

## HCL Detect v12.1.9 Installation Guide



# Contents

- Chapter 1. Installing HCL Detect On-Premise..... 3**
  - Prerequisites.....3
  - Dependencies..... 3
  - Installing OS-managed packages..... 5
  - Configuring MariaDB.....6
  - Configuring Redis for use by HCL Detect..... 9
  - Installing Detect Service..... 10
  - Configuring HCL Detect's Runtime Environment..... 15
  - Configuring Web Proxy..... 15
- Chapter 2. Installing HCLDetect on OpenShift..... 22**
  - Prerequisites.....22
  - Configuring user in VM.....23
  - Configuring MariaDB.....23
  - Installing Detect Services..... 24
  - Configuring Shared Drive.....26
  - Configuring Helm Charts.....27
- Chapter 3. Running HCL Detect..... 30**
- Chapter 4. Accessing the interface for the first time.....35**

# Chapter 1. Installing HCL Detect On-Premise

HCL Detect is an on-premise/Hybrid next generation enterprise marketing events detection product. Built for marketers, it enables marketing teams to plan and manage the customer life cycle events to enable engagement products deliver personalized and highly contextual campaigns quickly moving from micro segments to a segment of one.

It runs on Red Hat 8 (or CentOS 8).

In the next sections, we will go over the software dependencies that must be satisfied prior to the installation of HCL Detect.

## Prerequisites

Before you begin the installation process, ensure that the following prerequisites are met:

### Software Requirements

Software Type	Requirement
Operating System	Red Hat Enterprise Linux 8
Database	MariaDB 10.6
Distributed Event Streaming Platform	Apache Kafka 3.8.1

### Hardware Requirements

The minimum hardware requirements for HCL Detect are as follows

Hardware Type	Requirement
Architecture	x64 Architecture
Virtual Machine	Minimum 4 core with 16GB RAM
Storage	As required

Before proceeding with the installation of the required components, you have to ensure that Red Hat OpenShift is installed and properly configured. Refer to the Red Hat OpenShift documentation for [installation guidelines](#).

## Dependencies

The server where HCL Detect is installed must be configured according to a few guidelines. The software also relies on a set of external dependencies that must be installed prior to setting it up for use. In this section, we discuss both of these issues.

### Operating System Configuration

The server (or servers) where Detect runs must be properly configured as far as certain OS and shell resource limits.

While these limits are going to be checked during installation, we will provide certain helpful guidelines in this section.

First, the number of file descriptors that can be used by a process must be equal to or greater than 640,000.

The soft and hard limits are defined as part of the OS configuration (defined in `/etc/security/limits.conf`) and used/enforced by the (Bash) shell.

In the shell, the *soft* limits for each of these resources (including the maximum number of open files) can be inspected as follows:

```
ulimit -Sa
```

And the *hard* limits as follows:

```
ulimit -Ha
```

Naturally neither limit can go beyond the OS limit that applies to the server.

If the current limit is too low for the current shell, but sufficient as far as the OS is concerned, you can update your local `.bashrc` raising the limit as follows:

```
ulimit -n 640000
```

Nevertheless, in most cases, after a fresh OS install, it is necessary to update `/etc/security/limits.conf` to meet Detect's needs.

We recommend that you consult your OS documentation, but usually the following settings can be added towards the end of the file (assuming you want the limits applied to all users):

```
* soft nofile 640000
* hard nofile 640000

# End of file
```

Similarly, make sure that the soft core and hard core values are not zero. If so, add these lines at the end of file as shown below.

```
* soft core unlimited
* hard core unlimited
```



**Note:** After applying these changes, a new login will be created with the userid that will be used in Detect (no reboot is necessary).

## External Dependencies

The external software dependencies are required by Detect are specific to the operating system version and architecture where the software will run.

There are two types of external dependencies. The ones provided by the OS vendor and the ones from external vendors.

Starting with the dependencies provided by external vendors, requiring a manual installation, you will need to obtain:

- optional: IBM InfoSphere Streams, IBM SPSS, as well as Oracle WebLogic, if these are capabilities you have acquired in your particular HCL Detect installation. These are commercial products and you should consult their respective documentation to have them installed in your environment. Their integration with HCL Detect is described later in this section, where the `installer` will request additional information regarding their location in the file system.

## Installing OS-managed packages

When it comes to OS-managed dependencies, i.e., dependencies that can be installed using the operating system package management, the installer will look for and warn you about missing dependencies.

The installation package comes with a utility, `dependency_checker`, that can be used to ensure that all dependencies are in place ahead of the installation.

This utility inspects the environment for Red Hat or Ubuntu-provided software (referred to as OS-provided software in the rest of this documentation) as well as for specific Python packages required by HCL Detect, which are provided as a `virtualenv` environment, pre-configured to match the HCL Detect needs.

Operating System-software packages must be installed using the regular mechanism employed to download and install them, usually `yum` on Red Hat and `apt-get` on Ubuntu.

When using `dependency_checker` to extract the list of required dependencies, the output will be similar to (but not necessarily the same as) the following:

```
$ ./drive/bin/dependency_checker -l
List of OS package dependencies:

advance-toolchain-at8.0-runtime: 8.0 (installed)
mariadb: 5.5 (not installed)
mariadb-libs: 5.5 (installed)
mariadb-server: 5.5 (installed)

List of Python package dependencies (available in the HCL Detect virtualenv):
...
```



**Note:** The software versions mentioned are provided for reference purposes. Please be aware that the versions may vary depending on the specific release.

In this example, one external dependency (`mariadb`) is not currently installed. In this case, assuming this host is running Red Hat Linux, `yum` must be used to install `mariadb`.



**Note:**

*Installing OS packages on a server without Internet connection*

In many cases, the server (or cluster) where HCL Detect is going to be installed is not directly connected to the Internet.



In such cases, the installation of additional OS-level packages can be accomplished by having access to the Operating System installation CD/DVD or, simply, to an .iso image with the OS installation.

If your installation is Red Hat based, use one of the following alternatives:

- if a DVD is available, please follow the [DVD-based yum repository directions](#) outlined by Red Hat to create a locally available yum repository.
- if an .iso file is available, please follow these [.iso file directions](#) to create a locally available yum repository.

If your installation is Ubuntu-based, use one of the following alternatives:

- if a DVD is available, please follow the [DVD-based apt repository directions](#) outlined by Canonical to create a locally available aptitude repository.
- if an .iso file is available, mount it first and then use the mounting point in the steps above as the location when running `apt-cdrom`. To mount the .iso perform the following steps either `sudo-ed` or by logging in as `root`:

```
$ mkdir -p <mounting point location>
$ mount -o loop <file>.iso <mounting point> location
```

When installing external operating system-managed dependencies, as long as the major and minor version numbers match, the dependency is considered satisfied.

