ".keytab"); + KDC.createPrincipal(keytabFile, principal); + USER_KEYTAB_FILES.add(keytabFile);
+ } + } + + private static Entry<String,File> getUser(int offset) { + Preconditions.checkArgument(offset > 0 &&
offset <= NUM_CREATED_USERS); + return Maps.immutableEntry("user" + offset,
USER_KEYTAB_FILES.get(offset - 1)); + } + + /** + * Setup the security configuration for hdfs. + */ + private
static void setHdfsSecuredConfiguration(Configuration conf) throws Exception { + // Set principal+keytab
configuration for HDFS + conf.set(DFSConfigKeys.DFS_NAMENODE_KERBEROS_PRINCIPAL_KEY,
SERVICE_PRINCIPAL + "@" + KDC.getRealm()); +
conf.set(DFSConfigKeys.DFS_NAMENODE_KEYTAB_FILE_KEY, KEYTAB.getAbsolutePath()); +
conf.set(DFSConfigKeys.DFS_DATANODE_KERBEROS_PRINCIPAL_KEY, SERVICE_PRINCIPAL + "@" +
KDC.getRealm()); + conf.set(DFSConfigKeys.DFS_DATANODE_KEYTAB_FILE_KEY,
KEYTAB.getAbsolutePath()); +
conf.set(DFSConfigKeys.DFS_WEB_AUTHENTICATION_KERBEROS_PRINCIPAL_KEY,
SPNEGO_PRINCIPAL + "@" + KDC.getRealm()); + // Enable token access for HDFS blocks +
conf.setBoolean(DFSConfigKeys.DFS_BLOCK_ACCESS_TOKEN_ENABLE_KEY, true); + // Only use HTTPS
(required because we aren't using "secure" ports) + conf.set(DFSConfigKeys.DFS_HTTP_POLICY_KEY,
HttpConfig.Policy.HTTPS_ONLY.name()); + // Bind on localhost for spnego to have a chance at working +
conf.set(DFSConfigKeys.DFS_NAMENODE_HTTPS_ADDRESS_KEY, "localhost:0"); +
conf.set(DFSConfigKeys.DFS_DATANODE_HTTPS_ADDRESS_KEY, "localhost:0"); + + // Generate SSL certs
+ File keystoresDir = new File(UTIL.getDataTestDir("keystore").toUri().getPath()); + keystoresDir.mkdirs(); +
String sslConfDir = KeyStoreTestUtil.getClasspathDir(SecureQueryServerPhoenixDBIT.class); +
KeyStoreTestUtil.setupSSLConfig(keystoresDir.getAbsolutePath(), sslConfDir, conf, false); + + // Magic flag to
tell hdfs to not fail on using ports above 1024 + conf.setBoolean("ignore.secure.ports.for.testing", true); + } + +
private static void ensureIsEmptyDirectory(File f) throws IOException { + if (f.exists()) { + if (f.isDirectory()) { +
FileUtils.deleteDirectory(f); + } else { + assertTrue("Failed to delete keytab directory", f.delete()); + } + } +
assertTrue("Failed to create keytab directory", f.mkdirs()); + } + + /** + * Setup and start kerberos, hbase + */ +
@BeforeClass + public static void setUp() throws Exception { + final Configuration conf =
UTIL.getConfiguration(); + // Ensure the dirs we need are created/empty + ensureIsEmptyDirectory(TEMP_DIR);
+ ensureIsEmptyDirectory(KEYTAB_DIR); + KEYTAB = new File(KEYTAB_DIR, "test.keytab"); + // Start a
MiniKDC + KDC = UTIL.setupMiniKdc(KEYTAB); + // Create a service principal and spnego principal in one
keytab + // NB. Due to some apparent limitations between HDFS and HBase in the same JVM, trying to + // use
separate identies for HBase and HDFS results in a GSS initiate error. The quick + // solution is to just use a single
"service" principal instead of "hbase" and "hdfs" + // (or "dn" and "nn") per usual. +
KDC.createPrincipal(KEYTAB, SPNEGO_PRINCIPAL, PQS_PRINCIPAL, SERVICE_PRINCIPAL); + // Start
ZK by hand + UTIL.startMiniZKCluster(); + + // Create a number of unprivileged users + createUsers(3); + + //
Set configuration for HBase + HBaseKerberosUtils.setPrincipalForTesting(SERVICE_PRINCIPAL + "@" +
```

KDC.getRealm()); + HBaseKerberosUtils.setSecuredConfiguration(conf); + setHdfsSecuredConfiguration(conf); + UserGroupInformation.setConfiguration(conf); + conf.setInt(HConstants.MASTER_PORT, 0); + conf.setInt(HConstants.MASTER_INFO_PORT, 0); + conf.setInt(HConstants.REGIONSERVER_PORT, 0); + conf.setInt(HConstants.REGIONSERVER_INFO_PORT, 0); + conf.setStrings(CoprocessorHost.REGION_COPROCESSOR_CONF_KEY, + TokenProvider.class.getName()); + + // Secure Phoenix setup + conf.set("phoenix.queryserver.kerberos.http.principal", SPNEGO_PRINCIPAL + "@" + KDC.getRealm()); + conf.set("phoenix.queryserver.http.keytab.file", KEYTAB.getAbsolutePath()); + conf.set("phoenix.queryserver.kerberos.principal", PQS_PRINCIPAL + "@" + KDC.getRealm()); + conf.set("phoenix.queryserver.keytab.file", KEYTAB.getAbsolutePath()); + conf.setBoolean(QueryServices.QUERY_SERVER_DISABLE_KERBEROS_LOGIN, true); + conf.setInt(QueryServices.QUERY_SERVER_HTTP_PORT_ATTRIB, 0); + // Required so that PQS can impersonate the end-users to HBase + conf.set("hadoop.proxyuser.phoenixqs.groups", "*"); + conf.set("hadoop.proxyuser.phoenixqs.hosts", "*"); + + // Clear the cached singletons so we can inject our own. + InstanceResolver.clearSingletons(); + // Make sure the ConnectionInfo doesn't try to pull a default Configuration + InstanceResolver.getSingleton(ConfigurationFactory.class, new ConfigurationFactory() { + @Override + public Configuration getConfiguration() { + return conf; + } + @Override + public Configuration getConfiguration(Configuration confToClone) { + Configuration copy = new Configuration(conf); + copy.addResource(confToClone); + return copy; + } + }); + updateDefaultRealm(); + + // Start HDFS + UTIL.startMiniDFSCluster(1); + // Use LocalHBaseCluster to avoid HBaseTestingUtility from doing something wrong + // NB. I'm not actually sure what HTU does incorrect, but this was pulled from some test + // classes in HBase itself. I couldn't get HTU to work myself (2017/07/06) + Path rootdir = UTIL.getDataTestDirOnTestFS(SecureQueryServerPhoenixDBIT.class.getSimpleName()); + FSUtils.setRootDir(conf, rootdir); + HBASE_CLUSTER = new LocalHBaseCluster(conf, 1); + HBASE_CLUSTER.startup(); + + // Then fork a thread with PQS in it. + startQueryServer(); + } + + private static void startQueryServer() throws Exception { + PQS = new QueryServer(new String[0], UTIL.getConfiguration()); + // Get the PQS ident for PQS to use + final UserGroupInformation ugi = UserGroupInformation.loginUserFromKeytabAndReturnUGI(PQS_PRINCIPAL, KEYTAB.getAbsolutePath()); + PQS_EXECUTOR = Executors.newSingleThreadExecutor(); + // Launch PQS, doing in the Kerberos login instead of letting PQS do it itself (which would + // break the HBase/HDFS logins also running in the same test case). + PQS_EXECUTOR.submit(new Runnable() { + @Override public void run() { + ugi.doAs(new PrivilegedAction<Void>() { + @Override public Void run() { + PQS.run(); + return null; + } + }); + } + }); + PQS.awaitRunning(); + PQS_PORT = PQS.getPort(); + PQS_URL = ThinClientUtil.getConnectionUrl("localhost", PQS_PORT) + ";authentication=SPNEGO"; + } + + @AfterClass + public static void stopKdc() throws Exception { + // Remove our custom ConfigurationFactory for future tests + InstanceResolver.clearSingletons(); + if (PQS_EXECUTOR != null) { + PQS.stop(); + PQS_EXECUTOR.shutdown(); + if (!PQS_EXECUTOR.awaitTermination(5, TimeUnit.SECONDS)) { + LOG.info("PQS didn't exit in 5 seconds, proceeding anyways."); + } + } + if (HBASE_CLUSTER != null) { + HBASE_CLUSTER.shutdown(); + HBASE_CLUSTER.join(); + } + if (UTIL != null) { + UTIL.shutdownMiniZKCluster(); + } + if (KDC != null) { + KDC.stop(); + } + } + + @Test + public void testBasicReadWrite() throws Exception { + final Entry<String,File> user1 = getUser(1); + String currentDirectory; + File file = new File("."); + currentDirectory = file.getAbsolutePath(); + LOG.debug("Current working directory : "+currentDirectory); + LOG.debug("PQS_PORT:" + PQS_PORT); + LOG.debug("PQS_URL: " + PQS_URL); + ArrayList<String> cmdList = new ArrayList<>(); + // This assumes the test is being run from phoenix/phoenix-queryserver + cmdList.add(Paths.get(currentDirectory, "src", "it", "bin", "test_phoenixdb.sh").toString()); + cmdList.add(Paths.get(currentDirectory, "..", "python").toString()); + cmdList.add(user1.getKey() + "@" + KDC.getRealm()); + cmdList.add(user1.getValue().getAbsolutePath()); + String osName = System.getProperty("os.name").toLowerCase(); + LOG.info("OS is " + osName); + File krb5ConfFile = null; + if (osName.indexOf("mac") >= 0) { + int kdcPort = KDC.getPort(); + LOG.info("MINIKDC PORT " + kdcPort); + // Render a Heimdal compatible krb5.conf + // Currently kinit will only try tcp if the KDC is defined as + // kdc = tcp/hostname:port + StringBuilder krb5conf = new StringBuilder(); + krb5conf.append("[libdefaults]\n"); + krb5conf.append(" default_realm = EXAMPLE.COM\n"); + krb5conf.append(" udp_preference_limit = 1\n"); + krb5conf.append("\n"); + krb5conf.append("[realms]\n"); + krb5conf.append(" EXAMPLE.COM = {\n"); + krb5conf.append(" kdc = tcp/localhost:"); + krb5conf.append(kdcPort); + krb5conf.append("\n"); + krb5conf.append(" }\n"); + + LOG.info("Writing Heimdal style krb5.conf"); + LOG.info(krb5conf.toString()); + krb5ConfFile = File.createTempFile("krb5.conf", null); + FileOutputStream fos = new FileOutputStream(krb5ConfFile); + fos.write(krb5conf.toString().getBytes()); + fos.close(); + LOG.info("krb5.conf written to " + krb5ConfFile.getAbsolutePath()); + cmdList.add(krb5ConfFile.getAbsolutePath()); + } else { + cmdList.add(System.getProperty("java.security.krb5.conf")); + LOG.info("Using miniKDC provided krb5.conf " + KDC.getKrb5conf().getAbsolutePath()); + } + + cmdList.add(Integer.toString(PQS_PORT)); + cmdList.add(Paths.get(currentDirectory, "src", "it", "bin", "test_phoenixdb.py").toString()); + + // kinit in some random credcache --- End diff -- I'll give this one more shot in a bit

39. Github user pu239ppy commented on a diff in the pull request: https://github.com/apache/phoenix/pull/307#discussion_r201482973 --- Diff: phoenix-queryserver/pom.xml --- @@ -47,6 +47,11 @@ <plugin> <groupId>org.apache.maven.plugins</groupId> <artifactId>maven-failsafe-plugin</artifactId> + <configuration> + <excludes> + <exclude>**/SecureQueryServerPhoenixDBIT.java</exclude> --- End diff -- There are a few prerequisites - Either anaconda or virtual env *must* to be installed - System *must* provide either MIT or Heimdal kerberos utilities and libraries
40. Github user pu239ppy commented on a diff in the pull request: https://github.com/apache/phoenix/pull/307#discussion_r201716375 --- Diff: python/phoenixdb-module/phoenixdb/__init__.py --- @@ -1,11 +1,10 @@ -# Licensed to the Apache Software Foundation (ASF) under one or more -# contributor license agreements. See the NOTICE file distributed with -# this work for additional information regarding copyright ownership. -# The ASF licenses this file to You under the Apache License, Version 2.0 -# (the "License"); you may not use this file except in compliance with -# the License. You may obtain a copy of the License at +# Copyright 2015 Lukas Lalinsky --- End diff -- I am not sure how that went back in, it is possible that I may have copied __init__.py from the time I was doing this work on my own before I found out that this has been moved to phoenix. I will change the header
