KIP-374: Add '--help' option to all available Kafka CLI commands

Status

Current state: *Accepted*

Discussion thread: *here*

Vote thread: *here*

JIRA: *Unable to render Jira issues macro, execution error.*

Please keep the discussion on the mailing list rather than commenting on the wiki (wiki discussions get unwieldy fast).

Motivation

Currently, the '--help' option is not recognized by some Kafka commands . For example:

```
$ kafka-console-producer --help
help is not a recognized option
```

However, the '--help' option is supported by other commands:

```
$ kafka-verifiable-producer --help
usage: verifiable-producer [-h] --topic TOPIC --broker-list HOST1:PORT1[,HOST2:PORT2[...]] [--max-messages MAX-
MESSAGES]
                           [--throughput THROUGHPUT] [--acks ACKS] [--producer.config CONFIG_FILE]
                           [--message-create-time CREATETIME] [--value-prefix VALUE-PREFIX]
```

```
...
```

To provide a consistent user experience, it would be nice to add a '--help' option to all Kafka commands.

Public Interfaces

Add a '--help' option to all Kafka commands that provides information about their usage.

Proposed Changes

Parsing logic of the Kafka commands shall be modified to handle the "--help" option.

Code modifications (git diff)

core/src/main/scala/kafka/admin/BrokerApiVersionsCommand.scala

```
@@ -72,7 +72,13 @@ object BrokerApiVersionsCommand {
                                     .withRequiredArg
                                     .describedAs("server(s) to use for bootstrapping")
                                     .ofType(classOf[String])
+    val helpOpt = parser.accepts("help", "Print usage information.")
+
     val options = parser.parse(args : _*)
+
+    if (options.has(helpOpt))
+      CommandLineUtils.printUsageAndDie(parser, "Usage:")
+
     checkArgs()

     def checkArgs() {
```

core/src/main/scala/kafka/admin/ConfigCommand.scala

```
@@ -72,7 +72,7 @@ object ConfigCommand extends Config {
     try {
       val opts = new ConfigCommandOptions(args)

-      if (args.length == 0)
+      if (args.length == 0 || opts.options.has(opts.helpOpt))
         CommandLineUtils.printUsageAndDie(opts.parser, "Add/Remove entity config for a topic, client, user or
broker")

       opts.checkArgs()
```

core/src/main/scala/kafka/admin/ConsumerGroupCommand.scala

```
@@ -41,7 +41,7 @@ object ConsumerGroupCommand extends Logging {
   def main(args: Array[String]) {
     val opts = new ConsumerGroupCommandOptions(args)

-    if (args.length == 0)
+    if (args.length == 0 || opts.options.has(opts.helpOpt))
       CommandLineUtils.printUsageAndDie(opts.parser, "List all consumer groups, describe a consumer group,
delete consumer group info, or reset consumer group offsets.")

     // should have exactly one action
@@ -773,6 +773,7 @@ object ConsumerGroupCommand extends Logging {
                           .availableIf(describeOpt)
     val stateOpt = parser.accepts("state", StateDoc)
                          .availableIf(describeOpt)
+    val helpOpt = parser.accepts("help", "Print usage information.")

     parser.mutuallyExclusive(membersOpt, offsetsOpt, stateOpt)
```

core/src/main/scala/kafka/admin/DeleteRecordsCommand.scala

```
@@ -129,8 +129,13 @@ object DeleteRecordsCommand {
                                       .withRequiredArg
                                       .describedAs("command config property file path")
                                       .ofType(classOf[String])
+    val helpOpt = parser.accepts("help", "Print usage information.")

     val options = parser.parse(args : _*)
+
+    if (args.length == 0 || options.has(helpOpt))
+      CommandLineUtils.printUsageAndDie(parser, "Delete records from given partitions till specified offset")
+
     CommandLineUtils.checkRequiredArgs(parser, options, bootstrapServerOpt, offsetJsonFileOpt)
   }
 }
```

core/src/main/scala/kafka/admin/LogDirsCommand.scala

```
+    val helpOpt = parser.accepts("help", "Print usage information.")
+
    val options = parser.parse(args: _*)

-    if (args.length == 0)
+    if (args.length == 0 || options.has(helpOpt))
      CommandLineUtils.printUsageAndDie(parser, "Command to check log directory usage on specified brokers")

    CommandLineUtils.checkRequiredArgs(parser, options, bootstrapServerOpt, describeOpt)
```

