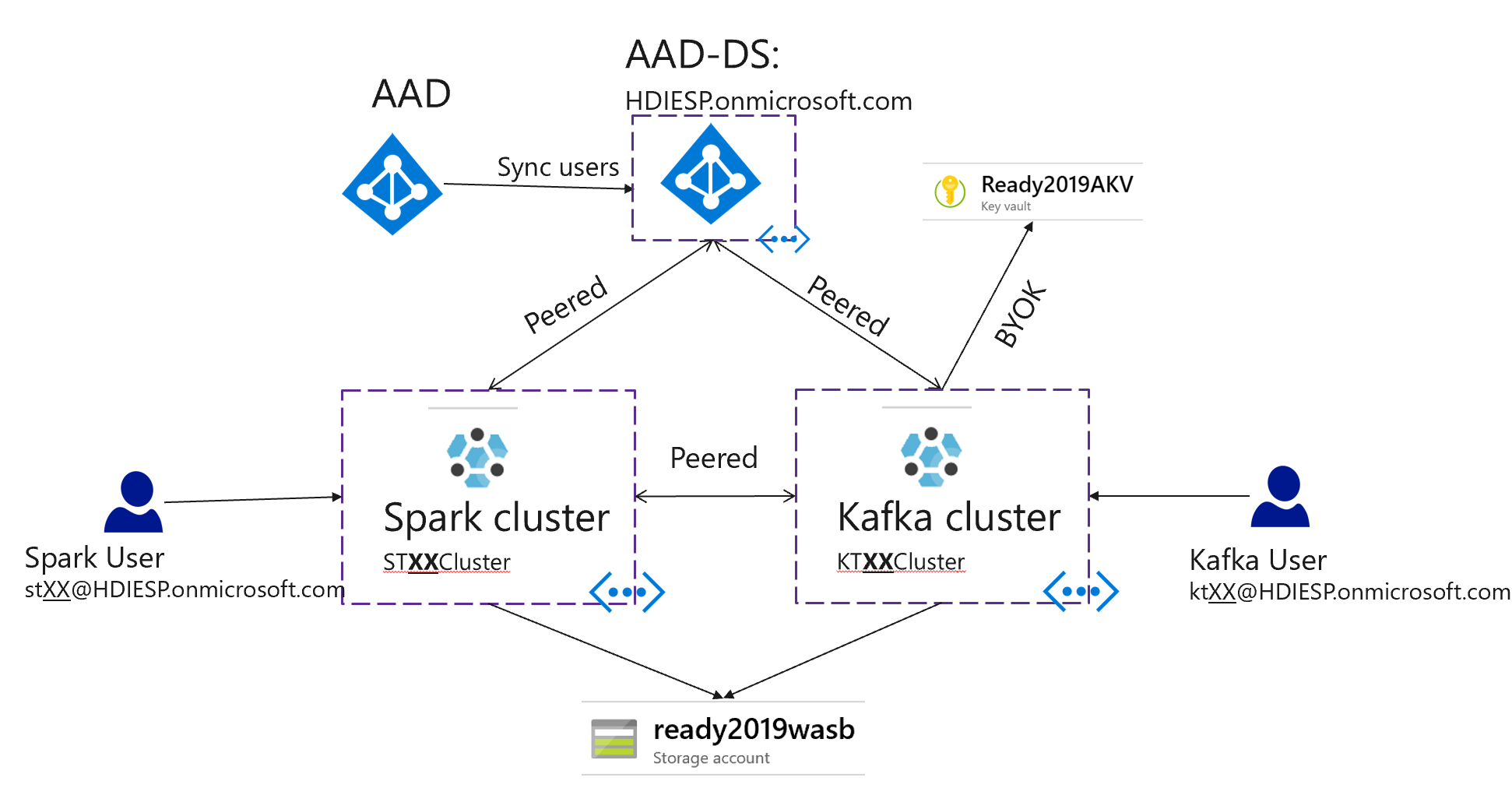
**Overview:**

This Lab walks through the setup of a typical streaming data pattern in Azure. Data is ingested in Apache Kafka on HDInsight into specific topics. This data in Kafka is then read by a Spark streaming job. What makes this pattern more relevant to enterprises is that both clusters have Enterprise Security Package enabled: hence access to Kafka topics can be gated by Apache Ranger policies.

