**1) Mention what is Apache Kafka?**

Apache Kafka is a publish-subscribe messaging system developed by Apache written in Scala. It is a distributed, partitioned and replicated log service.

**2) Mention what is the traditional method of message transfer?**

The traditional method of message transfer includes two methods

•  **Queuing:** In a queuing, a pool of consumers may read message from the server and each message goes to one of them  
•  **Publish-Subscribe:** In this model, messages are broadcasted to all consumers  
Kafka caters single consumer abstraction that generalized both of the above- the consumer group.

**3) Mention what is the benefits of Apache Kafka over the traditional technique?**

Apache Kafka has following benefits above traditional messaging technique

•**Fast:** A single Kafka broker can serve thousands of clients by handling megabytes of reads and writes per second  
•**Scalable**: Data are partitioned and streamlined over a cluster of machines to enable larger data  
•**Durable:** Messages are persistent and is replicated within the cluster to prevent data loss  
•  **Distributed by Design:** It provides fault tolerance guarantees and durability

**4) Mention what is the meaning of broker in Kafka?**

In Kafka cluster, broker term is used to refer Server.

**5) Mention what is the maximum size of the message does Kafka server can receive?**

The maximum size of the message that Kafka server can receive is 1000000 bytes.

[2014-10-28_13-26-49](http://career.guru99.com/wp-content/uploads/2014/10/2014-10-28_13-26-49.png)

**6) Explain what is Zookeeper in Kafka? Can we use Kafka without Zookeeper?**

Zookeeper is an open source, high-performance co-ordination service used for distributed applications adapted by Kafka.

No, it is not possible to bye-pass Zookeeper and connect straight to the Kafka broker. Once the Zookeeper is down, it cannot serve client request.

•  Zookeeper is basically used to communicate between different nodes in a cluster  
•  In Kafka, it is used to commit offset, so if node fails in any case it can be retrieved from the previously committed offset  
•  Apart from this it also does other activities like leader detection, distributed synchronization, configuration management, identifies when a new node leaves or joins, the cluster, node status in real time, etc.

**7) Explain how message is consumed by consumer in Kafka?**

Transfer of messages in Kafka is done by using sendfile API. It enables the transfer of bytes from the socket to disk via kernel space saving copies and call between kernel user back to the kernel.

**8) Explain how you can improve the throughput of a remote consumer?**

If the consumer is located in a different data center from the broker, you may require to tune the socket buffer size to amortize the long network latency.

**9) Explain how you can get exactly once messaging from Kafka during data production?**

During data, production to get exactly once messaging from Kafka you have to follow two things **avoiding duplicates during data consumption** and **avoiding duplication during data production.**

Here are the two ways to get exactly one semantics while data production:

•  Avail a single writer per partition, every time you get a network error checks the last message in that partition to see if your last write succeeded  
•  In the message include a primary key (UUID or something) and de-duplicate on the consumer

**10) Explain how you can reduce churn in ISR? When does broker leave the ISR?**

ISR is a set of message replicas that are completely synced up with the leaders, in other word ISR has all messages that are committed. ISR should always include all replicas until there is a real failure. A replica will be dropped out of ISR if it deviates from the leader.

**11) Why replication is required in Kafka?**

Replication of message in Kafka ensures that any published message does not lose and can be consumed in case of machine error, program error or more common software upgrades.

**12) What does it indicate if replica stays out of ISR for a long time?**

If a replica remains out of ISR for an extended time, it indicates that the follower is unable to fetch data as fast as data accumulated at the leader.

**13) Mention what happens if the preferred replica is not in the ISR?**

If the preferred replica is not in the ISR, the controller will fail to move leadership to the preferred replica.

**14) Is it possible to get the message offset after producing?**

You cannot do that from a class that behaves as a producer like in most queue systems, its role is to fire and forget the messages. The broker will do the rest of the work like appropriate metadata handling with id’s, offsets, etc.