## Configuring MariaDB

HCL Detect requires a relational database to store configuration as well as runtime data used by the applications as it runs.

Currently, the only supported database is [MariaDB](#).

### Installing MariaDB

If you need to have it installed prior to installing HCL Detect, please refer to [MariaDB's online installation instructions](#) to become acquainted with both the installation and configuration process. The actual installation is done via `apt-get` or `yum`, depending on whether you are installing it on Ubuntu or Red Hat.

MariaDB is executed as a service on both Ubuntu as well as on Red Hat and its behavior is controlled by a configuration file (usually located in `/etc/my.cnf.d/server.cnf` on Red Hat or `/etc/mysql/my.cnf` on Ubuntu). In this file, specifically in the `mysqld` section of the configuration, the following entry **must be commented out or removed**:

```
bind-address = 127.0.0.1
```

This change will ensure that other hosts in the cluster can interact with the MariaDB server.

If the version for the MariaDB server is older than 10.3.1, the InnoDB engine must also be appropriately configured with the following settings in the `[mysqld]` section:

```
innodb_file_format=Barracuda
innodb_file_per_table=ON
innodb_large_prefix=ON
```

The configuration modifications only become active after a server restart, which requires restarting the specific operating system service as follows. On Red Hat:

```
$ sudo systemctl restart mariadb
```

And on Ubuntu:

```
$ sudo systemctl restart mysql
```

Finally, ensure that the MariaDB server is automatically started at boot time by appropriately configuring `init`, `systemd`, `cron` or whatever mechanism is in place for automating the startup of services.

## Securing MariaDB

Once the installation is performed, it's also recommended that MariaDB installation and configuration be [hardened](#):

```
$ sudo mysql_secure_installation
```

The simplest way to make the MariaDB server (`mysqld`) available to HCL Detect is to install it on the same host where HCL Detect is going to be placed. It is also necessary to ensure that the server is using its default port (3306):

```
$ netstat -lptn | grep 3306
tcp    0      0 0.0.0.0:3306      0.0.0.0:*        LISTEN  11430/mysqld
```

Note that placing the MariaDB server on another host as well as using a different port number can be done, but both of these options require additional configuration.

## Populating the MariaDB timezone tables

Several tables in the MariaDB system database exist to store time zone information.

The MariaDB installation procedure creates the time zone tables, but does not load them.

To do so manually, as `root` run the following command (MariaDB's `root` password will be requested):

```
$ mysql_tzinfo_to_sql /usr/share/zoneinfo | mysql -u root -p mysql -D mysql
```

There will be one or two warnings similar to:

```
Warning: Unable to load '/usr/share/zoneinfo/leap-seconds.list' as time zone. Skipping it.
```

which you can ignore.



### Note:

Loading the time zone information is not necessarily a one-time operation because the information changes occasionally.

When such changes occur, applications that use the old rules become out of date and you may find it necessary to reload the time zone tables (using the instructions above) to keep the information used by the MariaDB server current. Please refer to the MariaDB [documentation](#) for more information.

## Setting up a MariaDB user

Once MariaDB is properly installed, it must be configured with a user and password to be used by HCL Detect when establishing connections with the database server. By default, both the user and password are set to `drive`.

To create a MariaDB user called `drive` with the password `drive`, start MariaDB's interactive shell using MariaDB's `root` user:

```
$ mysql -u root -p
```

And issue the following command:

```
MariaDB [(none)]> CREATE USER '<user>'@'%' IDENTIFIED BY '<password>';
```

Replacing `<user>` and `<password>`, with `drive`, specifically:

```
MariaDB [(none)]> CREATE USER 'drive'@'%' IDENTIFIED BY 'drive';
```

Once the user is created, it must be given privileges to create the HCL Drive databases:

```
MariaDB [(none)]> GRANT ALL PRIVILEGES ON `drive_%` . * TO 'drive'@'%';
MariaDB [(none)]> FLUSH PRIVILEGES;
```

You can now close the MariaDB interactive shell by pressing CTRL-D (the `Control` and the `D` key, together) or by using the `exit` command.

At this point you should be able to login to MariaDB using the user `drive` and authenticate using the initial, default, password you just configured:

```
$ mysql -u drive -p
```

If the new user has been properly configured, you will once again be greeted by the MariaDB interactive shell:

```
$ mysql -u drive -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 39
...
MariaDB [(none)]>
```

## Using a non-default MariaDB configuration

The use of MariaDB on a different host as well as the use of a different user name and/or password requires updating the HCL Detect configuration file `HCL/etc/drive.json`.

This JSON file can be edited using a regular text editor.

To change the password used by the HCL Detect database server user, it's necessary to first encrypt the new password so it's not stored in clear text in the configuration file. HCL Detect includes a tool that can be used to perform this operation:

```
$ $HCL_HOME/bin/password_encryptor -p <new_password>
```

Once you have the newly encrypted password, modify the `drive|database|{userName,userPassword}`, and, if necessary, the `solution|database|{userName,userPassword}` entries in the configuration file with the new password as well as a new user name.



As mentioned, some HCL Detect installations have optional features (i.e., a solution) that also make use of a relational database.

Note that the configuration file is nested, so the notation used in the prior sentence means, e.g., the entry `userName` as well as the entry `userPassword` are under the entry `database`, which is under the `solution` entry.

Note that if you are changing one or both of these settings, MariaDB itself must be updated with this new user/password information.

To change the host and port the MariaDB server uses, locate and update one or more of the following entries: `drive|database|address` and, if necessary, the `solution|database|{userName,userPassword}`. The address is a tuple with the following format: `hostname:port`, e.g., `foo.HCL.com:3306`.

## Configuring Redis for use by HCL Detect

HCL Detect uses [Redis](#), an in-memory data structure store.

