**Ans 1 : High availability of Namenode**

* Hadoop 2.0 boasts of improved scalability and availability of the system via a set ofbundled features that represent a generational swing in the Hadoop Architecture with the introduction of [YARN](https://www.dezyre.com/article/hadoop-2-0-yarn-framework-the-gateway-to-easier-programming-for-hadoop-users/84) .
* Hadoop 2.0 also introduces the solution to the much awaited High Availability problem.
* Hadoop 2.0 overcomes this SPOF shortcoming by providing support for multiple NameNodes. It introduces Hadoop 2.0 High Availability feature that brings in an extra NameNode (Passive Standby NameNode) to the Hadoop Architecture which is configured for automatic failover.
* The main motive of the Hadoop 2.0 High Availability project is to render availability to big data applications 24/7 by deploying 2  Hadoop NameNodes –One in active configuration and the other is the Standby Node in passive configuration.
* Earlier there was one Hadoop NameNode for maintaining the tree hierarchy of the HDFS files and tracking the data storage in the cluster. Hadoop 2.0 High Availability allows users to configure Hadoop clusters with uncalled- for NameNodes so as to eliminate the probability of SPOF in a given Hadoop cluster. The Hadoop Configuration capability allows users to build clusters horizontally with several NameNodes which can operate autonomously through a common data storage pool, thereby, offering better computing scalability when compared to Hadoop 1.0
* With Hadoop 2.0, Hadoop architecture is now configured in a manner that it supports automated failover with complete stack resiliency and a hot Standby NameNode.
* With HDP 2.0 High Availability, the complete Hadoop Stack i.e. HBase, Pig, Hive, MapReduce, Oozie are equipped to tackle the NameNode failure problem- without having to lose the job progress or any related data. Thus, any critical long running jobs that are scheduled to be completed at a specific time will not be affected by the NameNode failure.

**Ans 2: Check pointing and how it is useful**

* Hadoop Distributed File System (HDFS) is a journaled file system, where new changes to files in HDFS are captured in an edit log that’s stored on the NameNode in a file named. Periodically, when the file reaches a certain threshold or after a certain period has elapsed, the journaled entries need to be committed to the master file.
* Checkpointing services for a Hadoop cluster are handled by one of four possible daemons, which need to run on their own dedicated master node alongside the NameNode daemon’s master node:
* Secondary NameNode
* Checkpoint Node
* Backup Node
* Standby NameNode
* The following steps describe the checkpointing process as it’s carried out by the NameNode and the checkpointing service (note that four possible daemons can be used for checkpointing):
* When it’s time to perform the checkpoint, the NameNode creates a new file to accept the journaled file system changes.

It names the new file.

* As a result, the file accepts no further changes and is copied to the checkpointing service, along with the file.
* The checkpointing service merges these two files, creating a file named.
* The checkpointing service copies the file to the NameNode.
* The NameNode overwrites the file with.
* The NameNode renames the file to.

**Ans 3 : HDFS federation**

* Hadoop federation separates the **namespace** **layer** and **storage layer**. It enables the block storage layer. It also expands the architecture of an existing HDFS cluster to allow new implementations and use cases. The current HDFS architecture has two layers –
* **Namespace –** This layer manages files, directories and blocks. This layer supports the basic file system operations e.g. listing of files, creation of files, modification of files and deletion of files and folders.
* **Block Storage –**This layer has two parts –
  + **Block Management** This manages the datanodes in the cluster and provides operations like creation, deletion, modification and search. It also takes care of the replication management.
  + **Physical Storage** This stores the blocks and provides access for read or write operations.
* Hadoop federation allows scaling the name service horizontally. It uses several namenodes or namespaces which are independent of each other. These independent namenodes are **federated** i.e. they don’t require **inter coordination**.

Benefits of Hadoop Federation:

Hadoop federation comes up with some advantages and benefits which are listed as under –

* **Scalability and Isolation –**Multiple namenodes horizontally scales up in the file system namespace. This actually separates namespace volumes for users and categories of application and provides an absolute isolation.
* **Generic Storage Service –**The block level pool abstraction allows the architecture to build new file systems on top of block storage. We can easily build new applications on the block storage layer without using the file system interface. Customized categories of block pool can also be built which are different from the default block pool.
* **Simple Design –**Namenodes and namespaces are independent of each other. There is hardly any scenario which requires changing the existing name nodes. Each name node is built to be robust. Federation is also backward compatible. It easily integrates with the existing single node deployments which work without any configuration changes.

**Ans 4 :**

* Core-site.xml
* HDFS-site.xml
* YARN-site.xml
* Xml

### **In site.xml File**

In Hadoop, default replication factor is 3 but we can override that property by making replication factor as 1 by explicitly configuring the property in hdfs-site.xml.