41. Github user pu239ppy commented on a diff in the pull request: https://github.com/apache/phoenix/pull/307#discussion_r201717820 --- Diff: phoenix-queryserver/src/it/java/org/apache/phoenix/end2end/SecureQueryServerPhoenixDBIT.java --- @@ -0,0 +1,423 @@ +/* + * Licensed to the Apache Software Foundation (ASF) under one or more + * contributor license agreements. See the NOTICE file distributed with + * this work for additional information regarding copyright ownership. + * The ASF licenses this file to you under the Apache License, Version 2.0 + * (the "License"); you may not use this file except in compliance with + * the License. You may obtain a copy of the License at + * + * http://www.apache.org/licenses/LICENSE-2.0 + * + * Unless required by applicable law or agreed to in writing, software + * distributed under the License is distributed on an "AS IS" BASIS, + * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. + * See the License for the specific language governing permissions and + * limitations under the License. + */ +package org.apache.phoenix.end2end; + +import com.google.common.base.Preconditions; +import com.google.common.collect.Maps; +import org.apache.commons.io.FileUtils; +import org.apache.commons.logging.Log; +import org.apache.commons.logging.LogFactory; +import org.apache.hadoop.conf.Configuration; +import org.apache.hadoop.fs.Path; +import org.apache.hadoop.hbase.HBaseTestingUtility; +import org.apache.hadoop.hbase.HConstants; +import org.apache.hadoop.hbase.LocalHBaseCluster; +import org.apache.hadoop.hbase.coprocessor.CoprocessorHost; +import org.apache.hadoop.hbase.http.ssl.KeyStoreTestUtil; +import org.apache.hadoop.hbase.security.HBaseKerberosUtils; +import org.apache.hadoop.hbase.security.token.TokenProvider; +import org.apache.hadoop.hbase.util.FSUtils; +import org.apache.hadoop.hdfs.DFSConfigKeys; +import org.apache.hadoop.http.HttpConfig; +import org.apache.hadoop.minikdc.MiniKdc; +import org.apache.hadoop.security.UserGroupInformation; +import org.apache.hadoop.security.authentication.util.KerberosName; +import org.apache.phoenix.query.ConfigurationFactory; +import org.apache.phoenix.query.QueryServices; +import org.apache.phoenix.queryserver.client.ThinClientUtil; +import org.apache.phoenix.queryserver.server.QueryServer; +import org.apache.phoenix.util.InstanceResolver; +import org.junit.AfterClass; +import org.junit.BeforeClass; +import org.junit.Test; +import org.junit.experimental.categories.Category; + +import java.io.*; +import java.lang.reflect.Field; +import java.security.PrivilegedAction; +import java.security.PrivilegedExceptionAction; +import java.sql.DriverManager; +import java.sql.ResultSet; +import java.sql.Statement; +import java.util.ArrayList; +import java.util.List; +import java.util.Map.Entry; +import java.util.concurrent.ExecutorService; +import java.util.concurrent.Executors; +import java.util.concurrent.TimeUnit; + +import java.nio.file.Paths; +import java.util.Map; + +import static org.junit.Assert.*; + +@Category(NeedsOwnMiniClusterTest.class) +public class SecureQueryServerPhoenixDBIT { + private static final Log LOG = LogFactory.getLog(SecureQueryServerPhoenixDBIT.class); + + private static final File TEMP_DIR = new File(getTempDirForClass()); + private static final File KEYTAB_DIR = new File(TEMP_DIR, "keytabs"); + private static final List<File> USER_KEYTAB_FILES = new ArrayList<>(); + + private static final String SPNEGO_PRINCIPAL = "HTTP/localhost"; + private static final String PQS_PRINCIPAL = "phoenixqs/localhost"; + private static final String SERVICE_PRINCIPAL = "securecluster/localhost"; + private static File KEYTAB; + + private static MiniKdc KDC; + private static HBaseTestingUtility UTIL = new HBaseTestingUtility(); + private static LocalHBaseCluster HBASE_CLUSTER; + private static int NUM_CREATED_USERS; + + private static ExecutorService PQS_EXECUTOR; + private static QueryServer PQS; + private static int PQS_PORT; + private static String PQS_URL; + + private static String getTempDirForClass() { + StringBuilder sb = new StringBuilder(32); + sb.append(System.getProperty("user.dir")).append(File.separator); + sb.append("target").append(File.separator); + sb.append(SecureQueryServerPhoenixDBIT.class.getSimpleName()); + return sb.toString(); + } + + private static void updateDefaultRealm() throws Exception { + // (at least) one other phoenix test triggers the caching of this

field before the KDC is up + // which causes principal parsing to fail. + Field f = KerberosName.class.getDeclaredField("defaultRealm"); + f.setAccessible(true); + // Default realm for MiniKDC + f.set(null, "EXAMPLE.COM"); + } + + private static void createUsers(int numUsers) throws Exception { + assertNotNull("KDC is null, was setup method called?", KDC); + NUM_CREATED_USERS = numUsers; + for (int i = 1; i <= numUsers; i++) { + String principal = "user" + i; + File keytabFile = new File(KEYTAB_DIR, principal + ".keytab"); + KDC.createPrincipal(keytabFile, principal); + USER_KEYTAB_FILES.add(keytabFile); + } + } + + private static Entry<String,File> getUser(int offset) { + Preconditions.checkArgument(offset > 0 && offset <= NUM_CREATED_USERS); + return Maps.immutableEntry("user" + offset, USER_KEYTAB_FILES.get(offset - 1)); + } + + /** + * Setup the security configuration for hdfs. + */ + private static void setHdfsSecuredConfiguration(Configuration conf) throws Exception { + // Set principal+keytab configuration for HDFS + conf.set(DFSConfigKeys.DFS_NAMENODE_KERBEROS_PRINCIPAL_KEY, SERVICE_PRINCIPAL + "@" + KDC.getRealm()); + conf.set(DFSConfigKeys.DFS_NAMENODE_KEYTAB_FILE_KEY, KEYTAB.getAbsolutePath()); + conf.set(DFSConfigKeys.DFS_DATANODE_KERBEROS_PRINCIPAL_KEY, SERVICE_PRINCIPAL + "@" + KDC.getRealm()); + conf.set(DFSConfigKeys.DFS_DATANODE_KEYTAB_FILE_KEY, KEYTAB.getAbsolutePath()); + conf.set(DFSConfigKeys.DFS_WEB_AUTHENTICATION_KERBEROS_PRINCIPAL_KEY, SPNEGO_PRINCIPAL + "@" + KDC.getRealm()); + // Enable token access for HDFS blocks + conf.setBoolean(DFSConfigKeys.DFS_BLOCK_ACCESS_TOKEN_ENABLE_KEY, true); + // Only use HTTPS (required because we aren't using "secure" ports) + conf.set(DFSConfigKeys.DFS_HTTP_POLICY_KEY, HttpConfig.Policy.HTTPS_ONLY.name()); + // Bind on localhost for spnego to have a chance at working + conf.set(DFSConfigKeys.DFS_NAMENODE_HTTPS_ADDRESS_KEY, "localhost:0"); + conf.set(DFSConfigKeys.DFS_DATANODE_HTTPS_ADDRESS_KEY, "localhost:0"); + + // Generate SSL certs + File keystoresDir = new File(UTIL.getDataTestDir("keystore").toUri().getPath()); + keystoresDir.mkdirs(); + String sslConfDir = KeyStoreTestUtil.getClasspathDir(SecureQueryServerPhoenixDBIT.class); + KeyStoreTestUtil.setupSSLConfig(keystoresDir.getAbsolutePath(), sslConfDir, conf, false); + + // Magic flag to tell hdfs to not fail on using ports above 1024 + conf.setBoolean("ignore.secure.ports.for.testing", true); + } + + private static void ensureIsEmptyDirectory(File f) throws IOException { + if (f.exists()) { + if (f.isDirectory()) { + FileUtils.deleteDirectory(f); + } else { + assertTrue("Failed to delete keytab directory", f.delete()); + } + } + assertTrue("Failed to create keytab directory", f.mkdirs()); + } + + /** + * Setup and start kerberos, hbase + */ + @BeforeClass + public static void setUp() throws Exception { + final Configuration conf = UTIL.getConfiguration(); + // Ensure the dirs we need are created/empty + ensureIsEmptyDirectory(TEMP_DIR); + ensureIsEmptyDirectory(KEYTAB_DIR); + KEYTAB = new File(KEYTAB_DIR, "test.keytab"); + // Start a MiniKDC + KDC = UTIL.setupMiniKdc(KEYTAB); + // Create a service principal and spnego principal in one keytab + // NB. Due to some apparent limitations between HDFS and HBase in the same JVM, trying to + // use separate identies for HBase and HDFS results in a GSS initiate error. The quick + // solution is to just use a single "service" principal instead of "hbase" and "hdfs" + // (or "dn" and "nn") per usual. + KDC.createPrincipal(KEYTAB, SPNEGO_PRINCIPAL, PQS_PRINCIPAL, SERVICE_PRINCIPAL); + // Start ZK by hand + UTIL.startMiniZKCluster(); + + // Create a number of unprivileged users + createUsers(3); + + // Set configuration for HBase + HBaseKerberosUtils.setPrincipalForTesting(SERVICE_PRINCIPAL + "@" + KDC.getRealm()); + HBaseKerberosUtils.setSecuredConfiguration(conf); + setHdfsSecuredConfiguration(conf); + UserGroupInformation.setConfiguration(conf); + conf.setInt(HConstants.MASTER_PORT, 0); + conf.setInt(HConstants.MASTER_INFO_PORT, 0); + conf.setInt(HConstants.REGIONSERVER_PORT, 0); + conf.setInt(HConstants.REGIONSERVER_INFO_PORT, 0); + conf.setStrings(CoprocessorHost.REGION_COPROCESSOR_CONF_KEY, + TokenProvider.class.getName()); + + // Secure Phoenix setup + conf.set("phoenix.queryserver.kerberos.http.principal", SPNEGO_PRINCIPAL + "@" + KDC.getRealm()); + conf.set("phoenix.queryserver.http.keytab.file", KEYTAB.getAbsolutePath()); + conf.set("phoenix.queryserver.kerberos.principal", PQS_PRINCIPAL + "@" + KDC.getRealm()); + conf.set("phoenix.queryserver.keytab.file", KEYTAB.getAbsolutePath()); + conf.setBoolean(QueryServices.QUERY_SERVER_DISABLE_KERBEROS_LOGIN, true); + conf.setInt(QueryServices.QUERY_SERVER_HTTP_PORT_ATTRIB, 0); + // Required so that PQS can impersonate the end-users to HBase + conf.set("hadoop.proxyuser.phoenixqs.groups", "*"); + conf.set("hadoop.proxyuser.phoenixqs.hosts", "*"); + + // Clear the cached singletons so we can inject our own. + InstanceResolver.clearSingletons(); + // Make sure the ConnectionInfo doesn't try to pull a default Configuration + InstanceResolver.getSingleton(ConfigurationFactory.class, new ConfigurationFactory() { + @Override + public Configuration getConfiguration() { + return conf; + } + @Override + public Configuration getConfiguration(Configuration confToClone) { + Configuration copy = new Configuration(conf); + copy.addResource(confToClone); + return copy; + } + }); + updateDefaultRealm(); + + // Start HDFS + UTIL.startMiniDFSCluster(1); + // Use LocalHBaseCluster to avoid HBaseTestingUtility from doing something wrong + // NB. I'm not actually sure what HTU does incorrect, but this was pulled from some test + // classes in HBase itself. I couldn't get HTU to work myself (2017/07/06) + Path rootdir = UTIL.getDataTestDirOnTestFS(SecureQueryServerPhoenixDBIT.class.getSimpleName()); + FSUtils.setRootDir(conf, rootdir); + HBASE_CLUSTER = new LocalHBaseCluster(conf, 1); +

HBASE_CLUSTER.startup(); + + // Then fork a thread with PQS in it. + startQueryServer(); + } + + private static void startQueryServer() throws Exception { + PQS = new QueryServer(new String[0], UTIL.getConfiguration()); + // Get the PQS ident for PQS to use + final UserGroupInformation ugi = UserGroupInformation.loginUserFromKeytabAndReturnUGI(PQS_PRINCIPAL, KEYTAB.getAbsolutePath()); + PQS_EXECUTOR = Executors.newSingleThreadExecutor(); + // Launch PQS, doing in the Kerberos login instead of letting PQS do it itself (which would + // break the HBase/HDFS logins also running in the same test case). + PQS_EXECUTOR.submit(new Runnable() { + @Override public void run() { + ugi.doAs(new PrivilegedAction<Void>() { + @Override public Void run() { + PQS.run(); + return null; + } + }); + } + }); + PQS.awaitRunning(); + PQS_PORT = PQS.getPort(); + PQS_URL = ThinClientUtil.getConnectionUrl("localhost", PQS_PORT) + ";authentication=SPNEGO"; + } + + @AfterClass + public static void stopKdc() throws Exception { + // Remove our custom ConfigurationFactory for future tests + InstanceResolver.clearSingletons(); + if (PQS_EXECUTOR != null) { + PQS.stop(); + PQS_EXECUTOR.shutdown(); + if (!PQS_EXECUTOR.awaitTermination(5, TimeUnit.SECONDS)) { + LOG.info("PQS didn't exit in 5 seconds, proceeding anyways."); + } + } + if (HBASE_CLUSTER != null) { + HBASE_CLUSTER.shutdown(); + HBASE_CLUSTER.join(); + } + if (UTIL != null) { + UTIL.shutdownMiniZKCluster(); + } + if (KDC != null) { + KDC.stop(); + } + } + + @Test + public void testBasicReadWrite() throws Exception { + final Entry<String,File> user1 = getUser(1); + String currentDirectory; + File file = new File("."); + currentDirectory = file.getAbsolutePath(); + LOG.debug("Current working directory : "+currentDirectory); + LOG.debug("PQS_PORT:" + PQS_PORT); + LOG.debug("PQS_URL: " + PQS_URL); + ArrayList<String> cmdList = new ArrayList<>(); + // This assumes the test is being run from phoenix/phoenix-queryserver + cmdList.add(Paths.get(currentDirectory, "src", "it", "bin", "test_phoenixdb.sh").toString()); + cmdList.add(Paths.get(currentDirectory, "..", "python").toString()); + cmdList.add(user1.getKey() + "@" + KDC.getRealm()); + cmdList.add(user1.getValue().getAbsolutePath()); + String osName = System.getProperty("os.name").toLowerCase(); + LOG.info("OS is " + osName); + File krb5ConfFile = null; + if (osName.indexOf("mac") >= 0) { + int kdcPort = KDC.getPort(); + LOG.info("MINIKDC PORT " + kdcPort); + // Render a Heimdal compatible krb5.conf + // Currently kinit will only try tcp if the KDC is defined as + // kdc = tcp/hostname:port --- End diff -- Yes mini KDC will render a krb5.conf file, however it is useless on MAC OS as Heimdal seemingly decided to want to specify protocols as opposed to trying them all. This has been fixed but Apple has not packaged it yet I guess

42. Github user pu239ppy commented on the issue: https://github.com/apache/phoenix/pull/307 @joshelser If you are still unable to get .deactivate to work, remove it from the shell script for now and we can revisit it. Again I am being overly pedantic here, but a the shell exits, we really do not need to clean up any environment.