This is the architecture that you’ll be creating:



Inside the Kafka cluster there will be two topics (topicA and topicB) to be created by Kafka User, and inside Spark cluster there will be a Spark streaming job that consume the data from Kafka using Kerberos authentication. All of this traffic will go through the two VNETs that are peered using private IPs.

Peering

**Kafka User (kt**XX**)**



topicA

topicB

**Kafka Cluster**

**<Kafka VNET>**

**<Spark VNET>**

**Spark Cluster**

**Spark User (st**XX**)**



Spark Job



You are the **Kafka** user. In this document, you will go through the setup of the Kafka cluster, including VNET and security. You will access the Ranger portal for the cluster to setup custom security policies for topics. Then you will create 2 topics, topicA and topicB and send test data to those topics.

Your Spark counterpart also has a document outlining steps for them. You may have to collaborate to get the streaming job running end to end.

Your counterpart will consume from either topic but will succeed or fail based on the policies you set for them.

**Pre-requisites:**

* AAD-DS is already setup. The lab instructor is the AAD admin and can provide access when needed. Think of the instructor as the AD team which is typically different from the big data team in enterprises. The AD team has already created your domain credentials which you will use later in this lab.
* An ESP Spark cluster deployed in a different VNET: Your teammate in the lab will be creating and preparing this VNET and cluster. When needed you can coordinate with him.

**Common Info you may need:**

* AAD-DS VNET:  **AD-VNET**
* AAD-DS Resource Group:  **AADDS-RG**
* AAD-DS domain name: **HDIESP.onmicrosoft.com**
* AAD-DS DNS Servers IPs: **10.0.0.5,10.0.0.4**
* WASB account name: **ready2019wasb**
* WASB RG: **Ready2019RG**
* Managed Identity: **Ready2019MI** (has access to AAD-DS, AKV key)
* MSI Resource Group: **Ready2019RG**
* Resource Group: Team**XX**RG
* Login to the Azure Portal with the following domain credentials
  + Credential: Domain UPN: kt**XX**@HDIESP.onmicrosoft.com , Pass: Ask Instructors.
* Replace **XX** in the rest of this doc with you team number at the top of this page
* Kafka\_Disk\_Encryption\_Key\_URL: https://ready2019akv.vault.azure.net/keys/Team01KafkaKey/751546d0694e494682c50f56360d9722

**Instructions:**

Login with your domain username in the common section above to portal.azure.com

We will use ARM templates to deploy resources needed in this Lab. However, portal instructions are also provided for the curious.

You can find the required templates on GitHub, use the “Deploy to Azure” when needed:

<https://github.com/tylerfox/HDInsight-ESP-Kafka-Spark>

**Template Steps**

**Step 1 – Create a VNET & Peer** **to AD VNET**

1. Go to GitHub link above and click on “template\_vnet.json” Deploy to Azure button or click here:

[C:\Users\dhgoel\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\B45917D1.tmp](https://portal.azure.com/#create/Microsoft.Template/uri/https%3A%2F%2Fraw.githubusercontent.com%2Ftylerfox%2FHDInsight-ESP-Kafka-Spark%2Fmaster%2Ftemplate_vnet.json)

1. Resource Group: Team**XX**RG
2. VNET name: KT**XX**VNET
3. Your IP range depends on your team number (XX). Let Y = (2 \*XX), then your VNET IP range is 10.Y.0.0/16 and Subnet IP range is 10.Y.0.0/24. For e.g., if your team number is 03, VNET IP range would be 10.6.0.0/16.
4. AADDS\_DNS\_Server\_IPs: from common info (10.0.0.5,10.0.0.4)
5. AADDS\_Resource\_Group\_Name: from common info (AADDS-RG)
6. AD\_VNET\_NAME: from common info (AD-VNET)
7. Hit “Purchase” to start the deployment.
8. This deployment creates a VNET/Subnet with the specified IP range, peers it to the AAD-DS VNET and sets custom DNS server settings.