You might be required to address certain OS-level requirements to ensure that Redis runs efficiently. As the [Installing Detect Service on page 10](#) is carried out, you might face warnings similar to the following:



### Warning:

The operating system in this host is not optimally configured to run Redis, details:

- `overcommit_memory` is set to 0. Redis background save process may fail under low memory conditions. To fix this issue add `'vm.overcommit_memory = 1'` to `/etc/sysctl.conf` and then reboot or run the command `'sysctl vm.overcommit_memory=1'` for this to take effect. For more details, please refer to <http://redis.io/topics/admin>
- when Transparent Huge Pages (THP) support is enabled in the Linux kernel (as is the case now), it can lead to high latency when Redis forks to persist data to disk. THP support can be disabled by executing the command `'echo never > /sys/kernel/mm/transparent_hugepage/enabled'` as root, and adding it to your `/etc/rc.local` in order to retain the setting after a reboot. When making it permanent, make sure that the `rc.local` file has the execution permission for the owner (if this file is a symlink, ensure that the permission is set for actual file being linked). For more details, please refer to <http://redis.io/topics/admin>
- unable to enforce the TCP backlog settings of 511 yet-to-be-listened TCP connections in Redis. To fix this issue add `'net.core.somaxconn=511'` to `/etc/sysctl.conf` and then reboot or run the command `'sysctl -w net.core.somaxconn=511'` for this setting to take effect immediately, but only until the next reboot. For more details, please refer to <http://redis.io/topics/admin>

Ensure that, if/when these errors are displayed, the changes suggested by the error messages are put in place.

# Installing Detect Service

## Installation Package

The HCL Detect installation tarball includes:

- The HCL Detect software platform itself, comprising all of the necessary components to run HCL Detect-supported applications.
- The pre-configured Python `virtualenv` environment, comprising all Python dependencies required by HCL Detect applications to run.
- The external open source software required by HCL Detect, e.g., Apache Tomcat, Apache Kafka, among others.

## Installing Detect

The installation process must be carried out using the userid that will manage the HCL Detect software.

It is recommended that this user be named `drive` or, if using a hybrid HCL Detect/InfoSphere Streams environment, `streams`.

Prior to starting the actual installation, if HTTPS-secured web access to HCL Detect will be made available, a (optional) DNS entry must be configured to provide a user-friendly URL to end users as well as a self-signed or commercial SSL certificate must be on-hand as it will be required to complete the product installation.

It might be helpful to become familiar with the infrastructure used to provide HTTPS access to Detect by reading the steps outlined in the section on web proxy configuration before attempting the installation steps.

The installation process begins by un-tarring the software tarball:

```
$ tar xvfz drive-x.x.x-rhel07.tar.gz
```

The suffixes `x.x.x` denote the version you are installing and `rhel07` the specific operating system (`rhel07` for Red Hat 7). Please update the version number accordingly.

Extract the software:

```
$ mkdir -p /opt/HCL/drive/rhel07
$ tar xzvf drive-x.x.x-rhel07.tar.gz -C /opt/HCL/drive/rhel07
```

Again, other locations are acceptable, but `/opt/HCL/drive/rhel07` is the recommended path. Now, you are ready to perform the configuration steps:

```
$ cd /opt/HCL/drive/rhel07/HCL/bin
$ ./installer
```

`installer` is an interactive program and will guide you through specific installation and configuration choices:

```
HCL Detect is a commercial product, subjected to End-User License Agreement terms. A paper-based or digital
copy of these terms must have been signed and agreed by someone authorized to do so in your organization, prior
to
carrying out this configuration. A non-customer specific copy of these terms is included for your reference in
this
installation package (HCL/license/eula.pdf). Do you confirm that you are authorized to proceed with the
configuration based on the terms specified in your organization's own license agreement with HCL Inc. (y/n)?
```

As a first configuration step, you will be asked about the directory that will be used to host the HCL Detect instance, i.e., the location in the file system that HCL Detect will use to host its services their logs as well as the data that will be kept by the HCL Detect data management services.

Next, you will be asked which network interface to use for external TCP/IP traffic. You should choose the interface that provides connectivity to other hosts in the cluster (if any) and/or external services the HCL Detect application will interface with:

```
Please select the network interface to use for external TCP/IP traffic
(default: 'eth0'):

[0] lo: 127.0.0.1
[1] eth0 10.0.0.123
[2] tun0: 10.8.0.45

Selecting the number corresponding to the interface you want (default: '1' for interface 'eth0'): 1

The 'eth0' network interface will be used for all external TCP/IP traffic.
```

Once a network interface is selected, the installer will proceed to ask which transport protocol should be used between HCL Detect's web-based frontend and its backend services:

```
Do you want to use HTTPS-based interactions between the HCL Detect browser-based interface and its backend
([n]o: HTTP will be used (in production, a proxy such as nginx must be used to secure the client/server
communication via HTTPS); / [y]es: HTTPS will be used with a self-signed SSL certificate (recommended only for
short-term testing) (default: HTTP) (y/n)?
```



**Warning:** HCL Detect has access to and handles potentially sensitive information:

- **Authentication information:** the use of HCL Detect requires a user account. HCL Detect has its own directory of users or can, alternatively, defer to an enterprise-wide LDAP server. In either case, user passwords are employed to ensure that a user is both authenticated and authorized to use the system. While HCL Detect never stores user passwords in the clear, certain interactions require the transferring of passwords between the front end, web-based interface to the backend. Hence, encryption (through the use of HTTPS) is STRONGLY recommended.
- **Metadata and structural information about subscriber and corporate data feeds:** HCL Detect carries out analytics employing data that is often private and sensitive. Once, again, interactions between its web-based interface and its backend require manipulating such data and encryption, in the form of HTTPS interactions, is STRONGLY recommended to preserve end-to-end confidentiality.

While HCL Detect is generally hosted in an internal network, never facing non-corporate users, it does integrate with other segments of the enterprise computing environment. HCL recommends that administrators take every possible precaution to protect the integrity and confidentiality of the data consumed and produced by this platform.

There are multiple possible configurations to choose from and each one has certain advantages and risks:

- HTTP (without a proxy such as nginx securing the client/server interactions), available network-wide (STRONGLY DISCOURAGED): this is the simplest form of installing HCL Detect. Nevertheless, it is **insecure** as potentially sensitive information is transmitted in the clear over the network, flowing from the user's browser to the server without any encryption, including passwords (potentially, even the ones used for authenticating via enterprise-wide repositories such as a corporate LDAP server, if such an integration is eventually enabled).

**Note:***SSL certificates*

An SSL certificates is a data file that digitally binds a cryptographic key to an organization's identity. For instance, when installed on a web server, it activates the padlock used by the browser to indicate a `_secure_` connection and, hence, the HTTPS protocol in interactions between the browser and a server.

In general, an SSL certificate is obtained from a Certificate Authority (CA) and it attests the ownership of a public key by the named subject of the certificate, providing an assurance that an interaction is occurring between a client and a properly identified server. A CA acts as a third party, trusted both by the subject (owner) of the certificate and by the party relying upon the certificate.

The HCL Detect web service can make use of an SSL certificate to ensure that interactions between the web-based client and server are properly authenticated (i.e., to provide an assurance to the browser-based client that it is indeed speaking to an actual server) as well as encrypted, such that no sensitive information flows over between a client and server in clear text form.