core/src/main/scala/kafka/admin/PreferredReplicaLeaderElectionCommand.scala

```
@@ -43,13 +43,14 @@ object PreferredReplicaLeaderElectionCommand extends Logging {
       .withRequiredArg
       .describedAs("urls")
       .ofType(classOf[String])
-
-    if(args.length == 0)
-      CommandLineUtils.printUsageAndDie(parser, "This tool causes leadership for each partition to be
transferred back to the 'preferred replica'," +
-                                        " it can be used to balance leadership among the servers.")
+    val helpOpt = parser.accepts("help", "Print usage information.")

     val options = parser.parse(args : _*)

+    if(args.length == 0 || options.has(helpOpt))
+      CommandLineUtils.printUsageAndDie(parser, "This tool causes leadership for each partition to be
transferred back to the 'preferred replica'," +
+        " it can be used to balance leadership among the servers.")
+
     CommandLineUtils.checkRequiredArgs(parser, options, zkConnectOpt)

     val zkConnect = options.valueOf(zkConnectOpt)
```

core/src/main/scala/kafka/admin/ReassignPartitionsCommand.scala

```
@@ -415,7 +415,7 @@ object ReassignPartitionsCommand extends Logging {
   def validateAndParseArgs(args: Array[String]): ReassignPartitionsCommandOptions = {
     val opts = new ReassignPartitionsCommandOptions(args)

-    if(args.length == 0)
+    if(args.length == 0 || opts.options.has(opts.helpOpt))
       CommandLineUtils.printUsageAndDie(opts.parser, "This command moves topic partitions between replicas.")

     // Should have exactly one action
@@ -500,6 +500,7 @@ object ReassignPartitionsCommand extends Logging {
                       .describedAs("timeout")
                       .ofType(classOf[Long])
                       .defaultsTo(10000)
+    val helpOpt = parser.accepts("help", "Print usage information.")
     val options = parser.parse(args : _*)
   }
 }
```

core/src/main/scala/kafka/admin/TopicCommand.scala

```
@@ -43,7 +43,7 @@ object TopicCommand extends Logging {

     val opts = new TopicCommandOptions(args)

-    if(args.length == 0)
+    if(args.length == 0 || opts.options.has(opts.helpOpt))
       CommandLineUtils.printUsageAndDie(opts.parser, "Create, delete, describe, or change a topic.")

     // should have exactly one action
```

core/src/main/scala/kafka/admin/ZkSecurityMigrator.scala

```
@@ -76,7 +76,7 @@ object ZkSecurityMigrator extends Logging {
     val helpOpt = parser.accepts("help", "Print usage information.")

     val options = parser.parse(args : _*)
-    if (options.has(helpOpt))
+    if (args.length == 0 || options.has(helpOpt))
       CommandLineUtils.printUsageAndDie(parser, usageMessage)

     if (jaasFile == null) {
```

core/src/main/scala/kafka/tools/ConsoleConsumer.scala

```
@@ -274,11 +274,17 @@ object ConsoleConsumer extends Logging {
       .describedAs("consumer group id")
       .ofType(classOf[String])

+    val helpOpt = parser.accepts("help", "Print usage information.")
+
     if (args.length == 0)
       CommandLineUtils.printUsageAndDie(parser, "The console consumer is a tool that reads data from Kafka and
outputs it to standard output.")

     var groupIdPassed = true
     val options: OptionSet = tryParse(parser, args)
+
+    if (options.has(helpOpt))
+      CommandLineUtils.printUsageAndDie(parser, "Usage:")
+
     val enableSystestEventsLogging = options.has(enableSystestEventsLoggingOpt)

     // topic must be specified.
```