As a consumer of the message, you can get the offset from a Kafka broker. If you gaze in the **SimpleConsumer** class, you will notice it fetches **MultiFetchResponse** objects that include offsets as a list. In addition to that, when you iterate the Kafka Message, you will have **MessageAndOffset** objects that include both, the offset and the message sent.

op Answers to Kafka Interview Questions

**1. Compare Kafka & Flume**

|  |  |  |
| --- | --- | --- |
| **Criteria** | **Kafka** | **Flume** |
| Data flow | Pull | Push |
| Hadoop Integration | Loose | Tight |
| Functionality | Publish-subscribe model messaging system | System for data collection, aggregation & movement |

**2. Which are the elements of Kafka?**

The most important elements of Kafka:

* Topic – It is the bunch of similar kind of messages
* Producer – using this one can issue communications to the topic
* Consumer – it endures to a variety of topics and takes data from brokers.
* Brokers – this is the place where the issued messages are stored

**3. What role Zoo Keeper plays in a cluster of Kafka?**

Kafka is an open source system and also a distributed system is built to use Zookeeper. The basic responsibility of Zookeeper is to build coordination between [different nodes in a cluster](https://intellipaat.com/tutorial/oracle-dba-tutorial/real-application-clusters-rac/). Since Zookeeper works as periodically commit offset so that if any node fails, it will be used to recover from previously committed to offset. The zoo keeper is also responsible for configuration management, leader detection, detecting if any node leaves or joins the cluster, synchronization, etc.

**Get kafka Certification in just 6 Hours**

**GET CERTIFIED**

**4. What is Kafka?**

Kafka is a message divider project coded in Scala. Kafka is originally developed by LinkedIn and developed as an open sourced in early 2011. The purpose of the project is to achieve the best stand for conducting the real-time statistics nourishment.

Learn more about Kafka in this comprehensive [kafka Tutorial now](https://intellipaat.com/tutorial/big-data-and-hadoop-tutorial/the-ultimate-goal-of-this-tutorial/" \t "_blank)!

**5. Why do you think the replications are dangerous in Kafka?**

Duplication assures that issued messages which are available are absorbed in the case of any appliance mistake, plan fault or recurrent software promotions.

**6. What major role a Kafka Producer API plays?**

It is responsible for covering the two producers- kafka.producer.SyncProducer and the kafka.producer.async.AsyncProducer. The main aim is to disclose all the producer performance through a single API to the clients.

[**Wish to Learn Kafka? Click Here**](https://intellipaat.com/kafka-training-online/#course-content)

**7. Distinguish between the Kafka and Flume?**

Flume’s major use-case is to gulp down the data into Hadoop. The flume is incorporated with the [Hadoop’s monitoring system](https://intellipaat.com/tutorial/big-data-and-hadoop-tutorial/the-hadoop-module-high-level-architecture/), file formats, file system and utilities such as Morphlines. Flume’s design of sinks, sources and channels mean that with the aid of Flume one can shift data among other systems lithely, but the main feature is its Hadoop integration. The flume is the best option used when you have non-relational data sources if you have a long file to stream into the Hadoop.Kafka’s major use-case is a distributed publish- subscribe messaging system. Kafka is not developed specifically for Hadoop and using Kafka to read and write data to Hadoop is considerably trickier than it is in Flume. Kafka can be used when you particularly need a highly reliable and scalable enterprise messaging system to connect many multiple systems like Hadoop.

***For Additional Reading*** :- [Processing JSON Data in Real Time Streaming using Storm & Kafka](https://intellipaat.com/blog/processing-json-data-in-real-time-streaming-using-storm-kafka/)

**8. Describe partitioning key?**

Its role is to specify the target divider of the memo, within the producer. Usually, a hash-oriented divider concludes the divider ID according to the given factors. Consumers also use the tailored Partitions.

