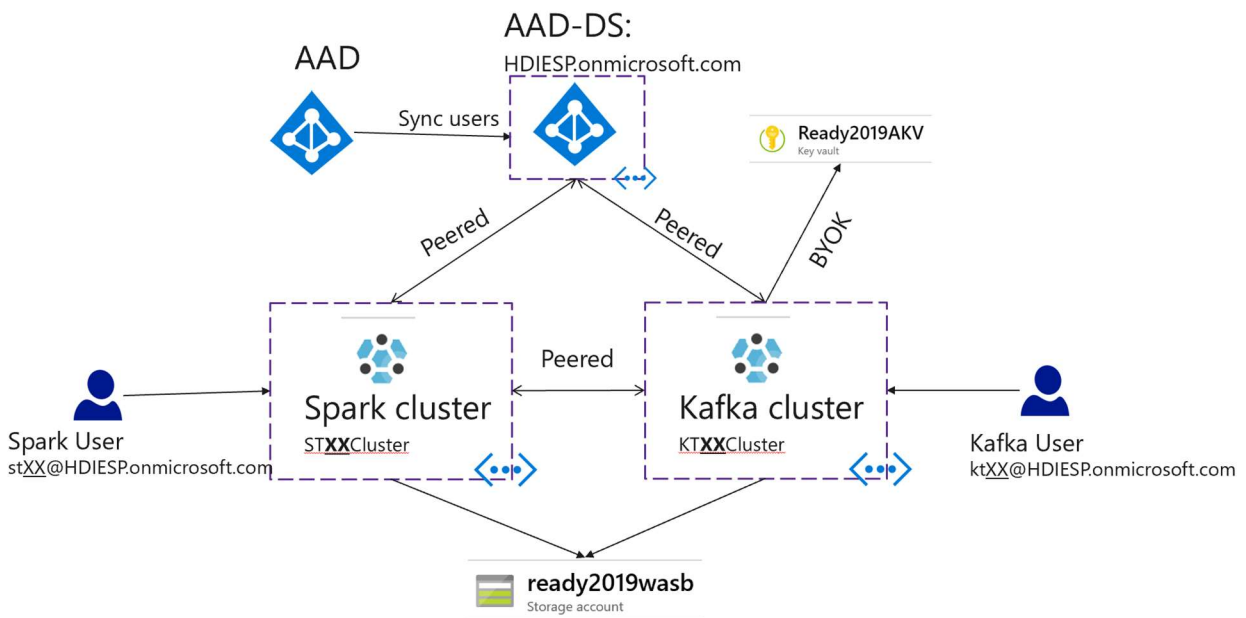


## Kafka User Instructions

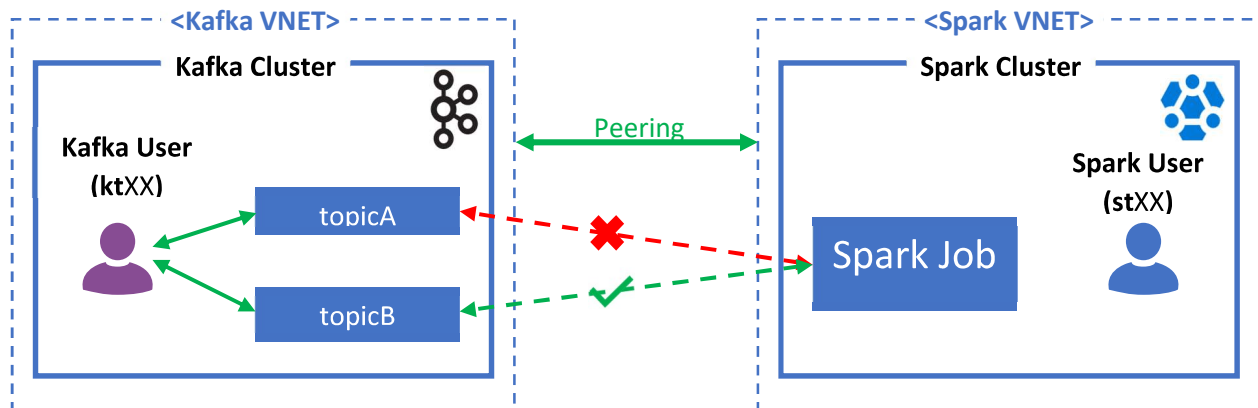
## Overview:

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.