43. Github user joshelser commented on a diff in the pull request: https://github.com/apache/phoenix/pull/307#discussion_r202106445 --- Diff: phoenix-queryserver/src/it/bin/test_phoenixdb.sh --- @@ -0,0 +1,59 @@ +#/usr/bin/env bash + +set -u +set -x +set -e + +function cleanup { + set +e + set +u + kdestroy --- End diff -- Ok, cool. I didn't think kdestroy was doing more than just cleaning up those token :)
44. Github user joshelser commented on a diff in the pull request: https://github.com/apache/phoenix/pull/307#discussion_r202108057 --- Diff: phoenix-queryserver/src/it/bin/test_phoenixdb.sh --- @@ -0,0 +1,59 @@ +#/usr/bin/env bash + +set -u +set -x +set -e + +function cleanup { + set +e + set +u + kdestroy + pushd ${PY_ENV_PATH}/bin + . deactivate "" + popd + rm -rf $PY_ENV_PATH +} + +trap cleanup EXIT + +echo "LAUNCHING SCRIPT" + +LOCAL_PY=$1 +PRINC=$2 +KEYTAB_LOC=$3 +KRB5_CFG_FILE=$4 +PQS_PORT=$5 +PYTHON_SCRIPT=$6 + +PY_ENV_PATH=$(mktemp -d) + +conda create -y -p $PY_ENV_PATH || virtualenv $PY_ENV_PATH + +pushd ${PY_ENV_PATH}/bin + +# conda activate does stuff with unbound variables :(+set +u +. activate "" + +popd + +set -u +echo "INSTALLING COMPONENTS" +pip install -e file:///${LOCAL_PY}/requests-kerberos +pip install -e file:///${LOCAL_PY}/phoenixdb-module + +export KRB5_CONFIG=$KRB5_CFG_FILE +cat $KRB5_CONFIG +export KRB5_TRACE=/dev/stdout + +#echo "RUNNING KINIT" +kinit -kt $KEYTAB_LOC $PRINC --- End diff -- > I just tried on the command line and MAC OS (Heimdal) kinit does not require a directory Yeah, convention is to use `${tmpdir}/krb5cc_$(current-user uid)`. > pass the when executing python or just continue running in the same shell, which is why I stopped attempts to make ny further reductions to the shell script Oh right, I forgot they would bash the environment. Let's just let this be for now. Will be easier to come back to it later.
45. Github user joshelser commented on the issue: https://github.com/apache/phoenix/pull/307 > If you are still unable to get .deactivate to work, remove it from the shell script for now and we can revisit it. Again I am being overly pedantic here, but a the shell exits, we really do not need to clean up any environment. Ok. I'm looking at this again today. Thanks for the feedback so far. The test/code looks good to me. I think we just need to update documentation to explain what we're doing (maybe a README.md in `python/`?)
46. Github user joshelser commented on the issue: https://github.com/apache/phoenix/pull/307 Ugh, I'm getting frustrated: I have MIT kerberos on my Mac, so I unblocked myself first by just forcing the minikdc config file to be made instead of the if-branch you added, Lev. The next thing, I get a failure trying to launch python from the virtualenv: ``` /Users/jelser/projects/phoenix.git/phoenix-queryserver/./src/it/bin/test_phoenixdb.sh: line 59: 66933 Illegal instruction: 4 python $PYTHON_SCRIPT $PQS_PORT ``` This was reproducible doing it by hand, so I

thought maybe it was related to me using python-2.7.14 (old). So, I switched over to Python-3.6.4, reinstalled everything, and I got this. ``` Traceback (most recent call last): File "/private/var/folders/4q/q02ykc2j5l1fg8nbs_sczskh0000gp/T/tmp.iUMwkyIZ/lib/python3.6/site-packages/requests_kerberos/kerberos_.py", line 2, in <module> import kerberos ImportError: dlopen(/private/var/folders/4q/q02ykc2j5l1fg8nbs_sczskh0000gp/T/tmp.iUMwkyIZ/lib/python3.6/site-packages/kerberos.cpython-36m-darwin.so, 2): Symbol not found: _mempcpy Referenced from: /private/var/folders/4q/q02ykc2j5l1fg8nbs_sczskh0000gp/T/tmp.iUMwkyIZ/lib/python3.6/site-packages/kerberos.cpython-36m-darwin.so Expected in: flat namespace in /private/var/folders/4q/q02ykc2j5l1fg8nbs_sczskh0000gp/T/tmp.iUMwkyIZ/lib/python3.6/site-packages/kerberos.cpython-36m-darwin.so ``` I ran into another GH issue saying that pykerberos==1.1.14 fixed it for them, but I'm not seeing a difference locally. How do you feel about requiring Docker, @pu239ppy? ;)

47. Github user pu239ppy commented on the issue: https://github.com/apache/phoenix/pull/307 @joshelser Not sure how you got MIT on Mac OS X, is it in some ports package? I can try this later on ubuntu perhaps, if you want a test with MIT. I suppose integrating docker into the mix would make things interesting. I'll try it over the weekend if I get the time
48. Github user joshelser commented on the issue: https://github.com/apache/phoenix/pull/307 > Not sure how you got MIT on Mac OS X, is it in some ports package Yeah, homebrew has a `krb5` package which I use for running stuff locally (e.g. I put it on the `PATH` before the Heimdal, osx-provided variant) > I suppose integrating docker into the mix would make things interesting. I'll try it over the weekend if I get the time. I'm playing with different versions of python now, but am just worried about the feasibility of this actually working on the general person's machine given how much I'm struggling :\
49. Github user joshelser commented on the issue: https://github.com/apache/phoenix/pull/307 Ah, I think my issue might have been cruft sitting in `python/requests-kerberos/`, the `build` dir and the `.egg-info` dir. Getting a straightforward HTTP/401 error now. That I know how to deal with :)
50. Github user joshelser commented on the issue: https://github.com/apache/phoenix/pull/307 Ok, where I'm at now: * Python 2.7.15 (installed via pyenv) * Using virtualenv to circumvent the .sh script * Modified the junit test to just leave it running * Modified the junit test to just use the minikdc's kdc.conf * Pulled back the pykerberos dependency to 1.1.14 to get past an "illegal instruction error" that I get with pykerberos-1.2.1 (or whatever pip found) This gets the phoenixdb client to actually submit the initial POST and get the `WWW-Authenticate: Negotiate` header back. However, my client seems to be unable to generate its challenge data from our mini kdc: ``` DEBUG:phoenixdb.avatica.client:POST http://localhost:60358/ '\n@org.apache.calcite.avatica.proto.Requests$CloseConnectionRequest\x12&\n$f71fb5c5-a814-4766-9691-8aeddfc0eea4' {'content-type': 'application/x-google-protobuf'} DEBUG:urllib3.connectionpool:Starting new HTTP connection (1): localhost:60358 send: 'POST / HTTP/1.1\r\nHost: localhost:60358\r\nConnection: keep-alive\r\nAccept-Encoding: gzip, deflate\r\nAccept: */*\r\nUser-Agent: python-requests/2.19.1\r\ncontent-type: application/x-google-protobuf\r\nContent-Length: 106\r\n\r\n\n@org.apache.calcite.avatica.proto.Requests$CloseConnectionRequest\x12&\n$f71fb5c5-a814-4766-9691-8aeddfc0eea4' reply: 'HTTP/1.1 401 Unauthorized\r\n' header: Date: Fri, 13 Jul 2018 17:06:02 GMT header: WWW-Authenticate: Negotiate header: Cache-Control: must-revalidate,no-cache,no-store header: Content-Type: text/html; charset=ISO-8859-1 header: Content-Length: 281 header: Server: Jetty(9.2.19.v20160908) DEBUG:urllib3.connectionpool:http://localhost:60358 "POST / HTTP/1.1" 401 281 DEBUG:requests_kerberos.kerberos_:handle_401(): Handling: 401 ERROR:requests_kerberos.kerberos_:generate_request_header(): authGSSClientStep() failed: Traceback (most recent call last): File "/Users/jelser/projects/phoenix.git/python/requests-kerberos/requests_kerberos/kerberos_.py", line 235, in generate_request_header negotiate_resp_value) GSSError: (('Unspecified GSS failure. Minor code may provide more information', 851968), ('Message stream modified', 100001)) ERROR:requests_kerberos.kerberos_:(('Unspecified GSS failure. Minor code may provide more information', 851968), ('Message stream modified', 100001)) Traceback (most recent call last): File "/Users/jelser/projects/phoenix.git/python/requests-kerberos/requests_kerberos/kerberos_.py", line 235, in generate_request_header negotiate_resp_value) GSSError: (('Unspecified GSS failure. Minor code may provide more information', 851968), ('Message stream modified', 100001)) ``` I can't seem to unwrap what's wrong with the request to the KDC which is preventing that from happening. Need to find more debug...