**9. Inside the manufacturer, when does the QueueFullException emerge?**

QueueFullException naturally happens when the manufacturer tries to propel communications at a speed which Broker can’t grip. Consumers need to insert sufficient brokers to collectively grip the amplified load since the Producer doesn’t block.

**Download Kafka Interview questions asked by top MNCs in 2017 ?**

Top of Form



Bottom of Form

**10. Can Kafka be utilized without Zookeeper?**

It is impossible to use [Kafka without Zookeeper](https://intellipaat.com/tutorial/kafka-tutorials/kafka-configuration/) because it is not feasible to go around Zookeeper and attach in a straight line to the server. If the Zookeeper is down for a number of causes, then we will not be able to serve any customer demand.

**11. Elaborate Kafka architecture.**

A cluster contains multiple brokers since it is a distributed system. Topic in the system will get divided into multiple partitions and each broker store one or more of those partitions so that multiple producers and consumers can publish and retrieve messages at the same time.

**12. How to start a Kafka server?**

Given that Kafka exercises Zookeeper, we have to start the Zookeeper’s server.

Learn more in this [Zookeeper Tutorial now](https://intellipaat.com/tutorial/hbase-tutorial/installation/). One can use the convince script packaged with Kafka to get a crude but effective single node Zookeeper instance> bin/zookeeper-server-start.shconfig/zookeeper.propertiesNow the Kafka server can start> bin/Kafka-server-start.shconfig/server.properties

**13. What are consumers or users?**

Kafka provides single consumer abstractions that discover both queuing and publish-subscribe Consumer Group. They tag themselves with a user group and every communication available on a topic is distributed to one user case within every promising user group. User instances are in disconnected process. We can determine the messaging model of the consumer based on the consumer groups.

* If all consumer instances have the same consumer set, then this works like a conventional queue adjusting load over the consumers.
* If all customer instances have dissimilar consumer groups, then this works like a publish-subscribe and all messages are transmitted to all the consumers.

**14. Describe an Offset?**

The messages in the partitions will be given a sequential ID number known as an offset, the offset will be used to identify each message in the partition uniquely. With the aid of Zookeeper Kafka stores the offsets of messages consumed for a specific topic and partition by this consumer group.

**15. What do you know about partitioning key?**

A partition key can be precise to point to the aimed division of a communication, in Kafka producer. Usually, a hash-oriented divider concludes the division id with the input and people uses modified divisions also.

**16. Why is Kafka technology significant to use?**

Kafka being distributed publish-subscribe system has the advantages as below.Fast: Kafka comprises of a broker and a single broker can serve thousands of clients by handling megabytes of reads and writes per second.Scalable: facts are partitioned and streamlined over a cluster of machines to enable large informationDurable: Messages are persistent and is replicated in the cluster to prevent record loss Distributed by Design: It provides fault tolerance guarantees and robust.

1. What is Kafka?

Wikipedia defines Kafka as “an open-source message broker project developed by the [Apache Software Foundation](https://en.wikipedia.org/wiki/Apache_Software_Foundation) written in Scala, where the design is heavily influenced by transaction logs”. It is essentially a distributed publish-subscribe messaging system.

2. List the various components in Kafka.

The four major components of Kafka are:

* Topic – a stream of messages belonging to the same type
* Producer – that can publish messages to a topic
* Brokers – a set of servers where the publishes messages are stored
* Consumer – that subscribes to various topics and pulls data from the brokers.

3. Explain the role of the offset.

Messages contained in the partitions are assigned a unique ID number that is called the offset. The role of the offset is to uniquely identify every message within the partition.

4. What is a Consumer Group?

Consumer Groups is a concept exclusive to Kafka.  Every Kafka consumer group consists of one or more consumers that jointly consume a set of subscribed topics.

5. What is the role of the ZooKeeper?

Kafka uses Zookeeper to store offsets of messages consumed for a specific topic and partition by a specific Consumer Group.