## Kafka User Instructions

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>

## Kafka User Instructions

### 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:



2. Resource Group: Team**XX**RG
3. VNET name: KT**XX**VNET
4. 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.
5. AADDS\_DNS\_Server\_IPs: from common info (10.0.0.5,10.0.0.4)
6. AADDS\_Resource\_Group\_Name: from common info (AADDS-RG)
7. AD\_VNET\_NAME: from common info (AD-VNET)
8. Hit “Purchase” to start the deployment.
9. 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:



2. Resource Group: Team**XX**RG
3. Name: KT**XX**Cluster
4. Admin\_SSH\_username: sshuser
5. Cluster\_Password: Pick a password and don't forget it
6. Storage Account Name: from common info (ready2019wasb)
7. Storage Resource Group: from common info (Ready2019RG)
8. Kafka VNET name: from previous section (KT**XX**VNET)
9. AAD-DS resource group: from common info (AADDS-RG)
10. AAD-DS domain name: **HDIESP.onmicrosoft.com**
11. Cluster Admin AAD account: kt**XX**@HDIESP.onmicrosoft.com
12. Cluster access group name: Team**XX**AADGroup
13. Managed Identity RG and name: from common info
14. 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.

## Kafka User Instructions



- Before you proceed check with your teammate or look in your resource group to make sure that Spark VNET with the name STXXVNET is already created.
- My VNET: KTXXVNET
- Remote VNET: STXXVNET

**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

- Go to <https://ktXXcluster.azurehdinsight.net/Ranger/> - (Note: the trailing '/' at the end of the URL is required). Login with:
  - User: ktXX
  - Password: Provided by Instructors
- Create policies to give Kafka User (ktXX) 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: ktXX
      - Permissions: Consume, Create, Delete, Describe, Publish
  - Click on "Add"
- Similarly, create a policy giving both ktXX and stXX access to topicB

**Ranger** Access Manager Audit Settings kt01

Service Manager kt01cluster\_kafka Policies Success Policy created successfully

List of Policies : kt01cluster\_kafka

Search for your policy...

Add New Policy

Policy ID	Policy Name	Status	Audit Logging	Groups	Users	Action
1	all - transactionalid	Enabled	Enabled	--	hdisp1d2ff3ff2d4c44 rangerlookup kafka kt01	
2	all - topic	Enabled	Enabled	--	hdisp1d2ff3ff2d4c44 rangerlookup kafka kt01	
3	topicAPolicy	Enabled	Enabled	--	kt01	
4	topicBPolicy	Enabled	Enabled	--	kt01 st01	

