CodingInfinity

Benchmark Service Installation Manual

Git:
`https://github.com/CodingInfinity/Benchmark-Service-Documentation`
GitHub Organisation: `https://github.com/CodingInfinity`

The Client:
Ms Vreda Pieterse
Department of Computer Science
University Of Pretoria

The Team:
Andrew Broekman *11089777*
Brenton Watt *14032644*
Fabio Loreggian *14040426*
Reinhardt Cromhout *14009936*

September 2016

Contents

1 Introduction

This is the installation manual for the Benchmark Service. It details how to install each part of the system in order to provide this service. An installed prototype of this service is hosted at the Department of Computer Science at the University of Pretoria.

2 Benchmark Management Server

The first thing is to make sure that you have Maven set up and working on the target machine.

Then the repository can be cloned using git with this command:

```
$ git clone https://github.com/CodingInfinity/Benchmark-Management-Server.
    git
```

Thereafter the desired properties can be set in the application-prod.yml file which can be found in the src/main/resources/config directory.

The server can then be run by executing the terminal command:

```
$ spring-boot:run
```

2.1 Required Dependencies

The Benchmarking system requires a PostgreSQL and Apache ActiveMQ setup. A docker-compose.yaml file, located in the root of the Management Server, can be used to provide the required infrastructure.

3 Benchmark Web Interface

On the machine which will be serving as the web host, the following commands can be used to compile the Angular2 front-end application:

```
$ git clone https://github.com/CodingInfinity/Benchmark-Web-Application.git
$ cd Benchmark-Web-Application
$ npm install
$ npm build
```

After the above commands have been issued, a *dist* folder can be located within the project root directory. This folder can be servered as the root directory with a web server such as Apache HTTPd or nginx.

4 Benchmark Instrumentation Application

This section details the installation steps required to setup the instrumentation application on the profiler node.

4.1 Required Dependencies

- Relative POSIX-compliant *NIX system
- g++ 4.2
- boost 1.53.0
- Runtime libraries for lex and yacc might be needed for the Apache Thrift compiler.

4.2 External Dependencies

- Apache Qpid Proton
- Apache Qpid CPP with AMQP version 1.0 (Requires Apache Qpid Proton)
- Apache Thrift C++ binding
- Hyperic SIGAR
- YAML CPP

For all dependencies listed below please refer to relevant documentation for building the dependency in question, such as required dependencies required by the libraries itself.

4.2.1 Apache Qpid Proton

```
$ git clone git://git.apache.org/qpid-proton.git
$ cd qpid
$ mkdir cmake-build
$ cd cmake-build
$ cmake ..
$ make all
$ make install
```

4.2.2 Apache Qpid CPP

```
$ git clone git://git.apache.org/qpid-cpp.git
$ cd qpid-cpp
$ mkdir cmake-build
$ cd cmake-build
$ cmake ..
```

```
$ make all
$ make install
```

4.2.3 Apache Thrift

```
$ git clone git://git.apache.org/thrift.git
$ cd thrift
$ mkdir cmake-build
$ cd cmake-build
$ cmake ..
$ make
$ make install
```

4.2.4 Hyperic SIGAR

```
$ git clone https://github.com/hyperic/sigar.git
$ cd siger
$ mkdir cmake-build
$ cd cmake-build
$ cmake ..
$ make
$ make install
```

4.2.5 YAML CPP

```
$ git clone https://github.com/jbeder/yaml-cpp
$ cd yaml-cpp
$ mkdir cmake-build
$ cd cmake-build
$ cmake ..
$ make
$ make install
```

To compile the Instrumentation software on the profiler node, the following commands can be issued from a terminal:

```
$ git clone https://github.com/CodingInfinity/Benchmark-Instrumentation-
    Application.git
$ cd Benchmark-Instrumentation-Application
$ mkdir cmake-build
$ cd cmake-build
```

```
$ cmake ..
$ make
$ make install
```

Upon completion, the compiled application, *Instrumentation*, can be found within the cmake-build directory.