**Step 2 – Create an HDInsight Kafka Cluster**

1. Go to GitHub link above and click on “template\_kafka.json” Deploy to Azure button:

[](https://portal.azure.com/#create/Microsoft.Template/uri/https%3A%2F%2Fraw.githubusercontent.com%2Ftylerfox%2FHDInsight-ESP-Kafka-Spark%2Fmaster%2Ftemplate_kafka.json)

1. Resource Group: Team**XX**RG
2. Name: KT**XX**Cluster
3. Admin\_SSH\_username: sshuser
4. Cluster\_Password: Pick a password and don’t forget it
5. Storage Account Name: from common info (ready2019wasb)
6. Storage Resource Group: from common info (Ready2019RG)
7. Kafka VNET name: from previous section (KT**XX**VNET)
8. AAD-DS resource group: from common info (AADDS-RG)
9. AAD-DS domain name: **HDIESP.onmicrosoft.com**
10. Cluster Admin AAD account: kt**XX**@HDIESP.onmicrosoft.com
11. Cluster access group name: Team**XX**AADGroup
12. Managed Identity RG and name: from common info
13. Kafka\_Disk\_Encryption\_Key\_URL: from common info

**Step 3 – Peer to Spark VNET:**

1. Go to GitHub link above and click on “template\_peer.json” Deploy to Azure button.

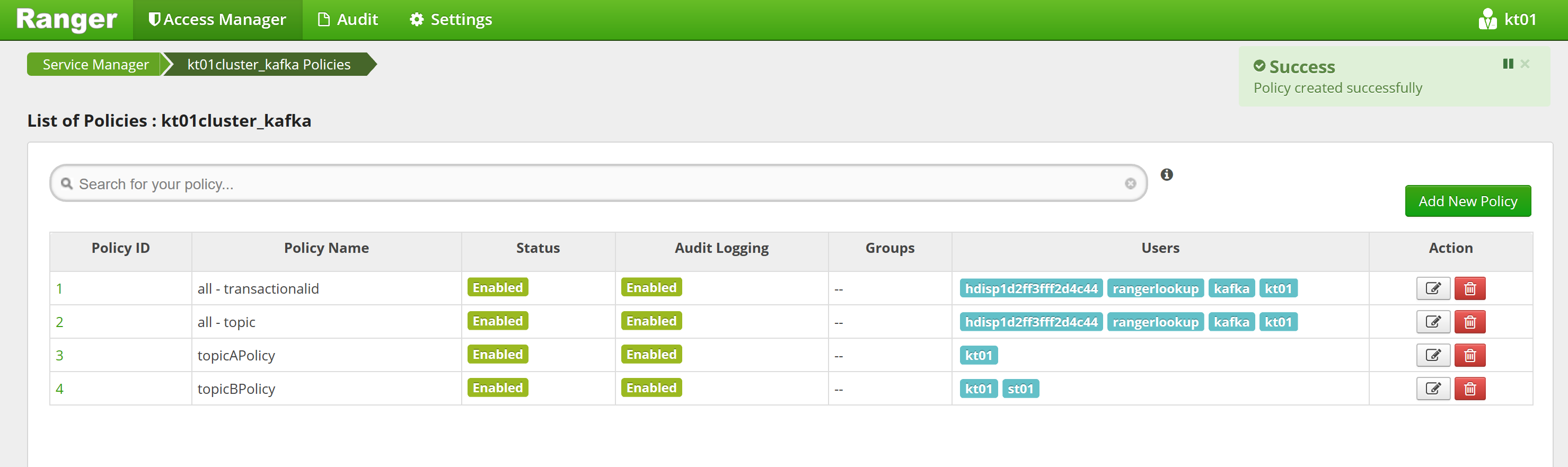
[C:\Users\dhgoel\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\10313B07.tmp](https://portal.azure.com/#create/Microsoft.Template/uri/https%3A%2F%2Fraw.githubusercontent.com%2Ftylerfox%2FHDInsight-ESP-Kafka-Spark%2Fmaster%2Ftemplate_peer.json)

1. Before you proceed check with your teammate or look in your resource group to make sure that Spark VNET with the name ST**XX**VNET is already created.
2. My VNET: KT**XX**VNET
3. Remote VNET: ST**XX**VNET

**Important: For the purposes of this lab, we have pre-created a cluster for you. For the remaining steps you can switch over to using this cluster. The resource group for your cluster is TeamXXARG, and the name of your cluster is KTXXACluster (instead of KTXXCluster that you just created), with XX being your team number. The steps below use the pre-created cluster name. If you want to use the cluster you just created, remove the ‘A’ in the cluster name.**