**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

### Kafka User Instructions

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 instructorssudo apt -y install jq
2. 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
2. Download the kafka-producer-consumer-esp.jar file from [Apache Kafka domain-joined producer consumer examples](#):  
wget https://github.com/Azure-Samples/hdinsight-kafka-java-get-started/raw/master/Prebuilt-Jars/kafka-producer-consumer-esp.jar
3. 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/co  
mponents/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:

```
kt01@hn0-kt01cl:~$ echo $KAFKABROKERS
wn0-kt01cl.hdiesp.onmicrosoft.com:9092,wn1-kt01cl.hdiesp.onmicrosoft.com:9092
kt01@hn0-kt01cl:~$
```

```
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

## Kafka User Instructions

## 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)

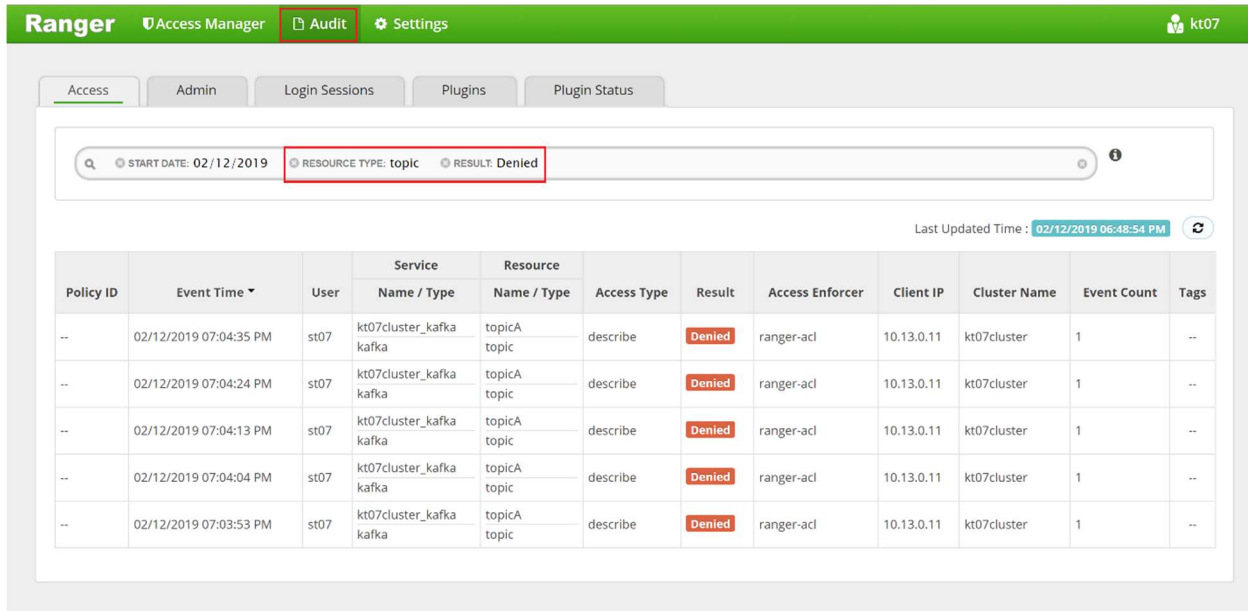
```
Last login: Wed Feb 13 00:58:06 2019 from 40.83.172.244
kt07@hn0-kt07cl:~$ cat /etc/hosts
127.0.0.1 localhost

