YARN Appstore

Design Document

Version 1.3
August 30, 2017

1 Overview **3**
1.1 Requirement Description 3
1.2 Solution Summary 3
1.3 Solution Differentiation 3

2 External **4**
2.1 Container based cloud provision system 4
2.2 Docker Image repository on HDFS 4
2.3 Docker configuration to pull image from private registry 4
2.4 User interface for searchable YARN applications 5
2.6 Application data are written to HDFS through NFS Client -> NFS Gateway 5
2.7 Application Status are reported on Ambari View 6
2.8 Security 6

3 Internal **7**
3.1 YARN Appstore Application Catalog 7
3.1.1 Docker Registry 8
3.2 Appstore User Interface 8
3.3 YARN Native Service Integration 8
3.3.1 Nodemanager UI Integration 8
3.4 File System Support 9
3.5 Security 9
3.5.1 YARN Application Registry Access Control 9
3.5.2 Docker Container UID/GID 9
3.5.3 NFS Gateway Locality Support 10

YARN Appstore

1 Overview

1.1 Requirement Description

Hadoop cluster has been designed to run analytic workload. YARN resource manager is limited to run Hadoop optimized applications. New technology such as machine learning and cloud data warehouse can benefit from Hadoop resilience without invest additional hardware and administration skillsets. This design document describes the approach to improve Hadoop eco-system for enabling docker containers to run on Hadoop cluster with minimum changes to daily administration of Hadoop clusters.

1.2 Solution Summary

YARN Appstore will be a new sub-project devoted to catalog and index Yarnfiles created by Yarn native service. From YARN Appstore, user can launch multi-tiered docker containers on Hadoop cluster. The resource utilized by docker containers will be computed by YARN resource manager to ensure organization enforced limit is respected in the available resource pools.

1.3 Solution Differentiation

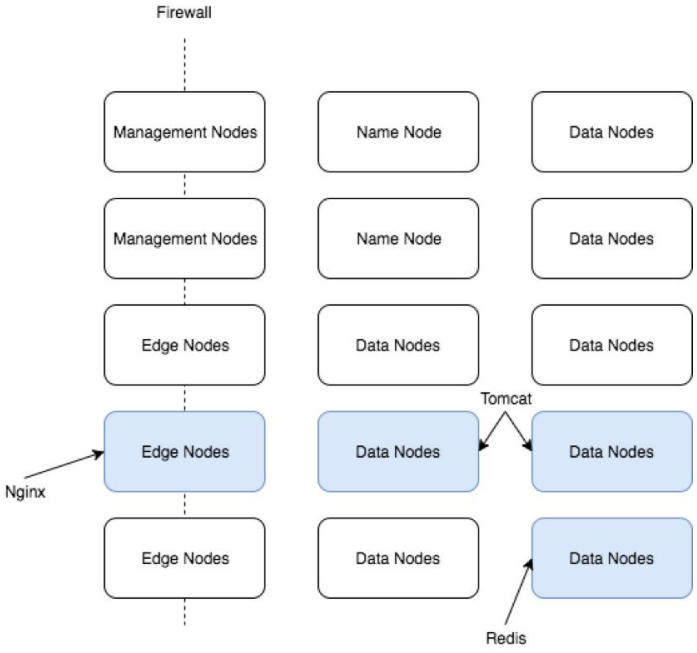
YARN Appstore may appear to look like existing technology such as Kubernetes or Kitematic. The common problem with existing technology is lack of persistence storage which is offered by HDFS for YARN native services. When combined YARN docker support, HDFS and YARN Appstore, Hadoop can enable highly available infrastructure to deploy micro services at scale that is difficult to achieve with Kubernetes or Kitematic.

In the initial implementation, YARN Appstore will only work on Linux platform. YARN Appstore is target as platform as a service instead of infrastructure as a service. The advantage of YARN Appstore is to create isolation between system framework and application framework with efficient elastic parallelization in Hadoop environment. YARN Appstore reduces development cost for applications and provide light-weight approach to manage large scale applications.