SSL certificates can be purchased from several vendors or obtained for free from organizations such as [Let's Encrypt](#). Commercial SSL certificates are typically verified and accepted by mainstream web browsers such as Google Chrome and Mozilla Firefox.

SSL certificates can also be provided by any entity hosting a Public Key Infrastructure (PKI). For instance, an organization's IT department might host its own internal PKI and issue self-signed certificates. These certificates work just like commercial certificates, but they will typically produce a warning or an outright rejection from web browsers, which will not recognized the PKI's CA. In such cases, upon being directed by the organization's IT department, the certificate may be added to the browser and accepted as legitimate, thus quieting down the warnings that would, otherwise, be raised every time a web server presenting it is accessed.

HCL Detect can be configured with either type of certificate, but HCL strongly recommends that a certificate be obtained from an officially recognized commercial or non-profit CA.

- HTTPS, available network-wide (RECOMMENDED for testing only): this configuration is deemed safe, as it minimizes the chances of a sensitive data breach. In such a configuration, the interactions between the browser-based user interface and HCL Detect's backend is carried out via HTTPS interactions (encrypted). In this case, the HCL Detect backend service (hosted by Apache Tomcat) is the direct destination of calls performed by a user's browser to the server servicing the REST APIs.

The disadvantages of this approach are two-fold. First, it employs a self-signed certificate generated by HCL and not a commercial certificate issued by a proper Certification Authority. Second, the URL to access HCL Detect will be in the form `https://drive.company.com:<port>/drive`. In other words, the `port` where the service runs will be part of the URL. Typically, HTTPS servers bind and run on the privileged port 443 and such a port number can be omitted from the URL. While the Apache Tomcat-based Detect backend can be configured to use port 443, this configuration is not recommended since it requires running the web server as `root`, which opens up the possibility of a complete takeover of the host where the web server runs should an unknown (but possible) security vulnerability be exploited.

Alternatively, it is also possible to fiddle with the Operating System settings to allow non-`root`-owned applications to use a privileged port. Nevertheless, such a configuration is both non-standard and complex from a system administration standpoint.

If this configuration is chosen, the `installer` will ask for a non-privileged port to be used by the web server and will automatically create a certificate authority, will issue a CA certificate, and install a self-signed SSL certificate to be used by the web server as part of the installation process.

- HTTP, available only in the `localhost` interface, proxied by an HTTPS web proxy (STRONGLY RECOMMENDED): this configuration is deemed the safest, as it minimizes the chances of a sensitive data breach. As with the prior configuration, the interactions between the browser-based user interface and the web proxy in front of HCL Detect's backend are HTTPS-encrypted.

The web proxy (both Apache `httpd` and `nginx` are supported) employs a regular local HTTP connection to the HCL Detect backend and a client will interact with the proxy via HTTPS, which will in turn relay the requests to the web server.

While interception of end-user communication with the web server is possible, this can only occur when the host where the web server is installed is compromised and `root` access is available to the malicious party.

The benefit of this approach is that the web proxy (which is specifically hardened for this task), not HCL Detect's backend, runs as `root`. Furthermore, the web proxy's internal design is optimized for these types of interactions, the use of commercially-issued or locally-issued SSL certificates, specific SSL ciphers, as well as many other SSL-related configurations much easier to put in place.

In this configuration, the installation of an SSL certificate as well as the configuration of the HTTPS endpoint is done by installing and configuring the web proxy, a step described in the web proxy configuration section.

Finally, an HCL Detect installation can be optionally configured with additional integration points to external software packages including IBM SPSS Modeler Solution Publisher, IBM InfoSphere Streams, as well as Oracle WebLogic Server. Each of these integration points requires a prior licensed installation for each of these software packages. When installing an HCL Detect version enabled with one or more of these integration points, additional installation steps will take place. These steps, applicable only to the specific integration points enabled in the installation, will be carried out by the `installer` program to ensure that it can find the proper version for each of the external packages needed by them. For instance:

```
Please enter the path to an existing IBM SPSS Modeler Solution Publisher
installation to use: /opt/x86_64/ibm/spss/ModelerSolutionPublisher/17.1
```

```
Please enter the path to an existing IBM InfoSphere Streams
installation to use: /opt/rhel07/ibm/streams/4.0.1.0

ERROR: failed to validate IBM InfoSphere Streams
installation at '/opt/rhel07/ibm/streams/4.0.1.0'
Product information file '/opt/rhel07/ibm/streams/4.0.1.0/.product' does not exist
Please provide a valid path to an existing IBM InfoSphere Streams installation

Please enter the path to an existing IBM InfoSphere Streams
installation to use: /opt/rhel07/ibm/streams/4.0.1.0

Please enter the path to an existing Oracle WebLogic Server
installation to use: /opt/HCL/drive/noarch/weblogic-10.3.6.0
```

If everything is correctly configured, a success status message will be printed out:

```
HCL Detect environment was successfully installed...
```

Now that the configuration is complete, a test can be executed. To run any HCL Detect application, the shell where the application is going to run must be configured by invoking the `environment_setter` script (<drive-install-

```
location>/HCL/bin/environment_setter):
```

```
$ ./environment_setter
```

Once the script is source'd, several environment variables are configured and the Python `virtualenv` environment is activated, indicating that we are ready to start an HCL Detect application:

```
[drive]]$ env | sort | grep HCL
HCL_ARCH=x86_64
HCL_HOME=/opt/HCL/drive/rhel07/HCL
HCL_JAVA_PATH=/usr/lib/jvm/java
HCL_OS=rhel07
HCL_PACKAGE_VERSION=2.2.0
HCL_PYTHON_PATH=/opt/HCL/drive/307/pyenv
HCL_READLINK=readlink
HCL_TOMCAT_PATH=/opt/HCL/drive/noarch/apache-tomcat-9.0.20
```

The preparation of the shell must be made for every shell and session where an HCL Detect application will run. Now, we can run a test application (<drive-install-location>/HCL/bin):

```
[drive]]$ cd HCL/bin
[drive]]$ ./installation_tester -c -sns
INFO: this application is using '/tmp/<user>/HCL' to output and store its
code-generated assets...
.
.
```



**Note:** We are invoking the test application `installation_tester` using the `-c` flag to indicate that the application should be re-built (just in case, it has been executed before), thus ensuring that a fresh version, based on the



current installation, is executed. Normally, HCL Detect applications are only rebuilt when needed and taking such a precaution is normally not necessary.