# The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts
10.14.0.15 hn0-kt07cl.hdiexp.onmicrosoft.com hn0-kt07cl headnodehost hn0-kt07cl.hdiexp.onmicrosoft.com headnodehost hn
-kt07cl.pzy2elytjmsehb3ryltuy5vhca.cx.internal.cloudapp.net # SlaveNodeManager
10.14.0.16 hn1-kt07cl.hdiexp.onmicrosoft.com hn1-kt07cl hn1-kt07cl.hdiexp.onmicrosoft.com hn1-kt07cl.pzy2elytjmsehb3ryltuy5vhca.cx.internal.cloudapp.net
10.14.0.9 zk0-kt07cl.hdiexp.onmicrosoft.com zk0-kt07cl zk0-kt07cl.hdiexp.onmicrosoft.com zk0-kt07cl.pzy2elytjmsehb3ryltuy5vhca.cx.internal.cloudapp.net
10.14.0.8 zk2-kt07cl.hdiexp.onmicrosoft.com zk2-kt07cl zk2-kt07cl.hdiexp.onmicrosoft.com zk2-kt07cl.pzy2elytjmsehb3ryltuy5vhca.cx.internal.cloudapp.net
10.14.0.11 zk5-kt07cl.hdiexp.onmicrosoft.com zk5-kt07cl zk5-kt07cl.hdiexp.onmicrosoft.com zk5-kt07cl.pzy2elytjmsehb3ryltuy5vhca.cx.internal.cloudapp.net
10.14.0.19 gw0-kt07cl.pzy2elytjmsehb3ryltuy5vhca.cx.internal.cloudapp.net gw0-kt07cl gw0-kt07cl.pzy2elytjmsehb3ryltuy5vhca.cx.internal.cloudapp.net
10.14.0.20 gw3-kt07cl.pzy2elytjmsehb3ryltuy5vhca.cx.internal.cloudapp.net gw3-kt07cl gw3-kt07cl.pzy2elytjmsehb3ryltuy5vhca.cx.internal.cloudapp.net
10.14.0.6 wn0-kt07cl.hdiexp.onmicrosoft.com wn0-kt07cl wn0-kt07cl.hdiexp.onmicrosoft.com wn0-kt07cl.pzy2elytjmsehb3ryltuy5vhca.cx.internal.cloudapp.net
10.14.0.10 wn1-kt07cl.hdiexp.onmicrosoft.com wn1-kt07cl wn1-kt07cl.hdiexp.onmicrosoft.com wn1-kt07cl.pzy2elytjmsehb3ryltuy5vhca.cx.internal.cloudapp.net
10.14.0.4 wn2-kt07cl.hdiexp.onmicrosoft.com wn2-kt07cl wn2-kt07cl.hdiexp.onmicrosoft.com wn2-kt07cl.pzy2elytjmsehb3ryltuy5vhca.cx.internal.cloudapp.net
```

## Kafka User Instructions

### 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:



The screenshot shows the Ranger web interface. The top navigation bar is green with the 'Ranger' logo and links to 'Access Manager', 'Audit' (highlighted with a red box), and 'Settings'. Below this is a sub-navigation bar with 'Access', 'Admin', 'Login Sessions', 'Plugins', and 'Plugin Status'. The 'Access' tab is selected. A search bar at the top of the main content area contains filters: 'START DATE: 02/12/2019', 'RESOURCE TYPE: topic' (highlighted with a red box), and 'RESULT: Denied'. Below the search bar, a table displays the audit results. The table has columns for Policy ID, Event Time, User, Service Name / Type, Resource Name / Type, Access Type, Result, Access Enforcer, Client IP, Cluster Name, Event Count, and Tags. The 'Result' column shows 'Denied' for all entries, which are highlighted in red. The 'Access Enforcer' column shows 'ranger-acl' for all entries. The 'Client IP' column shows '10.13.0.11' for all entries. The 'Cluster Name' column shows 'kt07cluster' for all entries. The 'Event Count' column shows '1' for all entries. The 'Tags' column shows '--' for all entries. The 'Last Updated Time' is 02/12/2019 06:48:54 PM.

Policy ID	Event Time	User	Service Name / Type	Resource Name / Type	Access Type	Result	Access Enforcer	Client IP	Cluster Name	Event Count	Tags
--	02/12/2019 07:04:35 PM	st07	kt07cluster_kafka kafka	topicA topic	describe	Denied	ranger-acl	10.13.0.11	kt07cluster	1	--
--	02/12/2019 07:04:24 PM	st07	kt07cluster_kafka kafka	topicA topic	describe	Denied	ranger-acl	10.13.0.11	kt07cluster	1	--
--	02/12/2019 07:04:13 PM	st07	kt07cluster_kafka kafka	topicA topic	describe	Denied	ranger-acl	10.13.0.11	kt07cluster	1	--
--	02/12/2019 07:04:04 PM	st07	kt07cluster_kafka kafka	topicA topic	describe	Denied	ranger-acl	10.13.0.11	kt07cluster	1	--
--	02/12/2019 07:03:53 PM	st07	kt07cluster_kafka kafka	topicA topic	describe	Denied	ranger-acl	10.13.0.11	kt07cluster	1	--

## Kafka User Instructions

# Appendix

## Portal Steps for VNET and cluster setup

### Step 1 – Create a VNET & Peer

#### 1a. Create a VNET

- VNET name: KTXXVNET
- Resource Group: TeamXXRG
- Location: East US 2
- All other settings are default

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

- Create Peering in KTXXVNET
  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: Peer2KTXXVNET
  2. Virtual network: KTXXVNET
  3. Configuration: "Enabled" for "Allow virtual network access"
  4. Configuration: Select "Allow forwarded Traffic"
- Confirm that the Peering status says "Connected"
- Similarly, Peer to STXXVNET (User B will create the STXXVNET in their instructions)
- Set custom DNS server - provide IP address of AADDS
  1. In KTXXVNET, select DNS Servers from the navigation blade
  2. Select Custom and provide from common info

### Step 2 - Create an HDInsight Kafka Cluster

- Name: KTXXcluster
- Cluster Type: Kafka
- Version: Kafka 1.1.0 on HDI 3.6
- Remember the sshuser and admin password that you choose.
- Resource Group: TeamXXRG
- Location: East US 2
- Enterprise Security Package: Select "Enabled"
- Admin user: ktXX
- Cluster users: ktXX and stXX; OR you can specify AAD Group: teamXXgrp
- Virtual Network: KTXXVNET
- Identity MSI: **Ready2019MI**
- Storage Azure Storage: ready2019wasb
- Choose 3 worker nodes and 1 disk per node with the default option for VM sizes.