6. Is it possible to use Kafka without ZooKeeper?

No, it is not possible to bypass Zookeeper and connect directly to the Kafka server. If, for some reason, ZooKeeper is down, you cannot service any client request.

7. Explain the concept of Leader and Follower.

Every partition in Kafka has one server which plays the role of a Leader, and none or more servers that act as Followers. The Leader performs the task of all read and write requests for the partition, while the role of the Followers is to passively replicate the leader. In the event of the Leader failing, one of the Followers will take on the role of the Leader. This ensures load balancing of the server.

8. What roles do Replicas and the ISR play?

Replicas are essentially a list of nodes that replicate the log for a particular partition irrespective of whether they play the role of the Leader. On the other hand, ISR stands for In-Sync Replicas. It is essentially a set of message replicas that are synced to the leaders.

9. Why are Replications critical in Kafka?

Replication ensures that published messages are not lost and can be consumed in the event of any machine error, program error or frequent software upgrades.

10. If a Replica stays out of the ISR for a long time, what does it signify?

It means that the Follower is unable to fetch data as fast as data accumulated by the Leader.

11. What is the process for starting a Kafka server?

Since Kafka uses ZooKeeper, it is essential to initialize the ZooKeeper server, and then fire up the Kafka server.

* To start the ZooKeeper server: > bin/zookeeper-server-start.sh config/zookeeper.properties
* Next, to start the Kafka server: > bin/kafka-server-start.sh config/server.properties

12. How do you define a Partitioning Key?

Within the Producer, the role of a Partitioning Key is to indicate the destination partition of the message. By default, a hashing-based Partitioner is used to determine the partition ID given the key. Alternatively, users can also use customized Partitions.

13. In the Producer, when does QueueFullException occur?

QueueFullException typically occurs when the Producer attempts to send messages at a pace that the Broker cannot handle. Since the Producer doesn’t block, users will need to add enough brokers to collaboratively handle the increased load.

14. Explain the role of the Kafka Producer API.

The role of Kafka’s Producer API is to wrap the two producers – kafka.producer.SyncProducer and the kafka.producer.async.AsyncProducer. The goal is to expose all the producer functionality through a single API to the client.

15. What is the main difference between Kafka and Flume?

Even though both are used for real-time processing, Kafka is scalable and ensures message durability.

These are some of the frequently asked Apache Kafka interview questions with answers. You can brush up on your knowledge of Apache Kafka with [these](http://www.edureka.co/blog/category/apache-kafka/)blogs.

*Got a question for us? Please mention it in the comments section and we will get back to you.*

# **Hadoop Questions and Answers – Kafka with Hadoop-2**

This set of Interview Questions & Answers focuses on “Kafka”.

1. \_\_\_\_\_\_\_\_\_\_ provides the functionality of a messaging system.  
a) Oozie  
b) Kafka  
c) Lucene  
d) BigTop  
View Answer

Answer:b  
Explanation:Kafka is a distributed, partitioned, replicated commit log service.

2. Point out the correct statement :  
a) With kafka, more users, whether using SQL queries or BI applications, can interact with more data  
b) A topic is a category or feed name to which messages are published.  
c) For each topic, the Kafka cluster maintains a partitioned log  
d) None of the mentioned  
View Answer

Answer:b  
Explanation:Kafka is possible through a single repository and metadata store from source through analysis.

3. Kafka maintains feeds of messages in categories called :  
a) topics  
b) chunks  
c) domains  
d) messages  
View Answer

Answer:a  
Explanation:We’ll call processes that publish messages to a Kafka topic producers.

4. Kafka is run as a cluster comprised of one or more servers each of which is called :  
a) cTakes  
b) broker  
c) test  
d) None of the mentioned  
View Answer

Answer:b  
Explanation:We’ll call processes that subscribe to topics and process the feed of published messages consumers.