2 External

2.1 Container based cloud provision system

Docker container can be spawn on any node manager and use nodes label to decide if the node should be an edge node or private node. Preconfigured firewall helps to isolate private network traffic from public network, and reduce the repetitive system and network configuration required in traditional system management.



2.2 Docker Image repository on HDFS

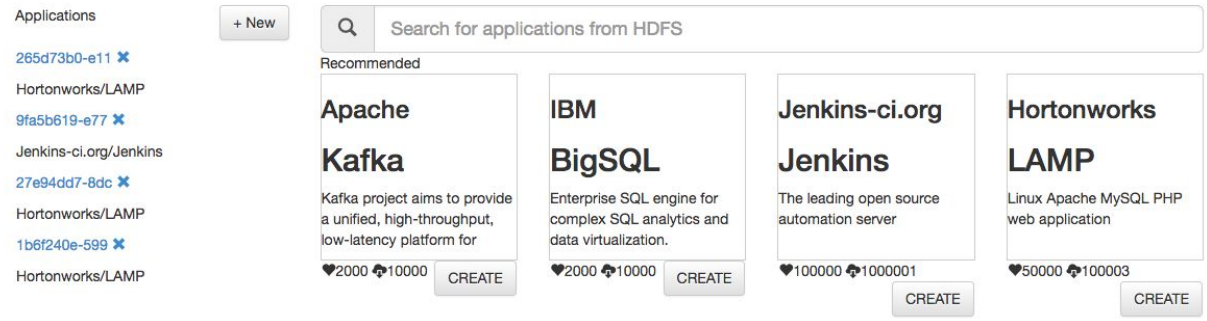
Docker Trusted Registry supports drivers to store docker images on various storage system. Docker images are immutable objects and this makes docker image storage on HDFS as an attractive alternative to S3 or local file system to improve resilience of docker images.

2.3 Docker configuration to pull image from private registry

Docker can pull image from private docker registry. The image only needs to be tagged with private registry location to enable uploading of private docker image. The syntax looks like:

```
docker tag [hostname]:[port]/[image_name]:[version]
docker push [hostname]:[port]/[image_name]:[version]
```

2.4 User interface for searchable YARN applications



A list of glossary items are:

Application Store - A central hub of applications categorized in YARN Appstore. Information about YARN applications are stored in Solr for fast searching of available applications.

Application Store Entry - Individual application information that describes the application, the docker components comprised the application.

Application List - List of deployed application.

Application Detail - Information about the per deployed instance of the application. This contains both the configuration of the application, and running status of the application. Configuration includes resource, and environment information. Status includes container ID, physical hostname and state of the deployed application.

Application Store lists recommended applications, and provide a SOLR based text box for finding applications from YARN Appstore. Application can be deployed to Hadoop cluster with a few click of buttons.

2.6 Application data are written to HDFS through NFS Client -> NFS Gateway

Docker can mount additional volume using –v flag. The data directories mounted through NFS Client can be exposed to container to write sequential data directly to HDFS.

