Support Interactive Docker Shell to running Containers

Zian Chen
with input from: Eric Yang, Vinod Vavilapalli, Wangda Tan

Motivation

Debugging distributed application can be challenging on Hadoop. Hadoop provide limited debugging ability through application log files. One of the most frequently requested feature is to provide interactive shell to assist real time debugging. This feature is inspired by docker exec to provide ability to run arbitrary commands in docker container.

Requirements

The intention for this feature is to allow user type command to interactive with Docker Container just like they do things in terminal. But the underlying strategy is through web protocol to build a

bi-directional pipeline so that client and server can send/receive messages from both sides. I will explain what the client, server refer to and how the whole system works in architecture design.

In order to make interactive shell easy to use, we need to achieve below requirements,

Support both CLI and Web front UI use case

Since the user of this feature might be system admin which familiar with yarn command line interface or end user from other backgrounds (like data scientists, etc) who might be more comfortable maintain their containers through GUI, we need to support both of these scenarios while maintaining back-end server relatively simple.

Prevent any security risk happen through this interface

The obvious concern for establishing this interactive shell is security. Docker container inherits unix process privileges. Shell provided through docker exec is granted the same privileges when the docker container is started. User is limited to access inside the container. Authentication is done by Kerberos authentication to login to remote shell. The network communication is encrypted by TLS protocol to keep the connection secured. User is authorized to login to the container with Kerberos login credential.

Design

Container-executor can interface with docker exec to debug or analyze docker containers while the application is running. It would be nice to support an API to invoke docker exec to perform UNIX commands and report back the output to application master. Application master can distribute and aggregate execution of the commands to record in the application master log file.

To address those requirements mentioned above, we will use WebSocket to bring up a servlet in the Nodemanager web server and use AuthenticationFilter to ensure the security. In this section, we will give the overall architecture design first, then give implementation details regarding our proposed changes.

Architecture Overview

The architecture of the system is shown in Figure 1.

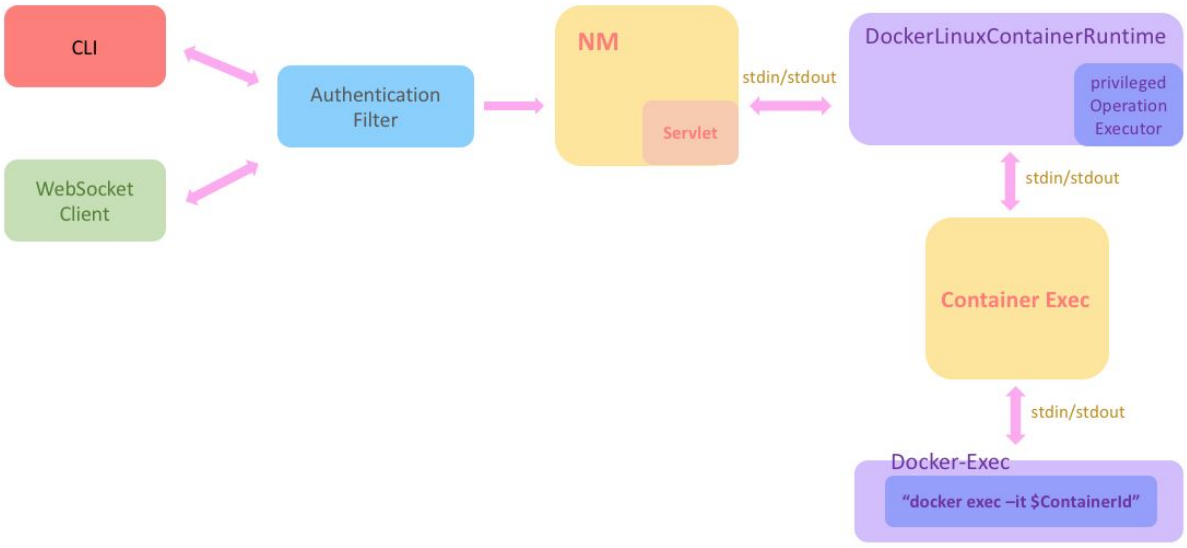

Figure 1 Architecture

Implementation Details

In this section we give the implementation details of the system design we proposed above. Each subsection corresponds to a sub-JIRA of the JIRA YARN-8523.