## Configuring HCL Detect's Runtime Environment

The HCL Detect runtime environment can be configured based on the specific performance and reliability requirements of a customer deployment. These runtime environment settings are specified in the following file: `$HCL_HOME/etc/runtime_environment.json`.

An important runtime configuration is the persistence behavior of PinPoint, which is the Redis-based distributed in-memory store used by Detect to maintain subscriber profiles. As a primary mechanism, PinPoint relies on append-only logging to persist its data. As an additional mechanism, it also periodically checkpoints its data to disk.

The checkpoint can be configured using the `pinPoint|server|checkpointConfiguration|schedules` setting inside the `runtime_environment.json` file. The default checkpoint configuration is given as follows:

```
"schedules":
[
  {
    "operationCount": 1,
    "timeInSeconds": 86400
  },
  {
    "operationCount": 1000000,
    "timeInSeconds": 3600
  }
]
```

This configuration indicates that the state is checkpointed every hour if at least 1 million profiles are updated since the last checkpoint, and every day if at least one profile is updated since the last checkpoint. More schedules can be added or the existing schedules can be updated, as needed. Since the append-only logging already provides persistence, specifying frequent checkpointing is unnecessary and can adversely affect the CPU usage of the PinPoint servers.

## Configuring Web Proxy

HCL Detect consists of a set of backend services, accessible via a web-based user interface.

Once installed, an instance of HCL Detect can be started. The communication between the web-based user interface and the backend employs REST APIs over HTTP or HTTPS.

HCL strongly recommends that HTTPS be used to ensure that the sensitive information that transits over the network is encrypted.

As seen in the [Installing Detect Service on page 10](#) section, HCL Detect offers the choice of web transport protocols during its installation process and recommends HTTPS with the use of a web proxy as the means to access its private backend services.

This option requires installing HCL Detect with HTTP support, where the backend is bound only to the local `lo` network interface, allowing HTTP access only within the host where the HCL Detect web server runs.

Either of two commonly used web proxies, packaged and available in the Linux distributions supported by HCL Detect, can be configured to provide the HTTPS-based access to the users interacting with HCL Detect via their web browsers: [Apache httpd](#) and [nginx](#).

In either case, two steps must be carried out prior to the web proxy installation and configuration. These steps and the specific configuration of a web proxy are outlined next.

## Configuring a DNS CNAME entry to point to HCL Detect (optional)

Ideally, the URL used to access HCL Detect will be in the form `https://HCL.acme.com/drive`, where `HCL` designates this locator as an HCL product installation at `acme.com` (the customer's Internet domain) and the context path, `drive`, indicates the name of the product.

Usually, the friendlier `HCL` name will map to an internal server hosting the HCL Detect installation, whose name will follow an (often less friendly) internal IT convention (e.g., `drive-cluster003-node001.acme.com`). Thus, we recommend that a DNS alias (specifically a `CNAME` entry) be obtained and registered prior to the HCL Detect installation.

Procedures to update the corporate DNS servers vary. We recommend that the system administrator installing Detect contacts a local IT specialist to understand the process and to carry out the actual DNS registration.

As an example, an organization using TinyDNS will need to add the following entry to its configuration:

```
CHCL.acme.com:drive-cluster003-node001.acme.com:120
```

And invoke the utility (`tinydns-data`) to activate this entry.

## Obtaining an SSL certificate

SSL certificates come in different forms. A certificate can be obtained for a single host name, for multiple host names, or as a wild card, accepting any name under a particular domain. While all of them ought to work with HCL Detect, a single-host certificate is sufficient and this option, if procuring a commercial certificate, is often the most economical.

Procedures to obtain a certificate vary both for commercial as well as for self-signed internal certificates.

Please consult a local IT specialist to understand which alternative is the most adequate in your environment.

In the rest of this document, it is assumed that a commercial certificate is on hand and available to be used during the configuration of the web proxy.

Regardless of how a certificate is obtained, in the end, a set of files related to the certificate must be available. In the example below, we are assuming that the certificates are being kept in a directory called `HCL.acme.com`:



- `HCL.acme.com/fullchain.pem`: this is the actual certificate plus the intermediate certificates (if any). See the documentation [the Apache httpd SSLCertificateFile configuration key](#) and the [the nginx ssl\\_certificate configuration key](#) for more information.
- `HCL.acme.com/privkey.pem`: this is the certificate private key for the server. See the documentation [the Apache httpd SSLCertificateKeyFile configuration key](#) and the [the nginx ssl\\_certificate\\_key configuration key](#) for more information.

Once the certificate has been obtained and these files are available either the Apache `httpd` or `nginx` must be installed and configured.

In the following sections, we will describe these steps assuming that the installation is being made on a RedHat server, where the HCL Detect web backend will eventually run.

Since the installation and configuration require updating system-owned resources, either `sudo` or `root` access is required.

## Configuring Apache `httpd`

The installation requires downloading and installing the necessary OS packages:

```
$ yum install -y httpd mod_ssl
```

To ensure that the Apache `httpd` server is started on boot, the service must be enabled:

```
$ systemctl enable httpd
```

Once installed, the configuration enabling the proxying must be added to `httpd`'s main configuration file `/etc/httpd/conf/httpd.conf`. The following excerpt demonstrates how this is done, by creating a new virtual host entry:

```
...
<VirtualHost *:443>
    ServerName HCL.acme.com
    ## HCL Detect will accessible at https://HCL.acme.com/drive
    ProxyPass /drive http://[::1]:8081/drive
    ## HCL Detect must be configured with HTTP access restricted to localhost binding to port 8081
    ## Note that the IPv6 address being used, [::1], is the IP address corresponding to the loopback interface.
    ## HCL Detect's Tomcat instance by ## default will use the IPv6 protocol stack
    ProxyPassReverse /drive http://[::1]:8081/drive
    ProxyPreserveHost on
    ProxyRequests off
    Header always set Strict-Transport-Security "max-age=31536000; includeSubdomains; preload"
    SSLCertificateFile /etc/httpd/stash/HCL.acme.com/fullchain.pem
    SSLCertificateKeyFile /etc/httpd/stash/HCL.acme.com/privkey.pem
    SSLEngine on
    SSLProtocol TLSv1.2
    SSLProxyEngine on
</VirtualHost>
...
```



### Note:

In our configuration, we chose to install the certificate-related files under `/etc/httpd/stash`. This is not necessary, but is convenient as far as applying the correct SE Linux context to these files, which is carried out as follows:

```
$ restorecon -vr /etc/httpd/stash
```



Once this is done, `httpd` can be started (or restarted):

```
$ systemctl start httpd
```

If all is well, you should be able to start HCL Detect, open a browser and point it to `https://HCL.acme.com/drive`.