51. Github user joshelser commented on the issue: https://github.com/apache/phoenix/pull/307 Turning back on `KRB5_TRACE`... ``` DEBUG:phoenixdb.avatica.client:POST http://localhost:60358/ '\n?org.apache.calcite.avatica.proto.Requests$OpenConnectionRequest\x12&\n$386e3317-e23e-4a0e-9fc6-2efaa546ffc4' {'content-type': 'application/x-google-protobuf'} DEBUG:urllib3.connectionpool:Starting new HTTP connection (1): localhost:60358 send: 'POST / HTTP/1.1\r\nHost: localhost:60358\r\nConnection: keep-alive\r\nAccept-Encoding: gzip, deflate\r\nAccept: */*\r\nUser-Agent: python-requests/2.19.1\r\ncontent-type: application/x-google-protobuf\r\nContent-Length: 105\r\n\r\n\n?org.apache.calcite.avatica.proto.Requests$OpenConnectionRequest\x12&\n$386e3317-e23e-4a0e-9fc6-2efaa546ffc4' reply: 'HTTP/1.1 401 Unauthorized\r\n' header: Date: Fri, 13 Jul 2018 17:23:46 GMT header: WWW-Authenticate: Negotiate header: Cache-Control: must-revalidate,no-cache,no-store header: Content-Type: text/html; charset=ISO-8859-1 header: Content-Length: 281 header: Server: Jetty(9.2.19.v20160908)

DEBUG:urllib3.connectionpool:http://localhost:60358 "POST / HTTP/1.1" 401 281 DEBUG:requests_kerberos.kerberos_:handle_401(): Handling: 401 [28575] 1531502626.856661: ccselect module realm chose cache FILE:/tmp/krb5cc_502 with client principal user1@EXAMPLE.COM for server principal HTTP/localhost@EXAMPLE.COM [28575] 1531502626.856662: Getting credentials user1@EXAMPLE.COM -> HTTP/localhost@ using ccache FILE:/tmp/krb5cc_502 [28575] 1531502626.856663: Retrieving user1@EXAMPLE.COM -> HTTP/localhost@ from FILE:/tmp/krb5cc_502 with result: -1765328243/Matching credential not found (filename: /tmp/krb5cc_502) [28575] 1531502626.856664: Retrying user1@EXAMPLE.COM -> HTTP/localhost@EXAMPLE.COM with result: -1765328243/Matching credential not found (filename: /tmp/krb5cc_502) [28575] 1531502626.856665: Server has referral realm; starting with HTTP/localhost@EXAMPLE.COM [28575] 1531502626.856666: Retrieving user1@EXAMPLE.COM -> krbtgt/EXAMPLE.COM@EXAMPLE.COM from FILE:/tmp/krb5cc_502 with result: 0/Success [28575] 1531502626.856667: Starting with TGT for client realm: user1@EXAMPLE.COM -> krbtgt/EXAMPLE.COM@EXAMPLE.COM [28575] 1531502626.856668: Requesting tickets for HTTP/localhost@EXAMPLE.COM, referrals on [28575] 1531502626.856669: Generated subkey for TGS request: aes128-cts/86C4 [28575] 1531502626.856670: etypes requested in TGS request: aes256-cts, aes128-cts, aes256-sha2, aes128-sha2, des3-cbc-sha1, rc4-hmac, camellia128-cts, camellia256-cts [28575] 1531502626.856672: Encoding request body and padata into FAST request [28575] 1531502626.856673: Sending request (807 bytes) to EXAMPLE.COM [28575] 1531502626.856674: Resolving hostname localhost [28575] 1531502626.856675: Initiating TCP connection to stream ::1:60299 [28575] 1531502626.856676: Terminating TCP connection to stream ::1:60299 [28575] 1531502626.856677: Initiating TCP connection to stream 127.0.0.1:60299 [28575] 1531502626.856678: Sending TCP request to stream 127.0.0.1:60299 [28575] 1531502626.856679: Received answer (119 bytes) from stream 127.0.0.1:60299 [28575] 1531502626.856680: Terminating TCP connection to stream 127.0.0.1:60299 [28575] 1531502626.856681: Sending DNS URI query for _kerberos.EXAMPLE.COM. [28575] 1531502626.856682: No URI records found [28575] 1531502626.856683: Sending DNS SRV query for _kerberos-master._udp.EXAMPLE.COM. [28575] 1531502626.856684: Sending DNS SRV query for _kerberos-master._tcp.EXAMPLE.COM. [28575] 1531502626.856685: No SRV records found [28575] 1531502626.856686: Response was not from master KDC [28575] 1531502626.856687: TGS request result: -1765328343/Message stream modified [28575] 1531502626.856688: Requesting tickets for HTTP/localhost@EXAMPLE.COM, referrals off [28575] 1531502626.856689: Generated subkey for TGS request: aes128-cts/F96F [28575] 1531502626.856690: etypes requested in TGS request: aes256-cts, aes128-cts, aes256-sha2, aes128-sha2, des3-cbc-sha1, rc4-hmac, camellia128-cts, camellia256-cts [28575] 1531502626.856692: Encoding request body and padata into FAST request [28575] 1531502626.856693: Sending request (807 bytes) to EXAMPLE.COM [28575] 1531502626.856694: Resolving hostname localhost [28575] 1531502626.856695: Initiating TCP connection to stream ::1:60299 [28575] 1531502626.856696: Terminating TCP connection to stream ::1:60299 [28575] 1531502626.856697: Initiating TCP connection to stream 127.0.0.1:60299 [28575] 1531502626.856698: Sending TCP request to stream 127.0.0.1:60299 [28575] 1531502626.856699: Received answer (119 bytes) from stream 127.0.0.1:60299 [28575] 1531502626.856700: Terminating TCP connection to stream 127.0.0.1:60299 [28575] 1531502626.856701: Sending DNS URI query for _kerberos.EXAMPLE.COM. [28575] 1531502626.856702: No URI records found [28575] 1531502626.856703: Sending DNS SRV query for _kerberos-master._udp.EXAMPLE.COM. [28575] 1531502626.856704: Sending DNS SRV query for _kerberos-master._tcp.EXAMPLE.COM. [28575] 1531502626.856705: No SRV records found [28575] 1531502626.856706: Response was not from master KDC [28575] 1531502626.856707: TGS request result: -1765328343/Message stream modified ERROR:requests_kerberos.kerberos_:generate_request_header(): authGSSClientStep() failed: Traceback (most recent call last): File "/Users/jelser/projects/phoenix.git/python/requests-kerberos/requests_kerberos/kerberos_.py", line 235, in generate_request_header negotiate_resp_value) GSSError: (('Unspecified GSS failure. Minor code may provide more information', 851968), ('Message stream modified', 100001)) ERROR:requests_kerberos.kerberos_:(('Unspecified GSS failure. Minor code may provide more information', 851968), ('Message stream modified', 100001)) Traceback (most recent call last): File "/Users/jelser/projects/phoenix.git/python/requests-kerberos/requests_kerberos/kerberos_.py", line 235, in generate_request_header negotiate_resp_value) GSSError: (('Unspecified GSS failure. Minor code may provide more information', 851968), ('Message stream modified', 100001)) ``` So, definitely the KDC throwing a fit and telling us to go away: `[28575] 1531502626.856707: TGS request result: -1765328343/Message stream modified`

52. Github user joshelser commented on the issue: https://github.com/apache/phoenix/pull/307 If there's a bright-side, it's something in my environment. Spinning up the JDBC thin client against this setup works: ``` $ PHOENIX_OPTS="$PHOENIX_OPTS -Dsun.security.krb5.debug=true -Djava.security.krb5.conf=/Users/jelser/projects/phoenix.git/phoenix-queryserver/target/test-data/8bc1abb8-79fa-4beb-aa56-fe3ae4edff64/kdc/1531499757782/krb5.conf " /usr/local/lib/phoenix-4.14.0-HBase-1.4/bin/sqlline-thin.py http://localhost:60358 -a SPNEGO ```
53. Github user joshelser commented on the issue: https://github.com/apache/phoenix/pull/307 On the note about docker, trying to think about this, I feel like a docker environment that can spin up all of the necessary stuff to run both of the python tests as well as this new one will be best. Essentially, in the docker container, we have PQS up with all of the necessary environment stuff which will make all of our current tests (and any future test) that much

easier to automate. I'm also happy to try to help with that. I know you've spent a bunch of time on this already @pu239ppy

54. **body:** Not a bad idea, I have been a little busy this week, but will try to get to this soon. If you have any suggestions on how to instrument this, let me know. I am assuming all we need is a well written docker file **label:** code-design
55. {quote}If you have any suggestions on how to instrument this, let me know. I am assuming all we need is a well written docker file {quote} Ditto :) My thinking would be that we make a Maven module which creates a Jar capable of running PQS in a standalone context (e..g remove the context of setting up hdfs/hbase/pqs in a Java test class). I did this up in Avatica [https://github.com/apache/calcite-avatica/tree/master/standalone-server|https://github.com/apache/calcite-avatica/tree/master/standalone-server,]; but Phoenix is a bit more complicated. From here, we'd have a jar that we could set up an environment as simple as "java -jar" which is nice. Then, we could have a Maven module which consumes this jar, provides that Dockerfile, and runs any/all tests we have. The perk of this approach is that we could also run the rest of the Python tests this way which would be nice.
56. May i ask what's the progress now? Or anything i can help?
57. Hoping to restart work this week. Need to figure out how how to run standalone PQS, although I am actually fine with the testing mechanism we have in place now. Either way I see 1. Standalone PQS 2. KDC ?? Also the requests kerberos people got back to me and informed me that I should use [https://github.com/pythongssapi/requests-gssapi] which I need to investigate, which if works makes dependency management simpler
58. [~reidchan], thanks for checking in. I think the biggest problem was that I couldn't get Lev's changes to work for me locally, even in a standalone environment. {quote}Also the requests kerberos people got back to me and informed me that I should use [https://github.com/pythongssapi/requests-gssapi] which I need to investigate, which if works makes dependency management simpler {quote} (eyeroll) but at least they eventually got back to you, I suppose. At this point, I think I'd be +1 if I can show it working myself (but having some kind of automated test would be really awesome â€“ I don't much care how we do it at this point).
59. {quote} Also the requests kerberos people got back to me and informed me that I should use https://github.com/pythongssapi/requests-gssapi which I need to investigate, which if works makes dependency management simpler {quote} Good to know, expecting your new pr. {quote} I think the biggest problem was that I couldn't get Lev's changes to work for me locally, even in a standalone environment. {quote} Found a place i can help, i will test those changes as well.
60. Some good news, I revisited using avatica 1.11 (I needed https://issues.apache.org/jira/browse/CALCITE-1922) and no longer need the modified libraries. However I now remember why I stuck to 1.10 in the first place, the surefire test got stuck and then exited with [ERROR] Caused by: org.apache.maven.surefire.booter.SurefireBooterForkException: The forked VM terminated without properly saying goodbye. VM crash or System.exit called? However I can see proper output in the output file, so perhaps I can deal with this later. As an aside requests-gssapi has the same drawback as requests-kerberos, however requests-kerberos used python-kerberos which included SPNEGO mechanism and could be used, python gsssapi has yet to even mention SPNEGO
61. {quote}However I now remember why I stuck to 1.10 in the first place, the surefire test got stuck and then exited with {quote} Hrm. Yeah, if it's not an immediate blocker, I can also help there. Should be able to track that one down without too much pain.
62. Josh, how would we generate a standalone binary? I was thinking of maybe just copying the contents of target post build
63. Perhaps just include the testing scaffolding? <artifactSet> <includes> <include>org.apache.calcite.avatica:*</include> <include>org.eclipse.jetty:*</include> <include>javax.servlet:*</include> <include>org.apache.hbase</include> <include>org.apache.hbase:hbase-testing-util</include> <include>org.apache.hadoop:hadoop-hdfs</include> <include>org.apache.hadoop:hadoop-minikdc</include> <include>org.apache.hadoop:hadoop-minicluster</include> </includes> </artifactSet> I added another execution target called kitchensink
64. Github user joshelser commented on the issue: https://github.com/apache/phoenix/pull/307 Thanks for the update, @pu239ppy. I think you got bit by 5.x moving over to master. Any chance you could throw a rebase on here? Holler if it's ready for me to look at, too. I see your new jar for testing.
65. Having a lot of conflicts with rebase? Even merging upstream master into local master.
66. I decided to just move my changes over to master as rebase proved intractable.
67. GitHub user pu239ppy opened a pull request: https://github.com/apache/phoenix/pull/344 PHOENIX-4688 Kerberize python phoenixdb -- Phoenix 5 You can merge this pull request into a Git repository by running: $ git pull https://github.com/pu239ppy/phoenix PHOENIX-4688.master.5 Alternatively you can review and apply these changes as the patch at: https://github.com/apache/phoenix/pull/344.patch To close this pull request, make a commit to your master/trunk branch with (at least) the following in the commit message: This closes #344 ---- commit 0a094bfba3c1776c222103cbc53f022744e23617 Author: Lev Bronshtein <lbronshtein@...> Date: 2018-08-29T21:19:51Z add phonixdb gssapi on master ----
68. Github user pu239ppy closed the pull request at: https://github.com/apache/phoenix/pull/307

69. I filed a new PR [https://github.com/apache/phoenix/pull/344,] as I do not want to blow away the existing branch and all the history and commentary in https://github.com/apache/phoenix/pull/307
70. Need to figure out why I am seeing this now {quote}2018-09-07 12:51:00,349 ERROR [main] end2end.SecureQueryServerPhoenixDBIT(352): + echo 'RUN PYTHON TEST on port 53029' 2018-09-07 12:51:00,349 ERROR [main] end2end.SecureQueryServerPhoenixDBIT(352): + python /Users/lbronshtein/DEV/phoenix/phoenix-queryserver/./src/ it/bin/test_phoenixdb.py 53029 2018-09-07 12:51:00,349 ERROR [main] end2end.SecureQueryServerPhoenixDBIT(352): /Users/lbronshtein/DEV/phoenix/python/phoenixdb/avatica/client.py: 123: RuntimeWarning: Unexpected end-group tag: Not all data was converted 2018-09-07 12:51:00,349 ERROR [main] end2end.SecureQueryServerPhoenixDBIT(352): err.ParseFromString(message.wrapped_message) 2018-09-07 12:51:00,349 ERROR [main] end2end.SecureQueryServerPhoenixDBIT(352): Traceback (most recent call last):(2018-09-07 12:51:00,349 ERROR [main] end2end.SecureQueryServerPhoenixDBIT(352): File "/Users/lbronshtein/DEV/phoenix/phoenix-queryserver/./src/i t/bin/test_phoenixdb.py", line 11, in <module> 2018-09-07 12:51:00,349 ERROR [main] end2end.SecureQueryServerPhoenixDBIT(352): conn = phoenixdb.connect(database_url, autocommit=True, auth=" SPNEGO") 2018-09-07 12:51:00,350 ERROR [main] end2end.SecureQueryServerPhoenixDBIT(352): File "/Users/lbronshtein/DEV/phoenix/python/phoenixdb/__init__.p y", line 72, in connect 2018-09-07 12:51:00,350 ERROR [main] end2end.SecureQueryServerPhoenixDBIT(352): return Connection(client, **kwargs) 2018-09-07 12:51:00,350 ERROR [main] end2end.SecureQueryServerPhoenixDBIT(352): File "/Users/lbronshtein/DEV/phoenix/python/phoenixdb/connection .py", line 57, in __init__ 2018-09-07 12:51:00,350 ERROR [main] end2end.SecureQueryServerPhoenixDBIT(352): self.open() 2018-09-07 12:51:00,350 ERROR [main] end2end.SecureQueryServerPhoenixDBIT(352): File "/Users/lbronshtein/DEV/phoenix/python/phoenixdb/connection .py", line 74, in open 2018-09-07 12:51:00,350 ERROR [main] end2end.SecureQueryServerPhoenixDBIT(352): self._client.open_connection(self._id, info=self._connection_a rgs) 2018-09-07 12:51:00,350 ERROR [main] end2end.SecureQueryServerPhoenixDBIT(352): File "/Users/lbronshtein/DEV/phoenix/python/phoenixdb/avatica/cl ient.py", line 319, in open_connection 2018-09-07 12:51:00,350 ERROR [main] end2end.SecureQueryServerPhoenixDBIT(352): response_data = self._apply(request) 2018-09-07 12:51:00,350 ERROR [main] end2end.SecureQueryServerPhoenixDBIT(352): File "/Users/lbronshtein/DEV/phoenix/python/phoenixdb/avatica/cl ient.py", line 206, in _apply 2018-09-07 12:51:00,350 ERROR [main] end2end.SecureQueryServerPhoenixDBIT(352): parse_error_protobuf(response_body) 2018-09-07 12:51:00,350 ERROR [main] end2end.SecureQueryServerPhoenixDBIT(352): File "/Users/lbronshtein/DEV/phoenix/python/phoenixdb/avatica/cl ient.py", line 127, in parse_error_protobuf 2018-09-07 12:51:00,350 ERROR [main] end2end.SecureQueryServerPhoenixDBIT(352): raise errors.InternalError(err.error_message) 2018-09-07 12:51:00,350 ERROR [main] end2end.SecureQueryServerPhoenixDBIT(352): phoenixdb.errors.InternalError: (u'', None, None, None) 2018-09-07 12:51:00,350 ERROR [main] end2end.SecureQueryServerPhoenixDBIT(352): Exception phoenixdb.errors.InternalError: InternalError(u'', None, None, None) in <bound method Connection.__del__ of <phoenixdb.connection.Connection object at 0x10f2d52d0>> ignored {quote}
71. Also why the test is not failing even though non-zero exit should be checked {code:java} int exitCode = runPythonProcess.waitFor(); LOG.info("test_phoenixdb.sh exited with: " + exitCode); ... assertEquals("Subprocess exited with errors", 0, exitCode);{code} And is captured {quote}2018-09-07 12:51:00,336 INFO [main] end2end.SecureQueryServerPhoenixDBIT(345): test_phoenixdb.sh exited with: 1 {quote}
72. Github user joshelser commented on the issue: https://github.com/apache/phoenix/pull/344 This looks awesome, @pu239ppy. Testing it out.