5. Point out the wrong statement :  
a) The Kafka cluster does not retain all published messages  
b) A single Kafka broker can handle hundreds of megabytes of reads and writes per second from thousands of clients  
c) Kafka is designed to allow a single cluster to serve as the central data backbone for a large organization  
d) Messages are persisted on disk and replicated within the cluster to prevent data loss  
View Answer

Answer:a  
Explanation:The Kafka cluster retains all published messages—whether or not they have been consumed—for a configurable period of time.

6. Communication between the clients and the servers is done with a simple, high-performance, language agnostic \_\_\_\_\_\_\_\_\_ protocol.  
a) IP  
b) TCP  
c) SMTP  
d) ICMP  
View Answer

Answer:b  
Explanation:Java client is provided for Kafka, but clients are available in many languages.

7. The only metadata retained on a per-consumer basis is the position of the consumer in the log, called :  
a) offset  
b) partition  
c) chunks  
d) All of the mentioned  
View Answer

Answer:a  
Explanation:offset is controlled by the consumer: normally a consumer will advance its offset linearly as it reads messages

8. Each kafka partition has one server which acts as the \_\_\_\_\_\_\_\_\_  
a) leaders  
b) followers  
c) staters  
d) All of the mentioned  
View Answer

Answer:a  
Explanation:Each partition is replicated across a configurable number of servers for fault tolerance.

9. \_\_\_\_\_\_\_\_\_ has stronger ordering guarantees than a traditional messaging system.  
a) kafka  
b) Slider  
c) Suz  
d) None of the mentioned  
View Answer

Answer:a  
Explanation:A traditional queue retains messages in-order on the server.

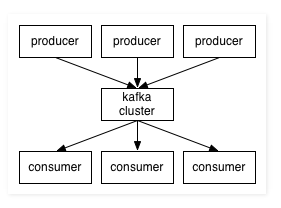
10. Kafka only provides a \_\_\_\_\_\_\_\_\_ order over messages within a partition.  
a) partial  
b) total  
c) 30%  
d) None of the mentioned  
View Answer

Answer:b  
Explanation:Per-partition ordering combined with the ability to partition data by key is sufficient for most applications.

## Apache Kafka Interview Questions

**What is Apache**[Kafka](http://en.wikipedia.org/wiki/Franz_Kafka)**?**

Apache Kafka is publish-subscribe messaging rethought as a distributed commit log. A high-throughput distributed messaging system. Kafka is a **general purpose** publish-subscribe model messaging system, which offers strong durability, scalability and fault-tolerance support. It is not specifically designed for Hadoop. Hadoop ecosystem is just be one of its possible consumers.



**Kafka Vs Flume**

Compared to Flume, Kafka wins on the its superb scalability and messsage durablity.

Kafka is very scalable. One of the key benefits of Kafka is that it is very easy to add large number of consumers without affecting performance and without down time. That’s because Kafka does not track which messages in the topic have been consumed by consumers. It simply keeps all messages in the topic within a configurable period. It is the consumers’ responsibility to do the tracking through offset.

In contrast, adding more consumers to Flume means changing the topology of Flume pipeline design, replicating the channel to deliver the messages to a new sink. It is not really a scalable solution when you have huge number of consumers. Also since the flume topology needs to be changed, it requires some down time.

Kafka’s scalability is also demonstrated by its ability to handle spike of the events. This is where Kakfa truly shines because it acts as a “shock absorber” between the producers and consumers. Kafka can handle events at 100k+ per second rate coming from producers. Because Kafka consumers pull data from the topic, different consumers can consume the messages at different pace. Kafka also supports different consumption model. You can have one consumer processing the messages at real-time and another consumer processing the messages in batch mode.

**Which components are used for stream flow of data?**

**Bolt :-** Bolts represent the processing logic unit in Storm. One can utilize bolts to do any kind of processing such as filtering, aggregating, joining, interacting with data stores, talking to external systems etc. Bolts can also emit tuples (data messages) for the subsequent bolts to process. Additionally, bolts are responsible to acknowledge the processing of tuples after they are done processing.