To start HCL Detect, see the [Starting and stopping HCL Detect on page 30](#) section. If a failure has happened when starting `httpd`, see the [Testing and troubleshooting the web proxy installation on page 19](#) section for additional help with troubleshooting.

## Configuring `nginx`

The installation requires downloading and installing the necessary OS packages:

```
$ yum install -y nginx
```

To ensure that the `nginx` server is started on boot, the service must be enabled:

```
$ systemctl enable nginx
```

Once installed, the configuration enabling the proxying must be added to `nginx`'s main configuration file `/etc/nginx/nginx.conf`. The following excerpt demonstrates how this is done, by creating a new server entry:

```
http {
...
    server {

        listen 443;
        server_name HCL.acme.com;

        add_header Strict-Transport-Security "max-age=31536000; includeSubDomains; preload";

        ssl on;
        ssl_certificate /etc/nginx/stash/HCL.acme.com/fullchain.pem;
        ssl_certificate_key /etc/nginx/stash/HCL.acme.com/privkey.pem;
        ssl_session_timeout 5m;
        ssl_protocols TLSv1.2;
        ssl_ciphers HIGH:!aNULL:!MD5;
        ssl_prefer_server_ciphers on;

        ssl_session_cache shared:SSL:10m;
        ## HCL Detect will accessible at https://HCL.acme.com/drive
        location /drive {

            ## HCL Detect must be configured with HTTP access restricted to localhost binding to port 8081
            proxy_pass http://[::1]:8081/drive;
            ## Note that the IPv6 address being used, [::1], is the IP address corresponding to the loopback
            ## interface. HCL Detect's Tomcat instance by default will use the IPv6 protocol stack
            proxy_set_header X-Forwarded-Host $host;
            proxy_set_header X-Forwarded-Server $host;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_redirect off;

        }
    }
}
```

```
}
}
```



**Note:** In our configuration, we chose to install the certificate-related files under `/etc/httpd/stash`. This is not necessary, but is convenient as far as applying the correct SE Linux context to these files, which is carried out as follows:

```
$ restorecon -vr /etc/nginx/stash
```

Once this is done, `nginx` can be started (or restarted):

```
$ systemctl start nginx
```

If all is well, you should be able to start HCL Detect, open a browser, and point it to `https://HCL.acme.com/drive`. troubleshooting.

To start HCL Detect, see the [Starting and stopping HCL Detect on page 30](#) section. If a failure has happened when starting `httpd`, see the [Testing and troubleshooting the web proxy installation on page 19](#) section for additional help with troubleshooting.

## Testing and troubleshooting the web proxy installation

Once the web proxy is configured and HCL Detect has been started, using the web interface is as simple as entering the access URL in the browser window.

There are typically two problems that might occur in a new installation, where either RedHat Linux or CentOS is being used:

- the firewall configuration: if the server running the proxy has `firewalld` installed, HTTPS access is typically blocked by default. While we provide some helpful directions below, we strongly recommend reading the `firewalld` documentation so the impact of these commands is fully understood.

To check on current status of `firewalld`, the following command can be executed:

```
$ firewall-cmd --list-all
public (active)
target: default
icmp-block-inversion: no
interfaces: enp0s3
sources:
services: dhcpv6-client https ssh
protocols:
masquerade: no
forward-ports:
sourceports:
icmp-blocks:
rich rules:
```

If `https` does not appear in the list of `services` (as it does above), it must be added, either temporarily:

```
$ firewall-cmd --zone=public --add-service=https
$ firewall-cmd --reload
```

Or permanently:

```
$ firewall-cmd --zone=public --permanent --add-service=https
```

At this point, re-running the status command will reveal whether HTTPS access to the server is now enabled.

- the SELinux configuration SELinux is a set of kernel modifications and tools that have been added to RedHat and CentOS, providing support for access control security policies. It may affect the web proxy in the sense.

Often, if the SELinux access control is not properly configured for the web proxy, there will be two symptoms. First, an access from the browser will be rejected, often, with a "bad gateway" error message. Second, the SELinux log file (`/var/log/audit/audit.log`, `root` access is required both for search the logs as well as for reconfiguring SELinux policies) will include a rejection of an operation attempted by the web proxy (in the example below, `nginx`, but very similar for `httpd` as well), possibly, similar to the following:

```
$ grep nginx /var/log/audit/audit.log
type=AVC msg=audit(1490570804.119:555): avc: denied { name_connect } for pid=5725 comm="nginx" dest=8080
scontext=system_u:system_r:httpd_t:s0 tcontext=system_u:object_r:http_cache_port_t:s0 tclass=tcp_socket
```

Fiddling with SELinux will have a direct impact on the security of a host as well as on the overall network where that host is located. HCL strongly recommends that changes being done on SELinux policies are undertaken by someone familiar with its configuration. The procedures we indicate below are not to be taken without understanding their potential repercussions.

This audit entry can be better understood by running the following command, which also prescribes the possible fixes:

```
$ ausearch -c nginx | audit2allow -m nginx

module nginx 1.0;

require {
    type httpd_t;
    type http_cache_port_t;
    class tcp_socket name_connect;
}

#===== httpd_t =====

#!!!! This avc can be allowed using one of the these booleans:
# httpd_can_network_connect, httpd_can_network_relay
allow httpd_t http_cache_port_t:tcp_socket name_connect;
```

You can check the current state of these Boolean's settings as follows, which at this point, is most likely `off`:

```
$ getsebool -a | grep httpd
```

Finally, the problem can be resolved by allowing network connections from the web proxy to the actual server (i.e., so it can act as a relay):

```
$ setsebool -P httpd_can_network_relay on
```

Alternatively, a new SELinux non-base policy can be put in place:

```
$ cd /tmp
$ ausearch -c nginx | audit2allow -M nginx
```

```
$ semodule -i nginx.pp  
$ rm nginx.pp
```

## Chapter 2. Installing HCLDetect on OpenShift

HCL Detect is an Hybrid next generation enterprise marketing events detection product. Built for marketers, it enables marketing teams to plan and manage the customer life cycle events to enable engagement products deliver personalized and highly contextual campaigns quickly moving from micro segments to a segment of one.

The installation process involves configuring both virtual machines (VMs) and the OpenShift environment to deploy and manage the application's services and components effectively.

In the next sections, we will go over the software dependencies that must be satisfied prior to the installation of HCL Detect.