73. Made a small change {quote}diff --git a/python/gen-protobuf.sh b/python/gen-protobuf.sh index f483561d6..4c5666c55 100755 --- a/python/gen-protobuf.sh +++ b/python/gen-protobuf.sh @@ -15,7 +15,8 @@ # See the License for the specific language governing permissions and # limitations under the License. -AVATICA_VER=rel/avatica-1.10.0 +set -x +AVATICA_VER=rel/avatica-1.12.0 set -e {quote} But still getting the error from above
74. **body:** I think I know what is happening, I am once again seeing this in the log {quote}2018-09-12 12:44:26,548 DEBUG [RedundancyMonitor] blockmanagement.BlockManager(1880): BLOCK* neededReconstruction = 0 pendingReconstruction = 0 2018-09-12 12:44:26,551 WARN [qtp1281394403-467] security.SpnegoLoginService(162): GSSException: No credential found for: 1.2.840.113554.1.2.2 usage: Accept at sun.security.jgss.GSSCredentialImpl.getElement(GSSCredentialImpl.java:600) at sun.security.jgss.GSSContextImpl.acceptSecContext(GSSContextImpl.java:317) at sun.security.jgss.GSSContextImpl.acceptSecContext(GSSContextImpl.java:285) at org.eclipse.jetty.security.SpnegoLoginService.login(SpnegoLoginService.java:138) at org.eclipse.jetty.security.authentication.LoginAuthenticator.login(LoginAuthenticator.java:61) at org.eclipse.jetty.security.authentication.SpnegoAuthenticator.validateRequest(SpnegoAuthenticator.java:99) at org.apache.calcite.avatica.server.AvaticaSpnegoAuthenticator.validateRequest(AvaticaSpnegoAuthenticator.java:43)

at org.eclipse.jetty.security.SecurityHandler.handle(SecurityHandler.java:483) at org.eclipse.jetty.server.handler.HandlerList.handle(HandlerList.java:52) at org.eclipse.jetty.server.handler.HandlerWrapper.handle(HandlerWrapper.java:134) {quote} And since for some reason an an exit with code of 1 does not cause an exception, the first time I tried to use new Avatica server, I amproperly assumed that the Kerberos mech vs SPNEGO mech issue has been solved (it is possible there was some other evidence to back this up, will have to review this very long thread). But as of now it seems we may be back to square one.
label: code-design

75. Judging by this stack-trace fragment {quote}911 at sun.security.jgss.GSSCredentialImpl.getElement(GSSCredentialImpl.java:600) 912 at sun.security.jgss.GSSContextImpl.acceptSecContext(GSSContextImpl.java:317) 913 at sun.security.jgss.GSSContextImpl.acceptSecContext(GSSContextImpl.java:285) 914 at org.eclipse.jetty.security.SpnegoLoginService.login(SpnegoLoginService.java:138) 915 at org.eclipse.jetty.security.authentication.LoginAuthenticator.login(LoginAuthenticator.java:61) 916 at org.eclipse.jetty.security.authentication.SpnegoAuthenticator.validateRequest(SpnegoAuthenticator.java:99) 917 at org.apache.calcite.avatica.server.AvaticaSpnegoAuthenticator.validateRequest(AvaticaSpnegoAuthenticator.java:43) {quote} It looks like we are still calling the original jetty code path and wind up executing *org.eclipse.jetty.security.SpnegoLoginService.login* instead of *org.apache.calcite.avatica.server.PropertyBasedSpnegoLoginService.java*
76. Hrm. That assessment looks right to me. I think you already came to it (judging by deleting the Jira comment) but this should be in the latest Avatica release. Promise I haven't forgotten about this one :)
77. There is a long thread on this http://jetty.4.x6.nabble.com/Jetty-custom-LoginService-question-td4966621.html
78. Not sure if that thread is relevant. We're programmatically launching the Jetty server and setting our custom LoginProviderService, not relying on that XML-configuration to instantiate jetty for us.
79. {quote}Hrm. That assessment looks right to me. I think you already came to it (judging by deleting the Jira comment) but this should be in the latest Avatica release. {quote} This was originally questioned since your PR was closed un merged so I was not sure how the change made it in.
80. {quote}This was originally questioned since your PR was closed un merged so I was not sure how the change made it in. {quote} Yeah, sorry. Github is just being misleading. I think I probably cherry-pick/rebased and didn't re-push to my fork. Thus, GH just sees the PR closed.
81. Auth is set up here [https://github.com/apache/calcite-avatica/blob/master/server/src/main/java/org/apache/calcite/avatica/server/HttpServer.java#L349] {code:java} return configureCommonAuthentication(Constraint.__SPNEGO_AUTH, allowedRealms, new AvaticaSpnegoAuthenticator(), realm, spnegoLoginService);{code} Seeing AvaticaSpnegoAuthenticatio in the stacktrace, however the spnegoLoginService may not have been set up
82. Went back to my old branch and reran looks like things did work at some point {quote}22158 2018-09-13 05:56:37,245 INFO [main] end2end.SecureQueryServerPhoenixDBIT(348): CREATING PQS CONNECTION 22159 2018-09-13 05:56:37,245 INFO [main] end2end.SecureQueryServerPhoenixDBIT(348): RESULTS 22160 2018-09-13 05:56:37,245 INFO [main] end2end.SecureQueryServerPhoenixDBIT(348): [[1, u'admin'], [2, u'user']] {quote} This test was done with avatica 1.11.0, going back to 1.10 again produces the familiar error {quote}2018-09-13 06:01:03,277 WARN [qtp918958156-413 - /] security.SpnegoLoginService(161): 289 GSSException: No credential found for: 1.2.840.113554.1.2.2 usage: Accept 290 at sun.security.jgss.GSSCredentialImpl.getElement(GSSCredentialImpl.java:600) {quote} Previously the jetty version was <jettyVersion>8.1.7.v20120910</jettyVersion> with <version>4.14.0-HBase-1.4</version> Currently it is <jetty.version>9.3.19.v20170502</jetty.version> with <version>5.1.0-HBase-2.0-SNAPSHOT</version>
83. Perhaps the old way of configuring SPNEGO no longer works {quote}at org.eclipse.jetty.security.authentication.LoginAuthenticator.login(LoginAuthenticator.java:61) {quote} {code:java} _loginService.login(username,password);{code} But for some reason the call is dispatched to org.eclipse.jetty.security.SpnegoLoginService.login(SpnegoLoginService.java:138) instead of org.apache.calcite.avatica.server.{color:#FF0000}[PropertyBasedSpnegoLoginService|https://github.com/apache/calcite-avatica/pull/15/files?utf8=%E2%9C%93&diff=split#diff-dd280db9e5949206ab19d453d55e4f59]{color}
84. Github user zhouwei0914 commented on the issue: https://github.com/apache/phoenix/pull/307 Hi pu239ppy, I have some questions, can you help me. Questions are as followsï¼š 1ã€In the fourth step, pip install phoenix/python/phoenixdv-module does not mean to install phoenixdb? I run: pip install phoenixdb 2ã€In the last setp, Profit: What is the operation to perform? 3ã€I have executed kinit in the virtual environment, but the code in the Example script is reported incorrectly. The detailed error message is as follows: HTTP ERROR: 401 Problem accessing /. Reason: Unauthorized Am I wrong, please guide me, thank you very much.
85. Github user pu239ppy commented on the issue: https://github.com/apache/phoenix/pull/307 @zhouwei0914 This PR is no longer active, please see #344 However here is a quick recap of the issue: It has been relatively easy to rewrite python-phoenixdb to use requests, however we found out that during authentication requests-kerberos

sends a kerberos OID rather then a SPNEGO OID for mechanism. There were two workarounds 1. Patch requests-kerberos (was never merged requests-kerberos#115) 2. A new SPNEGO handler for Avatica that would override Jetty's default handler (CALCITE-1922) It turned out that 2 Produced the desired effect that I abandoned the path of attempting to path requests-kerberos. However in the current PR #344 on top of Phoenix 5 it appears that this strategy no longer works. See additional details in later comments in PHOENIX-4688

86. Github user akhmadMizkat commented on the issue: https://github.com/apache/phoenix/pull/344 > This looks awesome, @pu239ppy. Testing it out. Hi @joshelser , did you finally managed to test it out successfully?
87. Github user pu239ppy commented on the issue: https://github.com/apache/phoenix/pull/344 @akhmadMizkat this solution currently does not work, please see latest commentary on PHOENIX-4688 for description. There are existing solutions to remediate this, however I was hoping not to distibute a patched version of requests-kerberos
88. Github user akhmadMizkat commented on the issue: https://github.com/apache/phoenix/pull/344 @pu239ppy thank you for the clarification. Hope they can accept your PR on the requests-kerberos. Really looking forward for that.
89. Until we figure out why CALCITE-1922 stopped working I have re-added modified requests-kerberos. Test is passing again
90. This is once again runnable, make sure you have kerberos utils (kinit, etc) and virtual env for the test to succeed
91. Github user joshelser commented on the issue: https://github.com/apache/phoenix/pull/344 > Hi @joshelser , did you finally managed to test it out successfully? No, sadly. I didn't get the last variant working. Pulling down this version and trying again.
92. Github user joshelser commented on the issue: https://github.com/apache/phoenix/pull/344 ``` 2018-09-27 14:05:06,308 ERROR [main] end2end.SecureQueryServerPhoenixDBIT(358): + python /Users/jelser/projects/phoenix.git/phoenix-queryserver/./src/it/bin/test_phoenixdb.py 50023 2018-09-27 14:05:06,309 ERROR [main] end2end.SecureQueryServerPhoenixDBIT(358): /Users/jelser/projects/phoenix.git/phoenix-queryserver/./src/it/bin/test_phoenixdb.sh: line 60: 7582 Illegal instruction: 4 python $PYTHON_SCRIPT $PQS_PORT ``` We're back to this junk again. @pu239ppy what version of Python are you running for which this is working?
93. Github user joshelser commented on the issue: https://github.com/apache/phoenix/pull/344 Switched to Python 3.6.5 and now hitting https://github.com/02strich/pykerberos/issues/37
94. Github user joshelser commented on the issue: https://github.com/apache/phoenix/pull/344 Ok, with python 3.6.5, downgrading to pykerberos to 1.1.14 (in requests-kerberos), and updating test_phoenixdb.py for python3 print syntax, it works!!!
95. Github user joshelser commented on the issue: https://github.com/apache/phoenix/pull/344 Oops, forgot one more thing. I changed the integration test that had heimdal syntax because, even though I have OSX, I have MIT Kerberos installed. Although, now that I think about it, do we need to generate a specific krb5.conf based on OS at all? The embedded KDC is provided by Kerby from Apache Directory -- it's not Heimdal or MIT Kerberos. That can be cleaned up completely. @pu239ppy let me know. I can send a pull request to your branch if you're ready to just be done with these changes. I want to poke around some more at Python versions that work locally.
96. [~elserj] the Heimdal parts if for MAC OS X users. On Mac OS X standard kerberos environment is provided by Heimdal depending on version of Mac OS X and the Heimdal utilities the krb5.conf file needs a small modification. Specifically it wants kdc = tcp/localhost:port to convince it to use TCP. As the miniKDC apperently listens on TCP only. MIT utilities will try both TCP and UDP where as older version of Heimdal need to be convinced to do so I suppose for Mac OS X we can generate two KDC lines one   kdc = tcp/localhost:port and one kdc = localhost:port Do you want to give it a shot and let me know if that works? I will do the same on my Mac Also [~elserj] I have no idea how and why you got MIT to work on a Mac, but we can try to run the test on Linux as well. We did float an idea of a docker test before but there are quite a few moving parts.