Add WebSocket logic to the Node Manager web server to establish servlet

The reason we want to use WebSocket servlet to serve the backend instead of establishing the connection through HTTP is that WebSocket solves a few issues with HTTP which needed for our scenario,

1. In HTTP, the request is always initiated by the client and the response is processed by the server — making HTTP a unidirectional protocol, while web socket provides the Bi-directional protocol which means either client/server can send a message to the other party.

2. Full-duplex communication — client and server can talk to each other independently at the same time
3. Single TCP connection — After upgrading the HTTP connection in the beginning, client and server communicate over that same TCP connection throughout the lifecycle of WebSocket connection

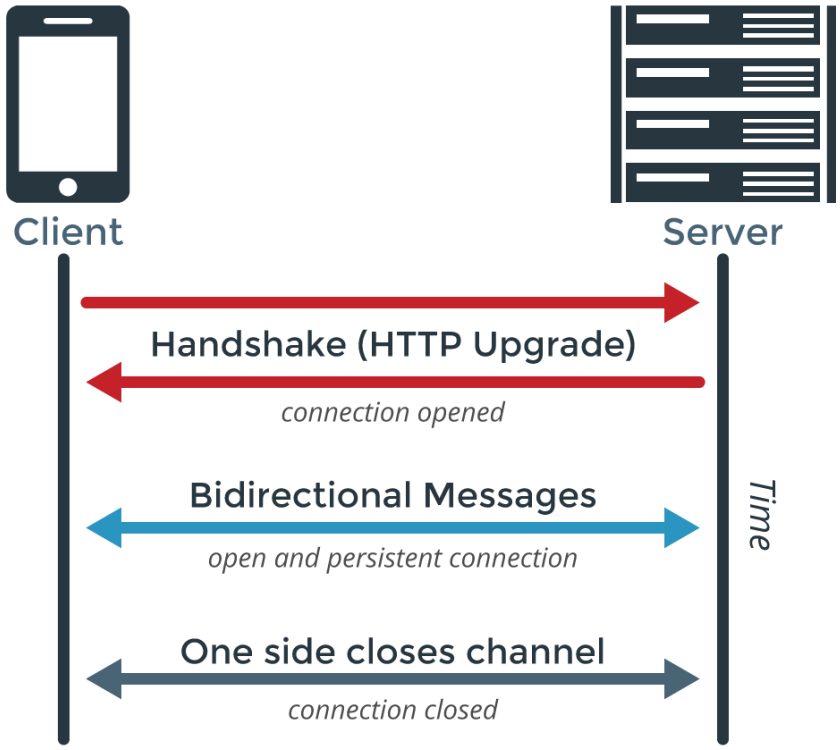

Figure 2 Websocket connection (Image from PubNub.com)

Container Executor change to create stdin/stdout pipeline

As shown in Figure 1, the pipeline is built to connect the stdin/stdout channel from WebSocket servlet through container-executor to docker executor. So when the WebSocket servlet is started, we need to invoke container-executor “dockerExec” method (which will be implemented) to create a new docker executor and use “docker exec -it $ContainerId” command which executes an interactive bash shell on the container.

Container Executor C binary change

Since Container Executor provides Container execution using the native container-executor binary, we also need to make changes to accept new “dockerExec” method to invoke the corresponding native function to execute docker exec command to the running container.

Add Command Line interface to invoke interactive docker shell

CLI will be the mandatory interface we are providing for a user to use interactive docker shell feature. We will need to create a new class “InteractiveDockerShellCLI” to read command line into the servlet and pass all the way down to docker executor.

Application Catalog and UI 2 changes for client side

We also provide front-end UI serve as an alternative input source to accept interactive user command. Since Application Catalog will become another user interface to launch yarn service applications, we will add support on WebSocket servlet side to receive shell command from Application Catalog as well. The ideal effect of backend servlet and front-end UI is user “seems to be tapping the command directly into the running container”

Authentication Filter change to force security check

Hadoop node manager REST API is authenticated using AuthenticationFilter from Hadoop-auth project. AuthenticationFilter is added to the new WebSocket URL path spec. The requested remote user is verified to match the container owner to allow WebSocket connection to be established. WebSocket servlet code enforces the username match check.

References

[1] https://hortonworks.com/blog/trying-containerized-applications-apache-hadoop-yarn-3-1/