**Spout:-** Spouts represent the source of data in Storm. You can write spouts to read data from data sources such as database, distributed file systems, messaging frameworks etc. Spouts can broadly be classified into following –

**-Reliable –** These spouts have the capability to replay the tuples (a unit of data in data stream). This helps applications achieve ‘at least once message processing’ semantic as in case of failures, tuples can be replayed and processed again. Spouts for fetching the data from messaging frameworks are generally reliable as these frameworks provide the mechanism to replay the messages.

**-Unreliable –** These spouts don’t have the capability to replay the tuples. Once a tuple is emitted, it cannot be replayed irrespective of whether it was processed successfully or not. This type of spouts follow ‘at most once message processing’ semantic.

**Tuple:-** The tuple is the main data structure in Storm. A tuple is a named list of values, where each value can be any type. Tuples are dynamically typed — the types of the fields do not need to be declared. Tuples have helper methods like getInteger and getString to get field values without having to cast the result. Storm needs to know how to serialize all the values in a tuple. By default, Storm knows how to serialize the primitive types, strings, and byte arrays. If you want to use another type, you’ll need to implement and register a serializer for that type.

***Get through the interview bar with our selected interview questions for*** [*Apache Kafka*](http://tekslate.com/apache-kafka-training) ***enthusiasts***

**What are the key benefits of using Storm for Real Time Processing?**

Easy to operate: Operating storm is quiet easy

Real fast: It can process 100 messages per second per node

[Fault Tolerant](http://en.wikipedia.org/wiki/Fault-tolerant_design): It detects the fault automatically and re-starts the functional attributes

Reliable: It guarantees that each unit of data will be executed at least once or exactly once

[Scalable](http://en.wikipedia.org/wiki/Scalability): It runs across a cluster of machine

**Does Apache act as a Proxy server?**

Yes, It acts as proxy also by using the mod\_proxy module. This module implements a proxy, gateway or cache for Apache. It implements proxying capability for AJP13 ([Apache JServ Protocol](http://en.wikipedia.org/wiki/Apache_JServ_Protocol) version 1.3), FTP, CONNECT (for SSL),[HTTP/0.9](http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol), HTTP/1.0, and (since Apache 1.3.23) HTTP/1.1. The module can be configured to connect to other proxy modules for these and other protocols.

**While installing, why does Apache have three config files – srm.conf, access.conf and httpd.conf?**

The first two are remnants from the NCSA times, and generally you should be ok if you delete the first two, and stick with httpd.conf.

**What is ZeroMQ?**

ZeroMQ is “a library which extends the standard socket interfaces with features traditionally provided by specialized messaging middleware products”. Storm relies on ZeroMQ primarily for task-to-task communication in running Storm topologies.

**How many distinct layers are of Storm’s Codebase?**

There are three distinct layers to Storm’s codebase.

-First, Storm was designed from the very beginning to be compatible with multiple languages. Nimbus is a Thrift service and topologies are defined as Thrift structures. The usage of Thrift allows Storm to be used from any language.

-Second, all of Storm’s interfaces are specified as Java interfaces. So even though there’s a lot of Clojure in Storm’s implementation, all usage must go through the Java API. This means that every feature of Storm is always available via Java.

-Third, Storm’s implementation is largely in Clojure. Line-wise, Storm is about half Java code, half Clojure code. But Clojure is much more expressive, so in reality the great majority of the implementation logic is in Clojure.

**When do you call the cleanup method?**

The cleanup method is called when a Bolt is being shutdown and should cleanup any resources that were opened. There’s no guarantee that this method will be called on the cluster: For instance, if the machine the task is running on blows up, there’s no way to invoke the method. The cleanup method is intended when you run topologies in local mode (where a Storm cluster is simulated in process), and you want to be able to run and kill many topologies without suffering any resource leaks.