2.7 Application Status are reported on Ambari View

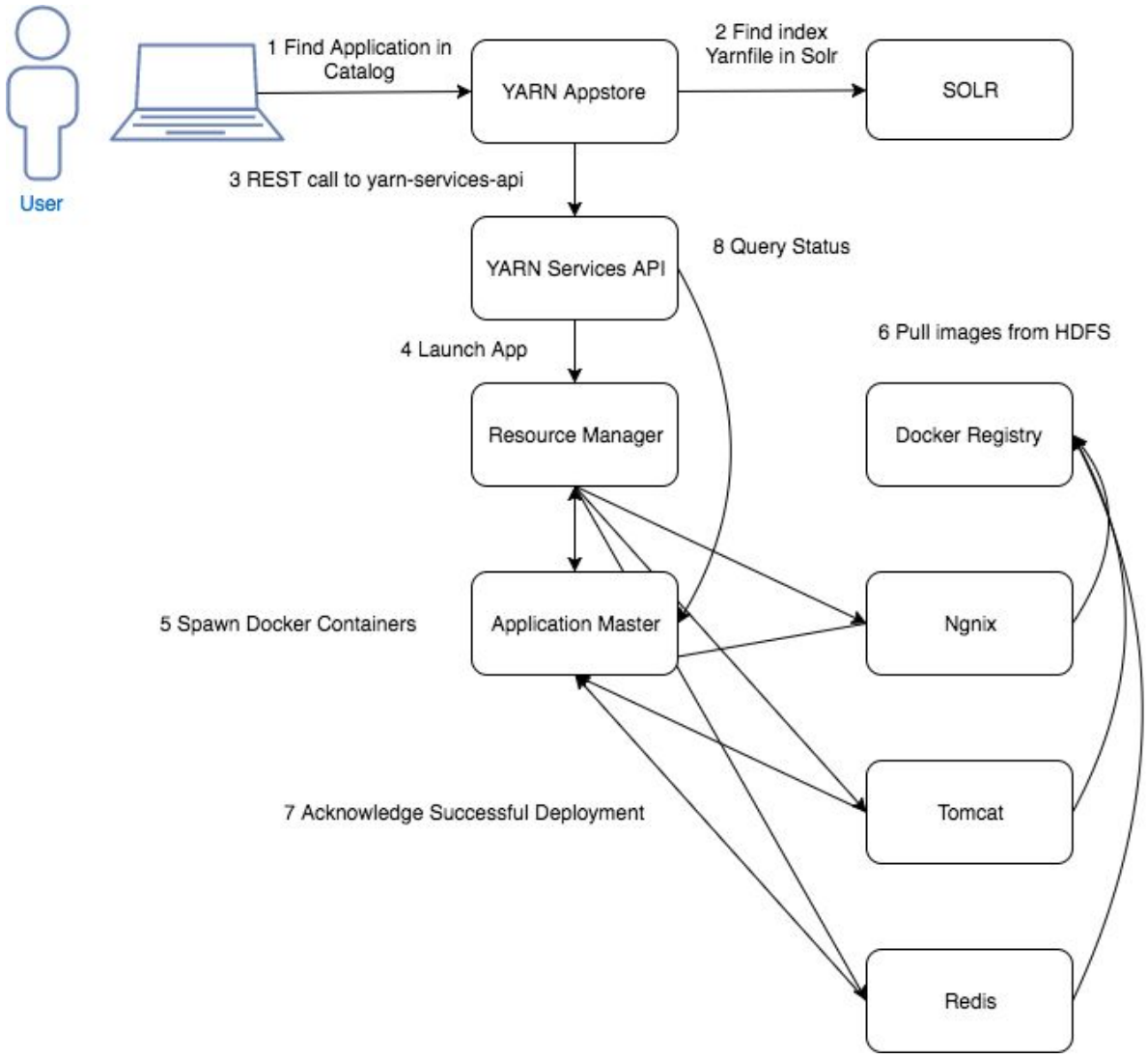
Status	Running
Started At	2017-07-11T17:23:47Z
Volumes	/hdfs/path1
Block Device IO	
Read Bps	1000MB/s
Write Bps	48MB/s
Read IOps	1024
Write IOps	23
Memory	16GB
Swap	10MB
CPU count	2
CPU Percent	50%

Docker inspect provides details of container metrics. Information such as CPU count, block IO device throughput and Memory Swappiness can be reported to Ambari View.

2.8 Security

Docker images will be running in non-privileged mode only. The file system access credential maps directly to the user who spawn container applications. Kerberos credential is presented by NFS gateway to namenode to verify the service user. OS will verify the container user using standard Linux file system access.