97. Tests worked after adding standard KDC notation I even placed it before the Heimdal one {code:java} if (osName.indexOf("mac") >= 0) { int kdcPort = KDC.getPort(); LOG.info("MINIKDC PORT " + kdcPort); // Render a Heimdal compatible krb5.conf // Currently kinit will only try tcp if the KDC is defined as // kdc = tcp/hostname:port StringBuilder krb5conf = new StringBuilder(); krb5conf.append("[libdefaults]\n"); krb5conf.append(" default_realm = EXAMPLE.COM\n"); krb5conf.append(" udp_preference_limit = 1\n"); krb5conf.append("\n"); krb5conf.append("[realms]\n"); krb5conf.append(" EXAMPLE.COM = {\n"); krb5conf.append(" kdc = localhost:"); krb5conf.append(kdcPort); krb5conf.append("\n"); krb5conf.append(" kdc = tcp/localhost:"); krb5conf.append(kdcPort); krb5conf.append("\n"); krb5conf.append(" }\n"); }{code}
98. **body:** So on the plus side we once again have a working module on top of the 5.x train. On the minus side it seem we once again lost the benefit of CALCITE-1922 and need to hack up requests-kerberos. I am not going to bother reopening requests-kerberos#115 as no one will merge it anyway.
    label: code-design
99. Github user akhmadMizkat commented on the issue: https://github.com/apache/phoenix/pull/344 Awesome guys. This is a very good improvement for the phoenixdb library.
100. {quote}On Mac OS X standard kerberos environment is provided by Heimdal depending on version of Mac OS X and the Heimdal utilities the krb5.conf file needs a small modification. {quote} I forgot that this is shell'ing out to do the `kinit` when I wrote the suggestion last night. My bad. {quote}Also [~elserj] I have no idea how and why you got MIT to work on a Mac, but we can try to run the test on Linux as well. We did float an idea of a docker

test before but there are quite a few moving parts. {quote} I'm not sure if should be reading this with incredulity :). {{brew install krb5}}, and modification of PATH to put those first is all that needs to be done. My point is that assuming that all users of OSX are using Heimdal is wrong. However, I don't know of a way to differentiate between MIT and Heimdal at a glance. That said... {quote}Tests worked after adding standard KDC notation I even placed it before the Heimdal one {quote} Any reason we have to generate two different krb5.conf at all? If we can generate a single krb5.conf that works for Heimdal and MIT, we don't need to have this indirection, right? {quote}On the minus side it seem we once again lost the benefit of CALCITE-1922 and need to hack up requests-kerberos. {quote} And, to be clear, the alternative is to use python-gssapi instead of the hacked requests-kerberos? Trying to make sure I understand everything so I can merge this today and write some documentation.

101. **body:** {quote}'m not sure if should be reading this with incredulity :). {{brew install krb5}}, and modification of PATH to put those first is all that needs to be done. My point is that assuming that all users of OSX are using Heimdal is wrong. However, I don't know of a way to differentiate between MIT and Heimdal at a glance. That said... {quote} I am not a very advanced Mac user, I find changing a lot of default behavior painful, but that could be my experience {quote}Any reason we have to generate two different krb5.conf at all? If we can generate a single krb5.conf that works for Heimdal and MIT, we don't need to have this indirection, right? {quote} We would have to teach Kirby (?) how to do that I suspect. I am not sure if this is that important for the scope of the test especially given that this is not an issue in future version of Heimdal {quote}And, to be clear, the alternative is to use python-gssapi instead of the hacked requests-kerberos? {quote} Sadly no. The reason I was pointed to python-gssapi is that it seem that the community is trying to deprecate python-kerberos and requests-kerberos (seems as it ios relagted to maintenance). However requests-gssapi does not support SPNEGO (i.e. it does not send the SPNEGO Mec OID)   See [https://github.com/requests/requests-kerberos/issues/116] for discussion, I since closed the issue and the associated PR as I in the 4.x branch CALCITE-1922 seemed to be working, but now it once again appears to be broken as evidenced by this stack trace {quote}911 at sun.security.jgss.GSSCredentialImpl.getElement(GSSCredentialImpl.java:600) 912 at sun.security.jgss.GSSContextImpl.acceptSecContext(GSSContextImpl.java:317) 913 at sun.security.jgss.GSSContextImpl.acceptSecContext(GSSContextImpl.java:285) *914 at org.eclipse.jetty.security.SpnegoLoginService.login(SpnegoLoginService.java:138) <-- This should call org.apache.calcite.avatica.server.[PropertyBasedSpnegoLoginService.java|https://github.com/apache/calcite-avatica/pull/15/files?utf8=%E2%9C%93&diff=split#diff-dd280db9e5949206ab19d453d55e4f59]* 915 at org.eclipse.jetty.security.authentication.LoginAuthenticator.login(LoginAuthenticator.java:61) 916 at org.eclipse.jetty.security.authentication.SpnegoAuthenticator.validateRequest(SpnegoAuthenticator.java:99) 917 at org.apache.calcite.avatica.server.AvaticaSpnegoAuthenticator.validateRequest(AvaticaSpnegoAuthenticator.java:43) {quote} It might be an error on my part as well. However if this worked we could easily have transitioned to requests-gssapi.   So the options are # Get to the bottom of CALCITE-1922 not working issue â€“ I spent quite a bit of time there and it seems like it should just work # Add SPNEGO support to requests-gsspi or python-gssapi [https://github.com/pythongssapi/python-gssapi/issues/50] no work has been done so far # Hack up requests-kerberos in [https://github.com/requests/requests-kerberos/issues/116] I specified what the problem , was bus as CALCITE-1922 was working I closed it saying that this is die to a pedantic Jetty configuration and is no longer an issue, however I could reopen, but do not have much hope for a resolution or at least one that will happen soon.
label: code-design
102. I went back to the requests-gssapi version of the code to see if I can figure out why that isn't working. The python library is stuck trying to talk to my KDC {code:java} 2018-09-28T16:44:21 submissing new requests to new host 2018-09-28T16:44:21 host_create: setting hostname hw13390.local 2018-09-28T16:44:21 connecting to host: udp fe80::810:c412:5688:3abf%en0:kerberos (hw13390.local) tid: 00000001 2018-09-28T16:44:21 host_create: setting hostname hw13390.local 2018-09-28T16:44:21 Queuing host in future (in 3s), its the 2 address on the same name: udp fe80::810:c412:5688:3abf%en0:kerberos (hw13390.local) tid: 00000002 2018-09-28T16:44:21 host_create: setting hostname hw13390.local 2018-09-28T16:44:21 Queuing host in future (in 6s), its the 3 address on the same name: udp 192.168.1.32:kerberos (hw13390.local) tid: 00000003 2018-09-28T16:44:21 host_create: setting hostname hw13390.local 2018-09-28T16:44:21 Queuing host in future (in 9s), its the 4 address on the same name: udp 192.168.1.32:kerberos (hw13390.local) tid: 00000004 2018-09-28T16:44:21 writing packet: udp fe80::810:c412:5688:3abf%en0:kerberos (hw13390.local) tid: 00000001 2018-09-28T16:44:22 Configuration exists for realm EXAMPLE.COM, wont go to DNS 2018-09-28T16:44:22 out of hosts, waiting for replies{code} I can't seem to get any more logging/information out of Python or the KDC to understand why these requests are going into the abyss. I'm not sure if my KDC is dropping UDP packets on the ground, nor can I figure out how to make Python send via TCP. I'm of the opinion that we ship this with clear documentation on what exactly is know to work. What I've been able to validate is: * MIT Kerberos * Python 3.6 * requests-kerberos * Phoenix 5.x * pykerberos 1.1.14 I want to test some more Python versions (e.g. 3.4 and 3.5) and try with Heimdal `kinit` on the path, but after that I'd like to just commit something and move forward. Thoughts [~lbronshtein]?
103. [~elserj] Using requests-gssapi or unhacked requests kerberos should cause you to see the following {quote}2018-09-12 12:44:26,548 DEBUG [RedundancyMonitor] blockmanagement.BlockManager(1880): BLOCK* neededReconstruction = 0 pendingReconstruction = 0 2018-09-12 12:44:26,551 WARN [qtp1281394403-467] security.SpnegoLoginService(162): *GSSException: No credential found for: 1.2.840.113554.1.2.2 usage: Accept* at sun.security.jgss.GSSCredentialImpl.getElement(GSSCredentialImpl.java:600) {quote} This is what

CALCITE-1922 attempts to fix. This would show up in failsafe output log   The output you posted looks strange as MIT should try to fallback to TCP or that was my impression. From [https://web.mit.edu/kerberos/krb5-1.12/doc/admin/conf_files/krb5_conf.html] {quote}*udp_preference_limit*When sending a message to the KDC, the library will try using TCP before UDP if the size of the message is above *udp_preference_limit*. If the message is smaller than*udp_preference_limit*, then UDP will be tried before TCP. *Regardless of the size, both protocols will be tried if the first attempt fails.* {quote} Also TCP and UDP seem to be mutually exclusive for MiniKDC   [https://github.com/apache/hadoop/blob/a55d6bba71c81c1c4e9d8cd11f55c78f10a548b0/hadoop-common-project/hadoop-minikdc/src/main/java/org/apache/hadoop/minikdc/MiniKdc.java#L308] {code:java} if (transport.trim().equals("TCP")) { simpleKdc.setKdcTcpPort(port); simpleKdc.setAllowUdp(false); } else if (transport.trim().equals("UDP")) { simpleKdc.setKdcUdpPort(port); simpleKdc.setAllowTcp(false); }{code}

104. {quote}This is what CALCITE-1922 attempts to fix. This would show up in failsafe output log {quote} I wasn't (and am not, again) getting that far. Something seems to change for me, and I haven't figured out exactly what yet. {quote}The output you posted looks strange as MIT should try to fallback to TCP or that was my impression {quote} Yeah, I have no idea. This doesn't appear to be how it's supposed to work. {quote}Also TCP and UDP seem to be mutually exclusive for MiniKDC {quote} Grumble. Ok, that's a good data point. Thanks.
105. {quote}I wasn't (and am not, again) getting that far. Something seems to change for me, and I haven't figured out exactly what yet. {quote} Nevermind. Forgot I was on your requests-gssapi branch. That's why this isn't working :)
106. **body:** Some follow-on things to improve: * Better detection when an unsupported Python version is on the PATH and when virtualenv is missing.
     label: code-design
107. [~elserj] * WRT to missing virtual env test: Are you looking to just abort the test early and report a failure? Because otherwise the test will fail when virtualenv has failed to run (set -e ensures that) and it will be reported in the failsafe output file.   Also if you want a more explicit error message I can add this {code:java} virtualenv $PY_ENV_PATH || echo "Failure while running virtualenv ${?}" && exit 1{code} * Python Version: I can detect if this is a 2.x or a 3.x and change the print statement depending on python version. In fact I can just import print from future and only make that part optional. Is this what you are looking for?
108. {quote} * WRT to missing virtual env test: Are you looking to just abort the test early and report a failure? Because otherwise the test will fail when virtualenv has failed to run (set -e ensures that) and it will be reported in the failsafe output file.{quote} Yes, it does fail now. Just thinking out loud that we could make it extremely clear without forcing the developer to go into the test log to find that. Would probably require a check in the Java code to get a clear message back out of Maven (essentially what I was thinking for the Python version too). To be clear, both of these can be "later" items.
109. [~elserj] I am not sure how we would do that as STDOUT and STDERR get swallowed up by the log
110. [~lbronshtein], did you test successfully with Heimdal on OSX? It does not work for me.
111. [~elserj] Yes I have, what are you seeing?
112. **body:** {code:java} + python /Users/jelser/projects/phoenix.git/phoenix-queryserver/./src/it/bin/test_phoenixdb.py 54673 DEBUG:phoenixdb.avatica.client:Sending request connection_id: "ff1fba48-68c4-42c7-af09-a9b598f74bac" DEBUG:phoenixdb.avatica.client:POST http://localhost:54673/ b'\n?org.apache.calcite.avatica.proto.Requests$OpenConnectionRequest\x12&\n$ff1fba48-68c4-42c7-af09-a9b598f74bac' {'content-type': 'application/x-google-protobuf'} DEBUG:urllib3.connectionpool:Starting new HTTP connection (1): localhost:54673 DEBUG:urllib3.connectionpool:http://localhost:54673 "POST / HTTP/1.1" 401 320 DEBUG:requests_kerberos.kerberos_:handle_401(): Handling: 401 ERROR:requests_kerberos.kerberos_:generate_request_header(): authGSSClientStep() failed: Traceback (most recent call last):   File "/Users/jelser/projects/phoenix.git/python/requests-kerberos/requests_kerberos/kerberos_.py", line 235, in generate_request_header   negotiate_resp_value) kerberos.GSSError: (('Unspecified GSS failure. Minor code may provide more information', 851968), ("Cannot contact any KDC for realm 'EXAMPLE.COM'", 100001)) ERROR:requests_kerberos.kerberos_:(('Unspecified GSS failure. Minor code may provide more information', 851968), ("Cannot contact any KDC for realm 'EXAMPLE.COM'", 100001)) Traceback (most recent call last):   File "/Users/jelser/projects/phoenix.git/python/requests-kerberos/requests_kerberos/kerberos_.py", line 235, in generate_request_header   negotiate_resp_value) kerberos.GSSError: (('Unspecified GSS failure. Minor code may provide more information', 851968), ("Cannot contact any KDC for realm 'EXAMPLE.COM'", 100001)) {code} This is what I've figured out so far. I have a couple more things to try before I throw in the towel. Just wanted to make sure that was something you had tested and were expecting to work :) (not that you were on a Linux box the whole time, haha)
     label: test
113. [~elserj] If I am reading this correctly it is requests kerberos telling you that it cannot obtain a service ticket for HTTP/pqs-host. Did you modify the krb5.conf generation script so it would output MIT compatible KDC definition as well as Heimdal tcp/host? also in your output can you see if kinit actually worked? You could also try removing my if statement completely as you are on MIT anyway
114. {quote}Did you modify the krb5.conf generation script so it would output MIT compatible KDC definition as well as Heimdal tcp/host? also in your output can you see if kinit actually worked? {quote} I should have been running verbatim with what is at the tip of your pull-request branch. I believe the kinit worked fine. I removed my PATH

modifications so that it would invoke the Heimdal executables; there shouldn't have been any MIT krb5 remnants on the path at all, but I'll have to investigate that some more. Still shooting to finish up this testing and push what we have now.