**How can we kill a topology?**

To kill a topology, simply run:

**storm kill {stormname}**

Give the same name to storm kill as you used when submitting the topology. Storm won’t kill the topology immediately. Instead, it deactivates all the spouts so that they don’t emit any more tuples, and then Storm waits Config.TOPOLOGY\_MESSAGE\_TIMEOUT\_SECS seconds before destroying all the workers. This gives the topology enough time to complete any tuples it was processing when it got killed.

***Learn Apache Kafka by Tekslate - Fastest growing sector in the industry.***

***Explore Online Apache Kafka Training and course is aligned with industry needs & developed by industry veterans.***

***Tekslate will turn you into*** [*Apache Kafka Expert*](http://tekslate.com/tutorials/apache-kafka-tutorials/)***.***

**What is combiner Aggregator?**

A Combiner Aggregator is used to combine a set of tuples into a single field. It has the following signature:

public interface CombinerAggregator {

T init (TridentTuple tuple);

T combine(T val1, T val2);

T zero();

}

Storm calls the init() method with each tuple, and then repeatedly calls the combine()method until the partition is processed. The values passed into the combine() method are partial aggregations, the result of combining the values returned by calls to init().

**Is it necessary to kill the topology while updating the running topology?**

Yes, to update a running topology, the only option currently is to kill the current topology and resubmit a new one. A planned feature is to implement a Storm swap command that swaps a running topology with a new one, ensuring minimal downtime and no chance of both topologies processing tuples at the same time.

**Explain how to write the Output into a file using Storm?**

In Spout, when you are reading file, make FileReader object in Open() method, as such that time it initializes the reader object for worker node. And use that object in nextTuple() method.

**Mention what is the difference between Apache Kafka and Apache Storm?**

–**Apach Kafeka:** It is a distributed and robust messaging system that can handle huge amount of data and allows passage of messages from one end-point to another.

–**Apache Storm**: It is a real time message processing system, and you can edit or manipulate data in real time. Apache storm pulls the data from Kafka and applies some required manipulation.

**Explain when using field grouping in storm, is there any time-out or limit to known field values?**

Field grouping in storm uses a mod hash function to decide which task to send a tuple, ensuring which task will be processed in the correct order. For that, you don’t require any cache. So, there is no time-out or limit to known field values.

**In which folder are Java Applications stored in Apache?**

Java applications are not stored in Apache, it can be only connected to a other [Java webapp](http://tekslate.com/java-with-jquery/)hosting [webserver](http://tekslate.com/interview-questions-on-websphere-process-server/) using the mod\_jk connector.

What is mod\_vhost\_alias?

This module creates dynamically configured virtual hosts, by allowing the IP address and/or the Host: header of the HTTP request to be used as part of the pathname to determine what files to serve. This allows for easy use of a huge number of virtual hosts with similar configurations.

What is struct and explain its purpose?

A struts is a open source framework for creating a[Java web applications.](http://tekslate.com/tutorials/java-webdynpro-tutorials/)

**Tell me Is running apache as a root is a security risk?**

No.root process opens port 80, but never listens to it, so no user will actually enter the site with root rights. If  
you kill the root process, you will see the other kids disappear as well.

In Kafka:

* Kafka maintains the messages feeds in categories called Topics
* Process which publish messages to a Kafka topic is called as Producer
* Process which consumes messages from a Kafka topic is called as Consumer
* All the above components are coordinated and managed via Kafka Broker. This broker or cluster could be set of physical servers.