**Step 4 – Creating Policies with Ranger for Kafka topics**

1. Go to https://kt**XX**acluster.azurehdinsight.net/Ranger/ - (Note: the trailing ‘/’ at the end of the URL is required). Login with:
   * User: kt**XX**
   * Password: Provided by Instructors
2. Create policies to give Kafka User (kt**XX**) access to topicA
   * On the Ranger "Access Manager" Dashboard, click on the cluster name under Kafka
   * Click on "Add New Policy" on the top right
     + PolicyName: topicApolicy
     + Topic: topicA
     + Allow Conditions:
       - Select User: kt**XX**
       - Permissions: Consume, Create, Delete, Describe, Publish
   * Click on "Add"
3. Similarly, create a policy giving both kt**XX** and st**XX** access to topicB



**NOTE: For the following shell commands, you can use whatever SSH client you prefer (Azure CloudShell, PuTTy, MobaXterm etc.)**

**Step 5 – Download the jq utility on the cluster**

1. SSH into the cluster as sshuser to install jq utility

ssh sshuser@KT**XXA**Cluster-ssh.azurehdinsight.net

* + - Use the sshuser password provided by instructors

sudo apt -y install jq

1. You can log out

**Step 6 – Creating Topics on Kafka cluster**

1. SSH to the Kafka cluster using the domain username:

ssh kt**XX**@KT**XXA**Cluster-ssh.azurehdinsight.net

* + - Login with the domain password provided by instructors

1. Download the kafka-producer-consumer-esp.jar file from [Apache Kafka domain-joined producer consumer examples](https://github.com/Azure-Samples/hdinsight-kafka-java-get-started/tree/master/Prebuilt-Jars):

wget https://github.com/Azure-Samples/hdinsight-kafka-java-get-started/raw/master/Prebuilt-Jars/kafka-producer-consumer-esp.jar

1. Run the following commands to create topics

export USER=kt**XX**@HDIESP.onmicrosoft.com

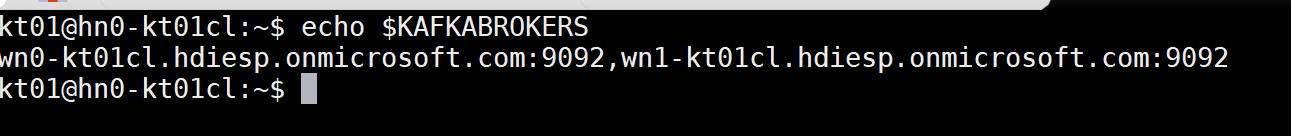
export CLUSTERNAME=kt**XXa**cluster

export KAFKABROKERS=`curl -sS -u $USER -G https://$CLUSTERNAME.azurehdinsight.net/api/v1/clusters/$CLUSTERNAME/services/KAFKA/components/KAFKA\_BROKER | jq -r '["\(.host\_components[].HostRoles.host\_name):9092"] | join(",")' | cut -d',' -f1,2`;

* + - You will have to provide the Kafka user domain password provided by instructors

echo $KAFKABROKERS

* + - It should look something like below:



java -jar -Djava.security.auth.login.config=/usr/hdp/current/kafka-broker/config/kafka\_client\_jaas.conf kafka-producer-consumer-esp.jar create **topicA** $KAFKABROKERS

* + - Output may include a “Failed” message, but you can ignore that as long as the final message is to load class“Topic topicA created”

java -jar -Djava.security.auth.login.config=/usr/hdp/current/kafka-broker/config/kafka\_client\_jaas.conf kafka-producer-consumer-esp.jar create **topicB** $KAFKABROKERS

**Step 7 – Send sample data to Kafka topic**

1. Continue previous session as domain User A (kt**XX**) into Kafka cluster
2. Send data to topicA and topicB

java -jar -Djava.security.auth.login.config=/usr/hdp/current/kafka-broker/config/kafka\_client\_jaas.conf kafka-producer-consumer-esp.jar producer **topicA** $KAFKABROKERS

java -jar -Djava.security.auth.login.config=/usr/hdp/current/kafka-broker/config/kafka\_client\_jaas.conf kafka-producer-consumer-esp.jar producer **topicB** $KAFKABROKERS

You can work with your teammate to consume from these topics in the spark cluster. You can repeat sending data to these topics as needed and see the results on the spark cluster.