### Prerequisites

Before you begin the installation process, ensure that the following prerequisites are met:

#### Software Requirements

Software Type	Requirement
Operating System	Red Hat Enterprise Linux 8
Database	MariaDB 10.6
Kubernetes	Kubernetes Version 1.21 and above
Container	Docker Version 20.10.7 and above
Package Manager	Helm Version 3
Container Platform	OpenShift Container Platform (OCP) 4.8
Distributed Event Streaming Platform	Apache Kafka 3.8.1

#### Hardware Requirements

The minimum hardware requirements for HCL Detect are as follows

Hardware Type	Requirement
OpenShift Kubernetes Cluster	10 CPUs with 16GiB of memory
Virtual Machine	Minimum 8 CPUs with 16GB RAM and 200GB disk
Ports required to access between Cluster and VM	80, 443, 2181, 2182, 2183, 3306, 8010, 8081, 9092, 9093, 9094, and range 32768-42769
OpenShift Storage for PVC	As required

There are two parts of the deployment:

- Micro services based components, to be deployed on containerization platform. Here, Red Hat OpenShift is the choice of the Container orchestration platform.
- VM based deployment.

Before proceeding with the installation of the required components, you have to ensure that Red Hat OpenShift is installed and properly configured. Refer to the Red Hat OpenShift documentation for [installation guidelines](#).

Also, below tools are expected to present in the deployment to facilitate the installation:

1. OpenShift CLI ("oc" command)
2. Kubernetes command line tool ("kubectl" command)
3. Helm CLI tool ("helm" command)

## Configuring user in VM

HCL Detect comes with MariaDB, and NameServer, Kafka, PinPoint and FastPast service as packages. You can install and configure these packages on VM.

Before configuring the VM, create user as driveadmin on the VM using the following command:

```
groupadd --gid 5020 driveadmin
useradd --home-dir /home/driveadmin --create-home --uid 5020 --gid 5020
```



**Note:** Make sure that the uid and gid created above should match the openshift uid and gid. Also for VM configuration, refer to [Dependencies on page 3](#).

## Configuring MariaDB

HCL Detect requires a relational database to store configuration as well as run-time data used by the applications as it runs.

### Installing MariaDB

MariaDB is executed as a service on OpenShift and its behavior is controlled by a configuration file (usually located in `/etc/my.cnf.d/server.cnf` on OpenShift. In this file, specifically in the `mysqld` section of the configuration, the following entry **must be updated as shown**:

```
bind-address = 0.0.0.0
```

This change will ensure that other hosts in the cluster can interact with the MariaDB server.

The configuration modifications only become active after a server restart, which requires restarting the specific operating system service as follows. On OpenShift:

```
- sudo systemctl restart mariadb
```

For more information on installing MariaDB, refer to [Installing MariaDB on page 6](#)

## Setting up a MariaDB user

Once MariaDB is properly installed, it must be configured with a user and password to be used by HCL Detect when establishing connections with the database server. By default, both the user and password are set to `drive`.

To create a MariaDB user called `drive` with the password `drive`, start MariaDB's interactive shell using MariaDB's `root` user:

```
$ mysql -u root -p
```

And issue the following command:

```
MariaDB [(none)]> CREATE USER 'drive'@'%' IDENTIFIED BY 'drive';
```

Once the user is created, it must be given privileges to create the HCL Drive databases:

```
MariaDB [(none)]> GRANT ALL PRIVILEGES ON `drive_%` . * TO 'drive'@'%';
MariaDB [(none)]> FLUSH PRIVILEGES;
```

You can now close the MariaDB interactive shell by pressing CTRL-D (the `Control` and the `D` key, together) or by using the `exit` command.

At this point you should be able to login to MariaDB using the user `drive` and authenticate using the initial, default, password you just configured:

```
$ mysql -u drive -h <VM IP address> -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 39
...
MariaDB [(none)]>
```



**Note:** For every new build, make sure to drop the existing database "drive\_acme\_core".

## Installing Detect Services

The HCL installation package comes as tarball which includes NameServer, Kafka, PinPoint and FastPast services.

### Installing services

After configuring the MariaDB, reach out to the HCL Support team to download the HCL Detect package containing *HCL\_Detect\_x.x.x\_Docker\_Services\_Installer.zip*, *HCL\_Detect\_x.x.x\_Docker\_Feed\_Image.zip*, *HCL\_Detect\_x.x.x\_Docker\_Tomcat\_Image.zip* and *Helm charts*.

The installation process begins by unzipping the "HCL\_Detect\_12.1.9\_Docker\_Services\_Installer.zip" file using the following command.

```
unzip HCL_Detect_12.1.9_Docker_Services_Installer.zip
```

A sample screenshot of the unzipped files is shown below:



```
$ unzip HCL_Detect_12.1.8_Docker_Services_Installer.zip
Archive:  HCL_Detect_12.1.8_Docker_Services_Installer.zip
  inflating: HCL_Detect_acme-12.1.8-rhel08.tar.gz
```

The extracted folder contains "HCL\_Detect\_acme-12.1.9-rhel08.tar.gz". Now, un-tar the software tarball using the following command.



**Note:** The rhe version referred above is for reference, and it might change accordingly to the specific releases of HCL Detect.

```
tar xvzf HCL_Detect_acme-12.1.9-rhel08.tar.gz
```

After successful extraction, the tar package contains the following folders and files as shown in image below.

```
$ ls
acme  drive  external  HCL_Detect_12.1.8_Docker_Services_Installer.zip  HCL_Detect_acme-12.1.8-rhel08.tar.gz  HCLDetectv12.1.8-LICENSE.txt  HCLDetectv12.1.8-NOTICE.txt
2024-07-13 03:46:49 TZ=0400  rhel08  root us-cxdev-lnx01 /rel_12.1.8/a5c
```

After untaring the installation file, in the terminal navigate to *drive/bin* folder. Now, you are ready to perform the configuration steps:

```
$ cd drive/bin
$ ./installer
```

For more information on interactive inputs in the terminal, refer to [Installing Detect on page 10](#).

Now, the configuration is completed, a test can be executed. To run any HCL Detect application, the shell where the application is going to run must be configured by invoking the `environment_setter` script located in (<extracted acme tar location>/drive/bin/) folder:

```
$ ./environment_setter
```

Once the script is source'd, several environment variables are configured and activated, indicating that we are ready to start an HCL Detect application.