3 Internal



3.1 YARN Appstore Application Catalog

A Searchable catalog can be developed to find available YARN Assembly applications hosted in HDFS. YARN Assembly applications are descriptor files that describe Docker images, and list of parameters to set for docker images. This REST API server provides detail information about the image name, version number, description, number of pulls, favors, and last updated status. Any YARN Assembly application uploaded to Application Catalog can be deployed by cluster users.

3.1.1 Docker Registry

Docker Inc provides Docker Registry available for third party to modify and contribute. Docker registry can be used to deposit private docker images on HDFS, and use NFS gateway to proxy docker images to Docker Registry. This enables to serve docker images from HDFS without code modification to docker registry.

Run docker registry:

```
docker run -d -p 5000:5000 --name registry -v
/mnt/hdfs/apps/docker:/var/lib/registry registry:2
```

The registry storage can be mounted from NFS mount point. See section 3.4 for mounting /mnt/hdfs on all nodes.

3.2 Appstore User Interface

User Interface can be built into Ambari View to provide end user access for deploying docker container applications. The standard features are:

1. Search for Docker images
2. Deploy Docker containers
3. Delete Docker containers
4. Monitor running instance of Docker containers

3.3 YARN Native Service Integration

In Hadoop 3, YARN descriptor files called Yarnfile describes the relationship between docker containers to deploy in Hadoop environment. YARN Appstore can register Yarnfile with YARN Appstore to provide catalog for indexing YARN applications. YARN Appstore utilize Yarn services API REST API to deploy, check status of application deployed in YARN.

3.3.1 Nodemanager UI Integration

YARN UI provide the real time information about the deployment of the docker containers. YARN Appstore can leverage YARN internal API to obtain real time statics about the containers to provide a user friendly view of the running status. Alternatively, YARN Appstore can redirect

to YARN UI for real time status but this is less secure because more firewall ports needs to be open up to display container information.

3.4 File System Support

When HDFS is exposed with NFS like mount points, each datanode can be configured to use NFS Gateway, and use FUSE client library to expose HDFS as a local mount point. Docker container does not require to include HDFS client libraries. This improves ability to integrate with other non-Java based applications. Docker container can be started with -v flag to map local mount point into container. This provides container to have ability to write data directly to HDFS. Ambari integration with NFS Gateway can modify /etc/fstab to include NFS mount point to include local NFS Gateway.

Local mount NFS proxy on all nodes:

```
mount -t nfs -o vers=3,proto=tcp,nolock,noacl,sync `hostname`:/
/mnt/hdfs
```

3.5 Security

3.5.1 YARN Application Registry Access Control

Ranger can be interface with Application Catalog to provide access control to who has authority to update or remove docker images from HDFS.

3.5.2 Docker Container UID/GID

Docker daemon control is managed through UNIX socket. When securing Docker daemon, only YARN will interact with Docker daemon control interface. Docker container is restricted to non-root user by passing -u UID:GID parameter to docker daemon control interface. This will ensure the program running in docker environment is running as designated user. This change has been proposed in YARN-4266. There should be no expose of docker command line internals to outside world to prevent security mistakes. User defined volumes will be checked for input verification before mount points are mapped to docker images.
User process inside docker container maps to UID/GID of the host operating system. Docker container launch command includes -u=$(id -u $(whoami)):$(id -g $(whoami)), this enables the

process maps directly to the same UID/GID of host operating system. This is required to ensure HDFS volume receives the same user/group membership for file system access.

3.5.3 NFS Gateway Locality Support

NFS Kerberos configuration can expose NFS export to read/writeable by local host only. NFS Gateway will be exported to all nodes, and each datanode can read/write to its own NFS Gateway. Ambari NFS Gateway configuration can be improved to reduce security exposure to unknown hosts. In hdfs-site.xml:

```
<property>
  <name>dfs.nfs.exports.allowed.hosts</name>
  <value>127.0.0.1 rw</value>
</property>
```