**Working with your Spark partner**

While you’re doing your steps, your Spark application developer might need some details from you. Here’s how you can help them.

**Getting Kafka Broker IP information**

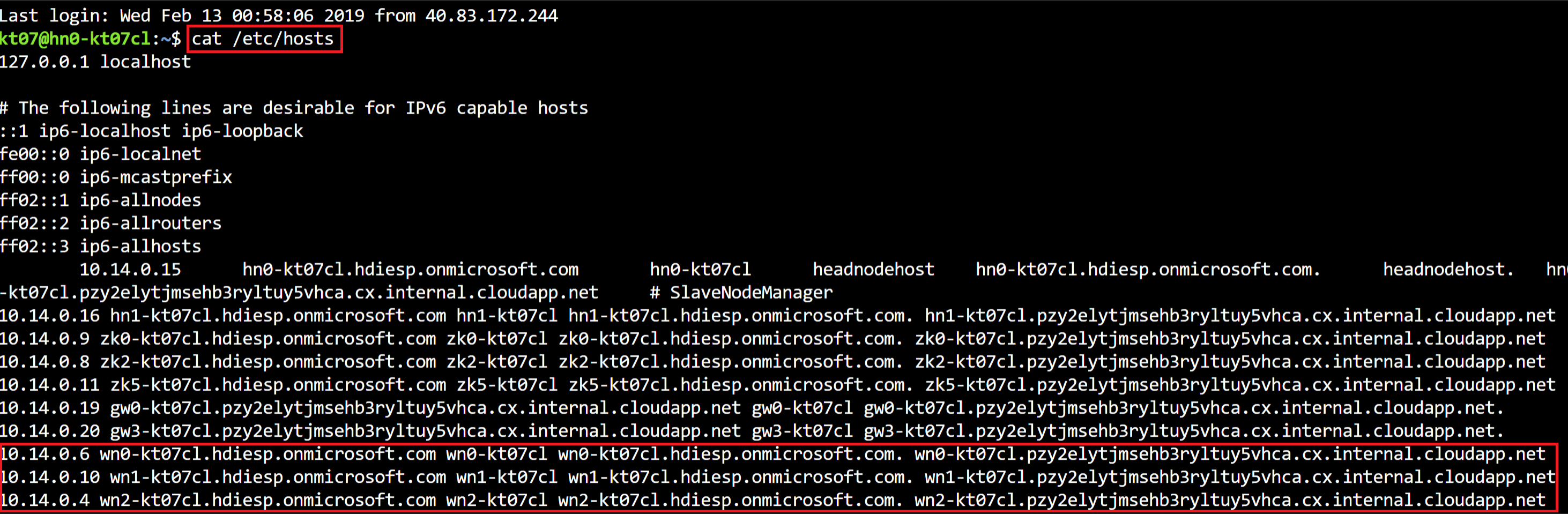
For the spark application to talk to Kafka brokers, it will need the IP addresses of the brokers. You provide that by looking into the /etc/hosts file in any node in the Kafka cluster

SSH to the Kafka cluster using the domain username:

ssh ktXX@KTXXACluster-ssh.azurehdinsight.net

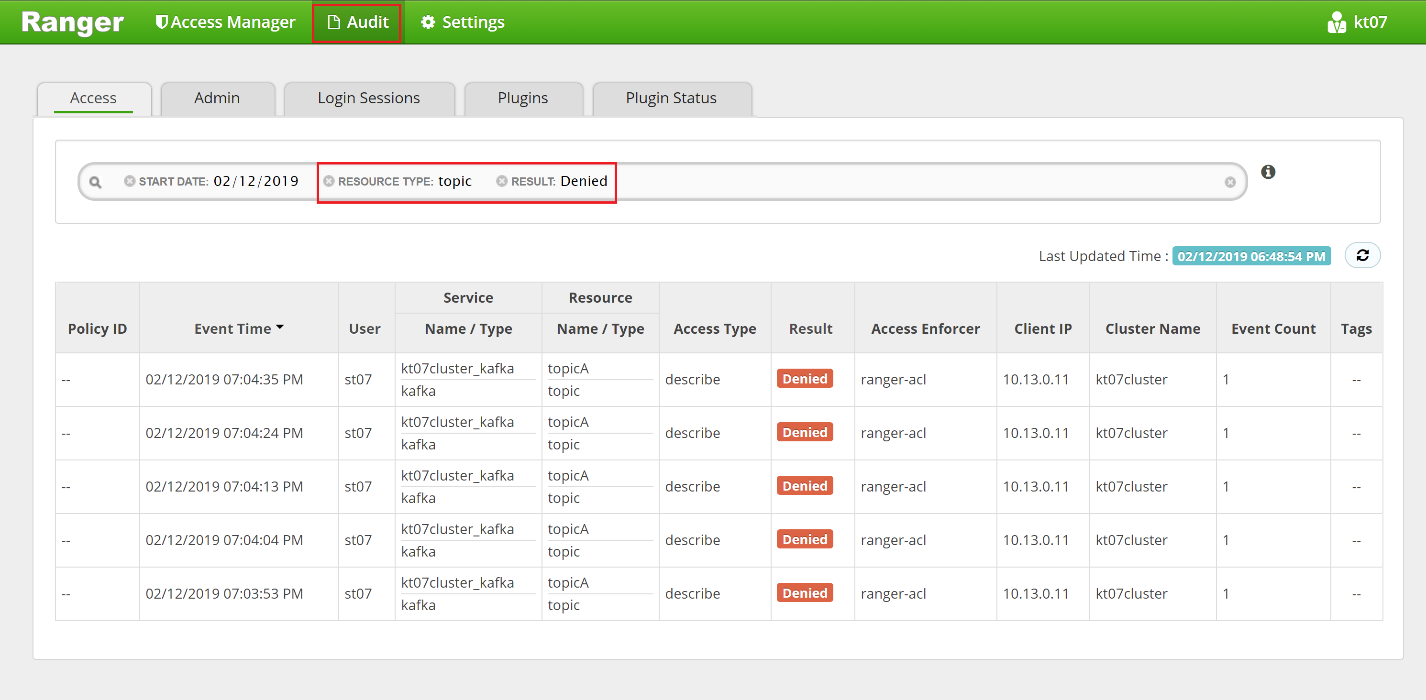
* + - Login with the domain password provided by instructors

cat /etc/hosts

* + - Result should look like below. Share with them the lines corresponding to the worker nodes (wn0, wn1 and wn2)

**Auditing topic access in Ranger**

As the Kafka admin, you can check on the access requests made to topics via Ranger. Go back to the Ranger application (described in step 4 in the instructions) and go to “Audit” view from the top navigation bar. Filter for Resource Type “topic”. It should look something like below:



**Appendix**

**Portal Steps for VNET and cluster setup**

**Step 1 – Create a VNET & Peer**

**1a. Create a VNET**

* VNET name: KT**XX**VNET
* Resource Group: TeamXXRG
* Location: East US 2
* All other settings are default

**1b. Peer VNETs and set custom DNS server**

* Create Peering in KT**XX**VNET
  1. Name: Peer2ADVNET
  2. Virtual network: Select the AAD-DS VNET from common info
  3. Configuration: "Enabled" for "Allow virtual network access"
  4. Configuration: Select "Allow forwarded Traffic"
* Create Peering in AD-VNET
  1. Name: Peer2KT**XX**VNET
  2. Virtual network: KT**XX**VNET
  3. Configuration: "Enabled" for "Allow virtual network access"
  4. Configuration: Select "Allow forwarded Traffic"
* Confirm that the Peering status says "Connected"
* Similarly, Peer to ST**XX**VNET (User B will create the ST**XX**VNET in their instructions)
* Set custom DNS server - provide IP address of AADDS
  1. In KT**XX**VNET, select DNS Servers from the navigation blade
  2. Select Custom and provide from common info

**Step 2 - Create an HDInsight Kafka Cluster**

* Name: KT**XX**cluster
* Cluster Type: Kafka
* Version: Kafka 1.1.0 on HDI 3.6
* Remember the sshuser and admin password that you choose.
* Resource Group: Team**XX**RG
* Location: East US 2
* Enterprise Security Package: Select "Enabled"
* Admin user: kt**XX**
* Cluster users: kt**XX** and st**XX**; OR you can specify AAD Group: team**XX**grp
* Virtual Network: KT**XX**VNET
* Identity MSI: **Ready2019MI**
* Storage Azure Storage: ready2019wasb
* Choose 3 worker nodes and 1 disk per node with the default option for VM sizes.