Image Credit: Kafka Wiki Documentation

**What I have observed in Kafka:**

* **Use of both push and pull mechanism in pub-sub** – Producer pushes the message to the broker and Consumers pulls the messages from the broker. It’s not like whenever message is published, consumer will receive automatically. Here Consumer has to pull the message from the Kafka broker. Few overheads solved here – as Consumer pulls the message from the topic, Consumer would have knowledge on the message offset (what is the next record that I need to pull). One more good news is – Kafka Consumer API will handle these message offset positioning internally including which copy of topic it needs to use for pull, what to do in case of broker failure scenarios etc. You can chose to override the consumer API methods if needed
* **High Availability:**High Availability of the Kafka broker cluster will be achieved based on the replications that you maintain for your topic. This is configurable parameter when you create a topic within the cluster. More number of replication could impact performance to some extent. Kafka also uses Apache zookeeper for cluster coordination and so high availability needs to be configured for zookeeper as well. We can see few recommendation on having at least three node cluster for zookeeper
* **Compression:**As Kafka involves movement of huge volume of messages across the network and to resolve any network bandwidth bottleneck issues, Kafka allows compression in the topics. It supports gzip and snappy compression codec. This is just configurable parameter in Kafka
* **Batching:**If you look into Kafka design – messages are written to file. Each message causes some IO within the broker. If the message size is going to be small and huge volume of messages comes into the cluster – there would be spike in IO which could impact performance. To avoid that – if you need to batch the messages within the producer thread, publish the message to broker only when you reach the configured size or no of messages – this could be easily achieved through configurable parameter
* **Kafka API’s:**Kafka provides set of API for Producer & Consumer. We have two API’s for Producer and two API’s for Consumer. API’s are available in several languages like Java, Python, C, C++, .net, Clojure, Ruby, Go, Node JS etc (List is huge). You need to know at least one of the languages to implement the Kafka Producer & Consumer. But before choosing the language, try to see what Kafka functionalities can be accomplished using that language (few functionalities might not be supported), do you have enough support in case if you run into trouble while wiring the API’s Kafka also has Hadoop Consumer API to aggregate and load data into Hadoop, this will spin off multiple mappers to pull data from Kafka cluster in parallel.
* **Parallelism:**Parallelism in Kafka Producer / Consumer could be achieved by defining partitions in the message topics and creating multiple threads for produce & consume operations. By running multiple producer threads – publishing messages to the topic in parallel – we could achieve good scalability in processing of messages. You can refer to interesting article from LinkedIn Benchmark study on Apache Kafka [here](https://engineering.linkedin.com/kafka/benchmarking-apache-kafka-2-million-writes-second-three-cheap-machines)
* **Partitioning:**As quoted above – Topic could be partitioned to allow greater scalability in processing messages. Partitioning could be customized – it could be implemented in a separate class and can be plugged in using the partitioner.class config parameter. Again important thing to note here – Message ordering will be guaranteed only within the topic partition, not across the topic partition
* **Synchronous / Asynchronous:**When you need to publish message to the Kafka cluster, you might need acknowledgement from broker or you might not. Remember broker is again set of servers (several replications) – you have to choose whether you need acknowledgement from one broker or all the brokers. If you need acknowledgement from all the brokers which maintains the copy of your message – it will have performance impact, here it could be enough if we just have one acknowledgement (one broker committed your message). If you want to build fire and forget kind of publish – you can use Asynchronous mode in Producer. Again this is just a configuration in the producer
* **Security:**Current version of Kafka 0.8 does not provide any security authentication methods in their API’s. Security authentication measures like role based on the message topic are considered as essential for any enterprise. Don’t worry – next version of Kafka will address the security authentication. As of now – security measures are implemented around Kafka layer by the community users
* **Monitoring:**Kafka emits lot of metrics which can be browsed with any monitoring solution which uses JMX. In the community, could see that monitoring could be achieved by using Ganglia / Graphite. I haven’t tried this so far. With Kafka – you might need to have attention of Garbage Collection cycles, CPU, IO, message transfer rate, broker state change etc. Also good level of attention should be there for zookeeper logs as well
* **Administration & Utilities:**Most of the Kafka administration work is done using Kafka Utilities. There is no UI interface provided from Kafka though there are few projects in the community which could provide UI kind of interface for broker cluster. Apart from that – lot of utilities are available to mirror the Kafka cluster across data center, work with zookeeper offsets, JMX tool, migration tool etc.