core/src/main/scala/kafka/tools/ConsoleProducer.scala

```
@@ -203,10 +203,13 @@ object ConsoleProducer {
       .withRequiredArg
       .describedAs("config file")
       .ofType(classOf[String])
+    val helpOpt = parser.accepts("help", "Print usage information.")

     val options = parser.parse(args : _*)
-    if (args.length == 0)
+
+    if (args.length == 0 || options.has(helpOpt))
       CommandLineUtils.printUsageAndDie(parser, "Read data from standard input and publish it to Kafka.")
+
     CommandLineUtils.checkRequiredArgs(parser, options, topicOpt, brokerListOpt)

     val topic = options.valueOf(topicOpt)
```

core/src/main/scala/kafka/tools/ConsumerPerformance.scala

```
@@ -253,9 +253,13 @@ object ConsumerPerformance extends LazyLogging {
       .describedAs("milliseconds")
       .ofType(classOf[Long])
       .defaultsTo(10000)
+    val helpOpt = parser.accepts("help", "Print usage information.")

     val options = parser.parse(args: _*)

+    if (args.length == 0 || options.has(helpOpt))
+      CommandLineUtils.printUsageAndDie(parser, "Performance test for the full zookeeper consumer.")
+
     CommandLineUtils.checkRequiredArgs(parser, options, topicOpt, numMessagesOpt, bootstrapServersOpt)

     val printMetrics = options.has(printMetricsOpt)
```

core/src/main/scala/kafka/tools/MirrorMaker.scala

```
@@ -157,16 +157,10 @@ object MirrorMaker extends Logging with KafkaMetricsGroup {

       val helpOpt = parser.accepts("help", "Print this message.")

-      if (args.length == 0)
-        CommandLineUtils.printUsageAndDie(parser, "Continuously copy data between two Kafka clusters.")
-
-
       val options = parser.parse(args: _*)

-      if (options.has(helpOpt)) {
-        parser.printHelpOn(System.out)
-        sys.exit(0)
-      }
+      if (args.length == 0 || options.has(helpOpt))
+        CommandLineUtils.printUsageAndDie(parser, "Continuously copy data between two Kafka clusters.")

       CommandLineUtils.checkRequiredArgs(parser, options, consumerConfigOpt, producerConfigOpt)
```

core/src/main/scala/kafka/tools/ReplicaVerificationTool.scala

```
@@ -105,11 +105,13 @@ object ReplicaVerificationTool extends Logging {
                           .describedAs("ms")
                           .ofType(classOf[java.lang.Long])
                           .defaultsTo(30 * 1000L)
+    val helpOpt = parser.accepts("help", "Print usage information.")

-    if (args.length == 0)
+    val options = parser.parse(args: _*)
+
+    if (args.length == 0 || options.has(helpOpt))
       CommandLineUtils.printUsageAndDie(parser, "Validate that all replicas for a set of topics have the same
data.")

-    val options = parser.parse(args: _*)
     CommandLineUtils.checkRequiredArgs(parser, options, brokerListOpt)
```

core/src/main/scala/kafka/tools/StreamsResetter.java

```
@@ -100,6 +100,7 @@ public class StreamsResetter {
     private static OptionSpecBuilder dryRunOption;
     private static OptionSpecBuilder executeOption;
     private static OptionSpec<String> commandConfigOption;
+    private static OptionSpecBuilder helpOption;

     private OptionSet options = null;
     private final List<String> allTopics = new LinkedList<>();
@@ -215,12 +216,16 @@ public class StreamsResetter {
             .describedAs("file name");
         executeOption = optionParser.accepts("execute", "Execute the command.");
         dryRunOption = optionParser.accepts("dry-run", "Display the actions that would be performed without
executing the reset commands.");
+        helpOption = optionParser.accepts("help", "Print usage information.");

         // TODO: deprecated in 1.0; can be removed eventually
         optionParser.accepts("zookeeper", "Zookeeper option is deprecated by bootstrap.servers, as the reset
tool would no longer access Zookeeper directly.");

         try {
             options = optionParser.parse(args);
+            if (args.length == 0 || options.has(helpOption)) {
+              CommandLineUtils.printUsageAndDie(optionParser, "Tool to resets the processing state of a Kafka
Streams application");
+            }
         } catch (final OptionException e) {
             printHelp(optionParser);
             throw e;
```

Compatibility, Deprecation, and Migration Plan

- *There won't be any change of current behavior. '--help' is a new option for CLI commands.*