115. {noformat} + python /Users/jelser/projects/phoenix.git/phoenix-queryserver/./src/it/bin/test_phoenixdb.py 51475 CREATING PQS CONNECTION DEBUG:phoenixdb.avatica.client:Sending request connection_id: "8a638e07-34cb-4d05-acfa-0dc9970e2522" DEBUG:phoenixdb.avatica.client:POST http://localhost:51475/ b'\n?org.apache.calcite.avatica.proto.Requests$OpenConnectionRequest\x12&\n$8a638e07-34cb-4d05-acfa-0dc9970e2522' {'content-type': 'application/x-google-protobuf'} DEBUG:urllib3.connectionpool:Starting new HTTP connection (1): localhost:51475 DEBUG:urllib3.connectionpool:http://localhost:51475 "POST / HTTP/1.1" 401 320 DEBUG:requests_kerberos.kerberos_:handle_401(): Handling: 401 [4684] 1538514004.487266: ccselect module realm chose cache FILE:/tmp/krb5cc_502 with client principal user1@EXAMPLE.COM for server principal HTTP/localhost@EXAMPLE.COM [4684] 1538514004.487267: Getting credentials user1@EXAMPLE.COM -> HTTP/localhost@ using ccache FILE:/tmp/krb5cc_502 [4684] 1538514004.487268: Retrieving user1@EXAMPLE.COM -> HTTP/localhost@ from FILE:/tmp/krb5cc_502 with result: -1765328243/Matching credential not found (filename: /tmp/krb5cc_502) [4684] 1538514004.487269: Retrying user1@EXAMPLE.COM -> HTTP/localhost@EXAMPLE.COM with result: -1765328243/Matching credential not found (filename: /tmp/krb5cc_502) [4684] 1538514004.487270: Server has referral realm; starting with HTTP/localhost@EXAMPLE.COM [4684] 1538514004.487271: Retrieving user1@EXAMPLE.COM -> krbtgt/EXAMPLE.COM@EXAMPLE.COM from FILE:/tmp/krb5cc_502 with result: 0/Success [4684] 1538514004.487272: Starting with TGT for client realm: user1@EXAMPLE.COM -> krbtgt/EXAMPLE.COM@EXAMPLE.COM [4684] 1538514004.487273: Requesting tickets for HTTP/localhost@EXAMPLE.COM, referrals on [4684] 1538514004.487274: Generated subkey for TGS request: aes128-cts/A291 [4684] 1538514004.487275: etypes requested in TGS request: aes256-cts, aes128-cts, aes256-sha2, aes128-sha2, des3-cbc-sha1, rc4-hmac, camellia128-cts, camellia256-cts [4684] 1538514004.487277: Encoding request body and padata into FAST request [4684] 1538514004.487278: Sending request (812 bytes) to EXAMPLE.COM [4684] 1538514004.487279: Resolving hostname tcp/localhost ERROR:requests_kerberos.kerberos_:generate_request_header(): authGSSClientStep() failed: Traceback (most recent call last):   File "/Users/jelser/projects/phoenix.git/python/requests-kerberos/requests_kerberos/kerberos_.py", line 235, in generate_request_header   negotiate_resp_value) kerberos.GSSError: (('Unspecified GSS failure. Minor code may provide more information', 851968), ("Cannot contact any KDC for realm 'EXAMPLE.COM'", 100001)) ERROR:requests_kerberos.kerberos_:(('Unspecified GSS failure. Minor code may provide more information', 851968), ("Cannot contact any KDC for realm 'EXAMPLE.COM'", 100001)) Traceback (most recent call last):   File "/Users/jelser/projects/phoenix.git/python/requests-kerberos/requests_kerberos/kerberos_.py", line 235, in generate_request_header   negotiate_resp_value) kerberos.GSSError: (('Unspecified GSS failure. Minor code may provide more information', 851968), ("Cannot contact any KDC for realm 'EXAMPLE.COM'", 100001)) {noformat} Re-running this on the command line by hand, we hang at the "Resolving hostname tcp/localhost" line (which feels like the a DNS timeout looking for a system with the hostname of "tcp/localhost" instead of just localhost). Makes me think that my Python library isn't correctly handling this syntax (but Heimdal's kinit is). No clue yet why I'm only seeing this..
116. kdc = tcp/localhost is what my code inserts if you are on OSX. Maybe disable that and see what your mileage is
117. {quote} kdc = tcp/localhost is what my code inserts if you are on OSX. Maybe disable that and see what your mileage is {quote} If I do this, then {{kinit}} fails, haha. Figures. Digging through requests-kerberos and pykerberos to see if I can get any more info as to what those are doing.
118. Ok, figured it out. Python is still (somehow) finding my MIT krb libraries. I get the same exact error if I use MIT kinit with the heimdal krb5.conf you generate. {noformat} % KRB5_TRACE=/dev/stdout KRB5_CONFIG=/var/folders/4q/q02ykc2j5l1fg8nbs_sczskh0000gp/T/krb5.conf4602740147062445533.tmp /usr/local/Cellar/krb5/1.16.1/bin/kinit -kt /Users/jelser/projects/phoenix.git/phoenix-queryserver/target/SecureQueryServerPhoenixDBIT/keytabs/user1.keytab user1 [5696] 1538518054.672090: Getting initial credentials for user1@EXAMPLE.COM [5696] 1538518054.672091: Looked up etypes in keytab: aes128-cts, des3-cbc-sha1 [5696] 1538518054.672093: Sending unauthenticated request [5696] 1538518054.672094: Sending request (171 bytes) to EXAMPLE.COM [5696] 1538518054.672095: Resolving hostname tcp/localhost{noformat} That makes sense now as to why you don't see this issue. I don't particularly think we should care about fixing it either :) Let me just add an option that users can set on the command-line to override what SecureQueryServerPhoenixDBIT will default to (heimdal or mit krb compatible krb5.conf).
119. I think for OSX I will just have Kdc = localhost:port Kdc = tcp/localhost:port Do you want to try that and see if it works for you? It worked for Heimdal even when Kdc = localhost:port is first
120. {quote}Do you want to try that and see if it works for you? It worked for Heimdal even when {quote} Curious. Yeah, that worked for me too. Any idea if that's intended to work or just circumstantial?
121. Well I guess if one kdc fails (for whatever reason) the other works. That's why you have multiple entries
122. Alright, I'm getting antsy now :). Itchy commit-finger. [~lbronshtein], I've taken the liberty of smashing in some other cleanup in with your change: * The IT will check to see if there is a python, virtualenv, and kinit on the PATH before running itself. It will gracefully "Ignore" itself when any of these are missing * I've added a

README to python/ to try to summarize some extra details (but not duplicate python/python-phoenixdb's README) * Updated requirements.txt * Updated L&N for bin and src releases * Ensure test instructions still worked for the python-based tests Going to push shortly. 4.x, as well as master, assuming this comes back gracefully to those branches.

123. Github user asfgit closed the pull request at: https://github.com/apache/phoenix/pull/344
124. Calling it! Pushed this to 4.15 and 5.1. It is more than good enough for a first stab at this. We can continue to iterate now that we have a starting point. Thanks for your Herculean efforts, Lev! Truly impressive.
125. FAILURE: Integrated in Jenkins build Phoenix-4.x-HBase-1.3 #226 (See [https://builds.apache.org/job/Phoenix-4.x-HBase-1.3/226/]) PHOENIX-4688 Support SPNEGO for python driver via requests-kerberos (elserj: rev e62be9c820aaf05266500cc509d4a89658cb6918) * (add) python/phoenixdb/phoenixdb/tests/test_errors.py * (add) python/phoenixdb/phoenixdb/avatica/client.py * (delete) python/phoenixdb/avatica/proto/requests_pb2.py * (add) python/phoenixdb/phoenixdb/avatica/proto/__init__.py * (add) python/phoenixdb/docker-compose.yml * (add) python/phoenixdb/README.rst * (add) python/phoenixdb/tox.ini * (add) python/phoenixdb/examples/basic.py * (delete) python/phoenixdb/connection.py * (delete) python/doc/conf.py * (add) python/requests-kerberos/.travis.yml * (delete) python/phoenixdb/tests/test_avatica.py * (add) python/phoenixdb/ci/build-env/Dockerfile * (delete) python/phoenixdb/tests/test_dbapi20.py * (delete) python/phoenixdb/avatica/proto/responses_pb2.py * (delete) python/phoenixdb/tests/test_types.py * (delete) python/examples/shell.py * (add) python/phoenixdb/RELEASING.rst * (add) python/requests-kerberos/requests_kerberos/exceptions.py * (add) python/requests-kerberos/requirements-test.txt * (delete) python/phoenixdb/avatica/proto/__init__.py * (delete) python/phoenixdb/types.py * (add) python/requests-kerberos/tests/test_functional_kerberos.py * (add) python/phoenixdb/phoenixdb/tests/__init__.py * (add) python/phoenixdb/phoenixdb/tests/test_dbapi20.py * (delete) python/NEWS.rst * (add) python/requests-kerberos/requests_kerberos/compat.py * (add) python/phoenixdb/ci/phoenix/hbase-site.xml * (delete) python/ci/phoenix/Dockerfile * (add) python/phoenixdb/phoenixdb/tests/test_db.py * (edit) LICENSE * (delete) python/phoenixdb/avatica/client.py * (delete) python/ci/build-env/Dockerfile * (add) python/phoenixdb/setup.py * (delete) python/ci/phoenix/hbase-site.xml * (delete) python/phoenixdb/tests/test_db.py * (add) python/requests-kerberos/LICENSE * (add) python/phoenixdb/doc/api.rst * (add) python/requests-kerberos/requirements.txt * (add) python/requests-kerberos/requests_kerberos/__init__.py * (add) python/phoenixdb/ci/phoenix/Dockerfile * (add) python/phoenixdb/phoenixdb/cursor.py * (delete) python/gen-protobuf.sh * (delete) python/phoenixdb/errors.py * (add) python/phoenixdb/phoenixdb/errors.py * (delete) python/doc/api.rst * (delete) python/RELEASING.rst * (add) python/README.md * (add) python/requests-kerberos/.travis.sh * (delete) python/phoenixdb/tests/test_errors.py * (add) python/phoenixdb/requirements.txt * (add) python/phoenixdb/ci/phoenix/docker-entrypoint.sh * (add) python/phoenixdb/phoenixdb/tests/test_connection.py * (delete) python/ci/phoenix/docker-entrypoint.sh * (delete) python/examples/basic.py * (delete) python/doc/Makefile * (delete) python/phoenixdb/avatica/__init__.py * (delete) python/doc/index.rst * (add) python/phoenixdb/phoenixdb/avatica/proto/common_pb2.py * (delete) python/requirements.txt * (add) python/requests-kerberos/AUTHORS * (add) python/requests-kerberos/README.rst * (add) python/phoenixdb/phoenixdb/tests/dbapi20.py * (add) python/phoenixdb/doc/versions.rst * (add) python/phoenixdb/phoenixdb/avatica/__init__.py * (edit) pom.xml * (add) python/requests-kerberos/setup.py * (add) python/phoenixdb/examples/shell.py * (delete) python/phoenixdb/avatica/proto/common_pb2.py * (add) python/phoenixdb/phoenixdb/avatica/proto/responses_pb2.py * (add) python/phoenixdb/phoenixdb/tests/test_avatica.py * (add) python/phoenixdb/doc/Makefile * (delete) python/phoenixdb/__init__.py * (delete) python/phoenixdb/cursor.py * (add) python/requests-kerberos/MANIFEST.in * (delete) python/tox.ini * (delete) python/doc/versions.rst * (delete) python/README.rst * (add) python/phoenixdb/phoenixdb/tests/test_types.py * (add) python/requests-kerberos/HISTORY.rst * (add) python/phoenixdb/gen-protobuf.sh * (add) python/phoenixdb/phoenixdb/types.py * (delete) python/setup.py * (delete) python/docker-compose.yml * (delete) python/phoenixdb/tests/test_connection.py * (edit) dev/release_files/LICENSE * (edit) dev/release_files/NOTICE * (add) python/phoenixdb/phoenixdb/__init__.py * (add) phoenix-queryserver/src/it/bin/test_phoenixdb.py * (add) python/requests-kerberos/requests_kerberos/kerberos_.py * (add) python/requests-kerberos/tests/test_requests_kerberos.py * (add) python/requests-kerberos/setup.cfg * (add) phoenix-queryserver/src/it/bin/test_phoenixdb.sh * (delete) python/phoenixdb/tests/dbapi20.py * (delete) python/setup.cfg * (add) python/phoenixdb/doc/conf.py * (add) python/requests-kerberos/tests/__init__.py * (add) python/phoenixdb/phoenixdb/avatica/proto/requests_pb2.py * (add) python/phoenixdb/phoenixdb/connection.py * (delete) python/phoenixdb/tests/__init__.py * (add) phoenix-queryserver/src/it/java/org/apache/phoenix/end2end/SecureQueryServerPhoenixDBIT.java * (add) python/phoenixdb/NEWS.rst * (add) python/phoenixdb/setup.cfg * (edit) NOTICE * (add) python/phoenixdb/doc/index.rst
