

# Apache Hadoop 3.0.0

Apache Hadoop 3.0.0 incorporates a number of significant enhancements over the previous major release line (hadoop-2.x).

This release is generally available (GA), meaning that it represents a point of API stability and quality that we consider production-ready.

## Overview

Users are encouraged to read the full set of release notes. This page provides an overview of the major changes.

### Minimum required Java version increased from Java 7 to Java 8

All Hadoop JARs are now compiled targeting a runtime version of Java 8. Users still using Java 7 or below must upgrade to Java 8.

### Support for erasure coding in HDFS

Erasur coding is a method for durably storing data with significant space savings compared to replication. Standard encodings like Reed-Solomon (10,4) have a 1.4x space overhead, compared to the 3x overhead of standard HDFS replication.

Since erasure coding imposes additional overhead during reconstruction and performs mostly remote reads, it has traditionally been used for storing colder, less frequently accessed data. Users should consider the network and CPU overheads of erasure coding when deploying this feature.

More details are available in the [HDFS Erasure Coding](#) documentation.

### YARN Timeline Service v.2


We are introducing an early preview (alpha 2) of a major revision of YARN Timeline Service: v.2. YARN Timeline Service v.2 addresses two major challenges: improving scalability and reliability of Timeline Service, and enhancing usability by introducing flows and aggregation.

YARN Timeline Service v.2 alpha 2 is provided so that users and developers can test it and provide feedback and suggestions for making it a ready replacement for Timeline Service v.1.x. It should be used only in a test capacity.

More details are available in the [YARN Timeline Service v.2](#) documentation.

### Shell script rewrite

The Hadoop shell scripts have been rewritten to fix many long-standing bugs and include some new features. While an eye has been kept towards compatibility, some changes may break existing installations.


Incompatible changes are documented in the release notes, with related discussion on [HADOOP-9902](#) .

More details are available in the [Unix Shell Guide](#) documentation. Power users will also be pleased by the [Unix Shell API](#) documentation, which describes much of the new functionality, particularly related to extensibility.

### Shaded client jars

The `hadoop-client` Maven artifact available in 2.x releases pulls Hadoop's transitive dependencies onto a Hadoop

application's classpath. This can be problematic if the versions of these transitive dependencies conflict with the versions used by the application.

[HADOOP-11804](#)  adds new `hadoop-client-api` and `hadoop-client-runtime` artifacts that shade Hadoop's dependencies into a single jar. This avoids leaking Hadoop's dependencies onto the application's classpath.

## Support for Opportunistic Containers and Distributed Scheduling.

A notion of `ExecutionType` has been introduced, whereby Applications can now request for containers with an execution type of `Opportunistic`. Containers of this type can be dispatched for execution at an NM even if there are no resources available at the moment of scheduling. In such a case, these containers will be queued at the NM, waiting for resources to be available for it to start. Opportunistic containers are of lower priority than the default `Guaranteed` containers and are therefore preempted, if needed, to make room for `Guaranteed` containers. This should improve cluster utilization.

Opportunistic containers are by default allocated by the central RM, but support has also been added to allow opportunistic containers to be allocated by a distributed scheduler which is implemented as an `AMRMProtocol` interceptor.

Please see [documentation](#) for more details.

## MapReduce task-level native optimization

MapReduce has added support for a native implementation of the map output collector. For shuffle-intensive jobs, this can lead to a performance improvement of 30% or more.

See the release notes for [MAPREDUCE-2841](#)  for more detail.

## Support for more than 2 NameNodes.



The initial implementation of HDFS NameNode high-availability provided for a single active NameNode and a single Standby NameNode. By replicating edits to a quorum of three `JournalNodes`, this architecture is able to tolerate the failure of any one node in the system.

However, some deployments require higher degrees of fault-tolerance. This is enabled by this new feature, which allows users to run multiple standby NameNodes. For instance, by configuring three NameNodes and five `JournalNodes`, the cluster is able to tolerate the failure of two nodes rather than just one.

The [HDFS high-availability documentation](#) has been updated with instructions on how to configure more than two NameNodes.

## Default ports of multiple services have been changed.

Previously, the default ports of multiple Hadoop services were in the Linux ephemeral port range (32768-61000). This meant that at startup, services would sometimes fail to bind to the port due to a conflict with another application.

These conflicting ports have been moved out of the ephemeral range, affecting the NameNode, Secondary NameNode, DataNode, and KMS. Our documentation has been updated appropriately, but see the release notes for [HDFS-9427](#)  and [HADOOP-12811](#)  for a list of port changes.

## Support for Microsoft Azure Data Lake and Aliyun Object Storage System filesystem connectors

Hadoop now supports integration with Microsoft Azure Data Lake and Aliyun Object Storage System as alternative Hadoop-compatible filesystems.


## Intra-datanode balancer


A single DataNode manages multiple disks. During normal write operation, disks will be filled up evenly. However, adding or replacing disks can lead to significant skew within a DataNode. This situation is not handled by the existing HDFS balancer, which concerns itself with inter-, not intra-, DN skew.

This situation is handled by the new intra-DataNode balancing functionality, which is invoked via the `hdfs diskbalancer` CLI. See the disk balancer section in the [HDFS Commands Guide](#) for more information.


## Reworked daemon and task heap management

A series of changes have been made to heap management for Hadoop daemons as well as MapReduce tasks.

[HADOOP-10950](#)  introduces new methods for configuring daemon heap sizes. Notably, auto-tuning is now possible based on the memory size of the host, and the `HADOOP_HEAPSIZE` variable has been deprecated. See the full release notes of HADOOP-10950 for more detail.

[MAPREDUCE-5785](#)  simplifies the configuration of map and reduce task heap sizes, so the desired heap size no longer needs to be specified in both the task configuration and as a Java option. Existing configs that already specify both are not affected by this change. See the full release notes of MAPREDUCE-5785 for more details.

## S3Guard: Consistency and Metadata Caching for the S3A filesystem client

[HADOOP-13345](#)  adds an optional feature to the S3A client of Amazon S3 storage: the ability to use a DynamoDB table as a fast and consistent store of file and directory metadata.

See [S3Guard](#) for more details.

## HDFS Router-Based Federation

HDFS Router-Based Federation adds a RPC routing layer that provides a federated view of multiple HDFS namespaces. This is similar to the existing [ViewFs](#)) and [HDFS Federation](#) functionality, except the mount table is managed on the server-side by the routing layer rather than on the client. This simplifies access to a federated cluster for existing HDFS clients.

See [HDFS-10467](#)  and the HDFS Router-based Federation [documentation](#) for more details.

## API-based configuration of Capacity Scheduler queue configuration

The OrgQueue extension to the capacity scheduler provides a programmatic way to change configurations by providing a REST API that users can call to modify queue configurations. This enables automation of queue configuration management by administrators in the queue's `administer_queue` ACL.

See [YARN-5734](#)  and the [Capacity Scheduler documentation](#) for more information.

## YARN Resource Types

The YARN resource model has been generalized to support user-defined countable resource types beyond CPU and memory. For instance, the cluster administrator could define resources like GPUs, software licenses, or locally-attached storage. YARN tasks can then be scheduled based on the availability of these resources.

See [YARN-3926](#)  and the [YARN resource model documentation](#) for more information.

# Getting Started

The Hadoop documentation includes the information you need to get started using Hadoop. Begin with the [Single Node Setup](#) which shows you how to set up a single-node Hadoop installation. Then move on to the [Cluster Setup](#) to learn how to set up a multi-node Hadoop installation.