Multiple Listeners for Kafka Brokers

As part of adding Kerberos and SSL authentication support for Kafka, community decided that it would be beneficial to have one port per connection type. E.g. to have one port for SSL, one port of Kerberos and one port for “plain” authentication (e.g. current case). Reasoning for this decision is not part of this proposal, but are discussed in the security design wiki: https://cwiki.apache.org/confluence/display/KAFKA/Security

This is a short document to detail the changes required for this.

This patch is only concerned with the ability for a broker to listen to multiple ports. It does not attempt to add different channel implementations to those ports.

Although not explicitly specified in the security design doc, this patch will add the ability to support not just multiple ports but multiple ip:port pairs - so it will be possible to use SSL over the external network and plaintext on internal networks.

Changes to Broker:

Broker object currently includes id, host and port. This means the broker object (as owner of replicas) and the information required to connect to the broker is tightly coupled throughout the code-base. Most of the work in this patch involves decoupling the broker entity with the communication channels to the broker.

The Broker object will include the id and a list of (host,port,protocol) pairs, which we will call end-points. This object is what gets persisted to ZK when registering brokers. The new broker will need to know how to de-serialize both old and new broker JSON from ZK, based on the version. The JSON was intentionally kept compatible with the previous version to allow rolling upgrades.

The following JSON will represent broker in ZK:

```
 { "version":1,
    "jmx_port":9999,
    "timestamp":2233345666,
    "host":"localhost",
   “port”:9092,
    "endpoints": [
            {"plaintext://localhost:9092"},
            {“ssl://localhost:9093”},
            {“kerberos://localhost:9094”}
]
}
```

We’ll also add a BrokerEndPoint object, which will represent a method of connecting to a broker. BrokerEndPoint contains id, host and port. This doesn’t get persisted. Components that need to connect to a broker will get a BrokerEndPoint from the broker based on the security protocol they wish to use. Basically, BrokerEndPoint replaces the old Broker in most protocols.

All these changes mean that we need to serialize and de-serialize Broker differently (to both ZK and wire) and we need to serialize / de-serialize BrokerEndPoints to wire.

Configuration changes:

1. ConsumerConfig: Adding a configuration to specify security.protocol (defaulting to plaintext). The new consumer (and producers) will just need to specify port, but the existing consumer takes the brokers directly from ZK and we’ll need to know which of the broker ports to use.
2. KafkaConfig: Instead of specifying port, advertised port and advertised host, we want to specify host-port-lists: comma separated pairs of protocol, host and port (ssl://192.1.1.8:9093, plaintext://10.1.1.5:9092).
   We also need to add replication.security.protocol configuration controlling which port and protocol the broker will use to connect to other brokers.

Protocol changes:

ConsumerMetadataResponse and TopicMetadataResponse return a list of Brokers that the consumer and producer can connect to. We will change these to return BrokerEndPoints instead, based on the security protocol the client is using (determined by the port the client connected to). Note that this means the wire protocol does not change.

UpdateMetadataRequest - is used between brokers and needs to contain the entire broker information. We bump the protocol version of this request and make sure that the new broker can accept both versions of requests, and will only send the new request version if a configuration was modified to useNewVersion=true

LeaderAndIsrRequest - will use BrokerEndPoint instead of broker (based on replication protocol specified in configs). Wire protocol will not change.

Tools that need to support multiple ports:

UpdateOffsetInZK

ConsumerOffsetChecker

GetOffsetShell

ReplicaVerificationTool

SimpleConsumerShell

I think for most of them its a matter of using BrokerEndPoint instead of Broker. Those that get list of brokers from ZK (ConsumerOffsetChecker for instance) will need to take security protocol as a parameter (and default to plain-text for compatibility)

Additional components that need to be modified:

1. ClientUtils should use BrokerEndPoint for connecting to brokers
2. Producer + ProducerPool
3. ZKUtil requires additional functionality:
    a. getBrokerEndPoint(zkClient, brokerId, protocol)
    b. getAllBrokerEndPointsForProtocol(zkClient,protocol)
    c. registerBroker will need to change to support new broker format (one that contains multiple end-points instead of single port)
    d. Probably more stuff I'm not seeing yet
4. Controller - opens channel to brokers. we'll use KafkaConfig's new SecurityProtocol configuration to decide which end-point to use
5. KafkaServer - needs to start listening on a list of ports and accepting connections there. For the moment all those connections will use the same SocketChannel we currently have
6. ReplicaFetcherThread will need to connect to brokers on a specific end-point
7. OffsetFetcher (on consumer)
8. Probably few more I'm currently missing

Upgrades and Backward Compatibility

This patch will change broker representation in ZK and will also modify part of the wire protocol.

Wire protocol will be versioned, so new brokers will be able to accept and reply to both old and new clients. This allows us to upgrade brokers first and also use old mirror-maker to replicate between old and new clusters.

ZK representation also has versions, but in order to allow old clients to connect to new brokers (or for old and new brokers to talk to each other), we are keeping the "plaintext" channel as the existing "host" and "port" fields in the broker JSON.

If users choose not to support plain-text (for security reasons), old brokers and clients will be unable to connect - but this is expected.

(To clarify the last line, because I just got bitten by this:

If your listeners do not contain PLAINTEXT for whatever reason, you need a cluster with 100% new brokers, you need to set replication.security.protocol to something non-default and you need to set use.new.wire.protocol=true for all brokers.