126. FAILURE: Integrated in Jenkins build PreCommit-PHOENIX-Build #2078 (See [https://builds.apache.org/job/PreCommit-PHOENIX-Build/2078/]) PHOENIX-4688 Support SPNEGO for python driver via requests-kerberos (elserj: rev 14b578012d45ce1ae40e599159ac96d23083f024) * (add) python/requests-kerberos/LICENSE * (delete) python/phoenixdb/tests/test_dbapi20.py * (delete) python/phoenixdb/connection.py * (delete) python/phoenixdb/avatica/__init__.py * (add) python/requests-kerberos/requests_kerberos/__init__.py * (delete) python/doc/index.rst * (add) python/phoenixdb/phoenixdb/tests/test_errors.py * (delete) python/tox.ini *

(add) python/phoenixdb/phoenixdb/tests/test_db.py * (add) python/phoenixdb/examples/basic.py * (add) python/phoenixdb/doc/versions.rst * (add) python/phoenixdb/phoenixdb/errors.py * (add) python/phoenixdb/doc/api.rst * (add) python/requests-kerberos/.travis.yml * (add) python/phoenixdb/phoenixdb/tests/__init__.py * (add) python/phoenixdb/doc/index.rst * (add) python/requests-kerberos/requests_kerberos/compat.py * (delete) python/phoenixdb/tests/__init__.py * (edit) dev/release_files/LICENSE * (delete) python/phoenixdb/avatica/proto/common_pb2.py * (add) python/phoenixdb/ci/phoenix/docker-entrypoint.sh * (delete) python/phoenixdb/__init__.py * (add) python/phoenixdb/phoenixdb/tests/test_types.py * (add) python/phoenixdb/phoenixdb/avatica/__init__.py * (add) python/requests-kerberos/tests/test_requests_kerberos.py * (add) python/phoenixdb/setup.cfg * (delete) python/NEWS.rst * (add) python/README.md * (add) python/phoenixdb/phoenixdb/avatica/proto/responses_pb2.py * (add) python/phoenixdb/doc/conf.py * (add) python/phoenixdb/tox.ini * (delete) python/ci/phoenix/hbase-site.xml * (delete) python/README.rst * (delete) python/ci/build-env/Dockerfile * (delete) python/doc/conf.py * (add) python/requests-kerberos/requests_kerberos/kerberos_.py * (add) python/phoenixdb/phoenixdb/tests/test_dbapi20.py * (add) python/phoenixdb/ci/phoenix/hbase-site.xml * (add) python/phoenixdb/ci/build-env/Dockerfile * (add) python/phoenixdb/phoenixdb/__init__.py * (add) python/phoenixdb/phoenixdb/cursor.py * (delete) python/doc/api.rst * (delete) python/setup.cfg * (delete) python/RELEASING.rst * (delete) python/phoenixdb/errors.py * (edit) NOTICE * (add) python/requests-kerberos/tests/test_functional_kerberos.py * (add) python/requests-kerberos/setup.cfg * (add) python/requests-kerberos/requirements-test.txt * (add) python/phoenixdb/gen-protobuf.sh * (add) python/requests-kerberos/tests/__init__.py * (delete) python/phoenixdb/avatica/proto/responses_pb2.py * (delete) python/phoenixdb/tests/test_types.py * (add) python/phoenixdb/requirements.txt * (delete) python/examples/basic.py * (add) python/phoenixdb/examples/shell.py * (delete) python/phoenixdb/cursor.py * (add) python/phoenixdb/phoenixdb/avatica/proto/common_pb2.py * (delete) python/phoenixdb/tests/test_db.py * (delete) python/phoenixdb/avatica/proto/__init__.py * (edit) LICENSE * (delete) python/phoenixdb/tests/test_connection.py * (delete) python/doc/Makefile * (add) python/phoenixdb/ci/phoenix/Dockerfile * (add) python/requests-kerberos/requests_kerberos/exceptions.py * (delete) python/ci/phoenix/Dockerfile * (add) python/phoenixdb/phoenixdb/tests/test_avatica.py * (add) python/phoenixdb/docker-compose.yml * (add) phoenix-queryserver/src/it/bin/test_phoenixdb.py * (add) python/requests-kerberos/AUTHORS * (add) python/phoenixdb/phoenixdb/types.py * (edit) dev/release_files/NOTICE * (delete) python/phoenixdb/tests/test_errors.py * (add) python/requests-kerberos/requirements.txt * (delete) python/gen-protobuf.sh * (add) python/requests-kerberos/README.rst * (add) python/phoenixdb/README.rst * (delete) python/ci/phoenix/docker-entrypoint.sh * (add) phoenix-queryserver/src/it/bin/test_phoenixdb.sh * (add) python/phoenixdb/phoenixdb/tests/test_connection.py * (add) python/phoenixdb/phoenixdb/connection.py * (delete) python/docker-compose.yml * (add) phoenix-queryserver/src/it/java/org/apache/phoenix/end2end/SecureQueryServerPhoenixDBIT.java * (add) python/requests-kerberos/setup.py * (delete) python/phoenixdb/avatica/client.py * (delete) python/requirements.txt * (add) python/phoenixdb/setup.py * (delete) python/doc/versions.rst * (add) python/phoenixdb/phoenixdb/avatica/client.py * (delete) python/setup.py * (add) python/requests-kerberos/.travis.sh * (add) python/phoenixdb/phoenixdb/tests/dbapi20.py * (edit) pom.xml * (add) python/phoenixdb/NEWS.rst * (add) python/requests-kerberos/MANIFEST.in * (add) python/phoenixdb/doc/Makefile * (delete) python/examples/shell.py * (add) python/requests-kerberos/HISTORY.rst * (delete) python/phoenixdb/tests/test_avatica.py * (delete) python/phoenixdb/types.py * (add) python/phoenixdb/phoenixdb/avatica/proto/__init__.py * (delete) python/phoenixdb/tests/dbapi20.py * (add) python/phoenixdb/RELEASING.rst * (add) python/phoenixdb/phoenixdb/avatica/proto/requests_pb2.py * (delete) python/phoenixdb/avatica/proto/requests_pb2.py

127. FAILURE: Integrated in Jenkins build Phoenix-omid2 #124 (See [https://builds.apache.org/job/Phoenix-omid2/124/]) PHOENIX-4688 Support SPNEGO for python driver via requests-kerberos (elserj: rev e62be9c820aaf05266500cc509d4a89658cb6918) * (delete) python/ci/phoenix/docker-entrypoint.sh * (add) python/phoenixdb/gen-protobuf.sh * (add) python/phoenixdb/phoenixdb/avatica/__init__.py * (add) python/requests-kerberos/.travis.sh * (add) python/phoenixdb/phoenixdb/tests/dbapi20.py * (add) python/phoenixdb/phoenixdb/avatica/proto/responses_pb2.py * (add) python/phoenixdb/setup.cfg * (add) python/phoenixdb/phoenixdb/avatica/client.py * (delete) python/docker-compose.yml * (add) python/phoenixdb/ci/phoenix/docker-entrypoint.sh * (add) phoenix-queryserver/src/it/bin/test_phoenixdb.py * (delete) python/phoenixdb/avatica/proto/common_pb2.py * (add) python/phoenixdb/doc/api.rst * (delete) python/phoenixdb/avatica/__init__.py * (add) python/phoenixdb/ci/phoenix/hbase-site.xml * (delete) python/phoenixdb/tests/test_connection.py * (add) python/requests-kerberos/MANIFEST.in * (add) python/requests-kerberos/setup.cfg * (delete) python/ci/phoenix/Dockerfile * (add) python/phoenixdb/phoenixdb/avatica/proto/requests_pb2.py * (add) python/phoenixdb/phoenixdb/tests/test_connection.py * (delete) python/phoenixdb/tests/dbapi20.py * (add) python/requests-kerberos/README.rst * (delete) python/gen-protobuf.sh * (add) python/requests-kerberos/tests/__init__.py * (add) python/phoenixdb/doc/index.rst * (delete) python/tox.ini * (add) python/phoenixdb/phoenixdb/tests/test_db.py * (delete) python/phoenixdb/avatica/client.py * (add) phoenix-

queryserver/src/it/java/org/apache/phoenix/end2end/SecureQueryServerPhoenixDBIT.java * (add) python/phoenixdb/ci/phoenix/Dockerfile * (add) python/requests-kerberos/LICENSE * (add) python/requests-kerberos/requests_kerberos/kerberos_.py * (delete) python/phoenixdb/avatica/proto/requests_pb2.py * (add) python/phoenixdb/requirements.txt * (add) python/phoenixdb/phoenixdb/connection.py * (edit) dev/release_files/LICENSE * (add) python/phoenixdb/phoenixdb/types.py * (delete) python/examples/shell.py * (delete) python/phoenixdb/errors.py * (edit) LICENSE * (delete) python/phoenixdb/tests/test_dbapi20.py * (delete) python/phoenixdb/tests/test_errors.py * (add) python/phoenixdb/README.rst * (add) python/phoenixdb/phoenixdb/avatica/proto/__init__.py * (add) python/requests-kerberos/requests_kerberos/__init__.py * (edit) dev/release_files/NOTICE * (delete) python/doc/Makefile * (add) python/requests-kerberos/requirements-test.txt * (delete) python/phoenixdb/tests/__init__.py * (add) python/requests-kerberos/requests_kerberos/compat.py * (add) python/phoenixdb/doc/Makefile * (add) python/phoenixdb/phoenixdb/tests/test_dbapi20.py * (add) python/requests-kerberos/tests/test_requests_kerberos.py * (add) python/phoenixdb/phoenixdb/errors.py * (delete) python/ci/build-env/Dockerfile * (add) python/phoenixdb/phoenixdb/tests/__init__.py * (add) python/phoenixdb/ci/build-env/Dockerfile * (delete) python/setup.cfg * (delete) python/phoenixdb/types.py * (delete) python/phoenixdb/connection.py * (delete) python/phoenixdb/tests/test_db.py * (add) python/phoenixdb/setup.py * (delete) python/phoenixdb/tests/test_avatica.py * (edit) pom.xml * (add) python/requests-kerberos/.travis.yml * (delete) python/phoenixdb/avatica/proto/responses_pb2.py * (add) python/phoenixdb/tox.ini * (add) python/phoenixdb/phoenixdb/tests/test_avatica.py * (delete) python/phoenixdb/avatica/proto/__init__.py * (add) python/phoenixdb/phoenixdb/tests/test_errors.py * (add) python/requests-kerberos/HISTORY.rst * (delete) python/doc/api.rst * (delete) python/RELEASING.rst * (delete) python/NEWS.rst * (add) python/phoenixdb/RELEASING.rst * (add) python/README.md * (delete) python/phoenixdb/cursor.py * (delete) python/phoenixdb/tests/test_types.py * (add) python/phoenixdb/doc/versions.rst * (delete) python/examples/basic.py * (delete) python/README.rst * (delete) python/doc/index.rst * (add) phoenix-queryserver/src/it/bin/test_phoenixdb.sh * (delete) python/doc/conf.py * (add) python/requests-kerberos/AUTHORS * (add) python/phoenixdb/examples/shell.py * (add) python/requests-kerberos/requirements.txt * (delete) python/phoenixdb/__init__.py * (add) python/requests-kerberos/tests/test_functional_kerberos.py * (delete) python/setup.py * (edit) NOTICE * (delete) python/ci/phoenix/hbase-site.xml * (add) python/phoenixdb/phoenixdb/tests/test_types.py * (add) python/phoenixdb/phoenixdb/cursor.py * (delete) python/doc/versions.rst * (delete) python/requirements.txt * (add) python/phoenixdb/examples/basic.py * (add) python/phoenixdb/NEWS.rst * (add) python/phoenixdb/doc/conf.py * (add) python/requests-kerberos/setup.py * (add) python/phoenixdb/docker-compose.yml * (add) python/phoenixdb/phoenixdb/__init__.py * (add) python/phoenixdb/phoenixdb/avatica/proto/common_pb2.py * (add) python/requests-kerberos/requests_kerberos/exceptions.py

128. Github user Reidddddd commented on the issue: https://github.com/apache/phoenix/pull/344 Python version has to >= 3.6.5? @joshelser @pu239ppy
129. Github user joshelser commented on the issue: https://github.com/apache/phoenix/pull/344 Hey @Reidddddd -- likely other version of Python will work too, we just only tested using 3.6.5
130. Bulk closing Jiras for the 4.15.0 release.