Before starting the detect application, update the `nameServicePort` number in the *drive.json* file (suggested before pinpoint) available in the *drive/etc* folder.

```
"nameServicePort": 41232
```

Similarly, in the *runtime\_environment.json* file, update the network section as shown below:

- **"ipV6Available"**: false,
- **"usingIpV4"**: true

To start the Detect services, navigate to the directory *drive/bin*, and run the below command.

```
./services_manager start -o -p -s -t
```

In the `producer_config.properties` file, available in the `acme/etc` folder, edit the `bootstrap.servers` key with the VM IP address as shown below.

```
bootstrap.servers=<VM IP address>:9092
```



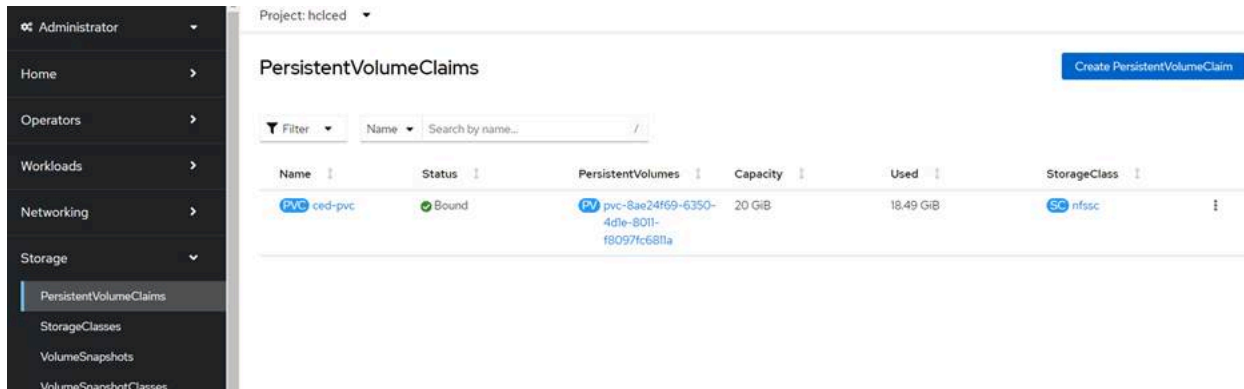
**Note:** Copy the `Acme` and `instance_home` folders to PVC.

## Configuring Shared Drive

### Creating Persistent Volume Claim for shared drive

After installing Detect services, create and configure a shared drive in kubernetes. There are two ways to create Persistent Volume Claim (PVC) in the kubernetes either using YAML file or through UI console.

In the Kubernetes UI console, click **Create PersistentVolumeClaim**, enter required parameters to create a PVC and create the PVC as shown in the below screenshot.



Alternatively, you can create a yaml file, refer [here](#), and update the required parameters like name, namespace, storageClassName and storage size. After updating the corresponding values in the yaml, run the following command to create PVC.

```
kubectl apply -f sample_pvc.yaml
```

### Configuring Mount Points

To configure PVC for mounting data to the pod, create and configure the `sample-pod-to-copy-appsrc-n-instancehome.yaml` file.

Now, copy the `acme` and `instance_home` zip files to this the `/data` folder using the `kubectl` command.

```
Kubectl cp <acme zip file> <pod-name>:/data/
```



**Note:** Update the image name of your choice.

After the files are copied, rename the `acme` folder to the `appsrc` folder, and delete all the folders in the `appsrc/lib` path.

```
unzip acme.zip
mv acme appsrc
cd appsrc/lib
rm -rf *
```

Finally, change the folder permissions of appsrc and instance\_home.

```
chmod -R 764 appsrc instance_home
```

## Configuring Helm Charts

### Configuring Services

Copy the drive.json and runtime\_environment.json files, which are extracted, available inside the drive/etc folder to the helm chart folder, and update the config parameters before deploying the helm charts

In the drive.json file, update the ip address value with 127.0.0.1 (replace <VM IP address>) as shown below.

```
"kafkaServer": {
    "ip": "<VM IP address>",
    "port": 9092
  },
  "zooKeeperServer": {
    "ip": "<VM IP address>",
    "name": "bootstrapServers",
    "stringListValue": {
      "values": [
        "<VM IP address>:9092"
      ]
    }
  },
  {
    "name": "databaseServerIp",
    "stringValue": "<VM IP address>"
  },
```

In the tomcat configuration section, update the host name as shown below:

```
"tomcatConfiguration": {
  "hostname": "tomcat",
  "httpConfiguration": {
    "httpBoundOnlyToLocalhost": false
  },
  "httpsEnabled": false,
  "port": 8081,
  "shutdownPort": 8010,
  "usingSelfSignedCertificate": false
}
```

Also, update the runningMode as ScaledDistributed.

```
"runningMode": "ScaledDistributed",
"solution": {
  ...
```

In the interface\_application.env file, update the VM IP Address.

```
export DRIVE_DB_IP_ADDRESS="<VM IP address>"
export NAMESERVICE_SERVER_IP="<VM IP address>"
export EXTERNAL_KAFKA_BROKERS="<VM IP address>:9092"...
```

Similarly for the runtime\_environment.json file, update the network values as shown below:

```
"network": {
  "interface": "eth0",
  "ipV6Available": false,
  "usingIPv4": true
},
```

## Tagging Docker Images

Contact with the HCL support team to get the images for tomcat (HCL\_Detect\_x.x.x\_Docker\_Tomcat\_Image.zip) and feed (HCL\_Detect\_x.x.x\_Docker\_Feed\_Image.zip). Unzip both the images, and tag the images with required name. After tagging the images, push the images to the repository. For more information of tagging docker images, refer <https://docs.docker.com/reference/cli/docker/image/tag/>.

```
docker load -input HCL_Detect_Feed_12.1.8_Docker.tar
docker tag <imageid> <gcr repo tag>
docker push <gcr repo tag>
```

After performing the loading and tagging of docker images, in the values.yaml file, update the volumeName with the pvc name like shown below.

```
pvc:
  volumeName: ced-pvc
```

In the values.yaml file of helm charts, update the timezonevalue wrt. the timezone of the VM, where services like fastpast, pinpoint, nameserver, are deployed. For example, update the "timezonevalue:" as "EST5EDT" to set the US Eastern Timezone.

## Deploying Detect Services

Using the below Helm command, install the Detect services on the OpenShift cluster.

```
helm install detect .
```

As a result, the output will be as shown below.

```

PS C:\Openshift_cluster_deployment\helm> helm install detect .
NAME: detect
LAST DEPLOYED: Mon Aug  5 23:39:12 2024
NAMESPACE: hclced
STATUS: deployed
REVISION: 1
TEST SUITE: None
PS C:\Openshift_cluster_deployment\helm> oc get pods
NAME                                READY   STATUS    RESTARTS   AGE
campaign-actuator-feed-767cb76bd9-mqdhq  1/1     Running   0           99s
customer-profile-refresh-feed-b74f465cd-r4vhr  1/1     Running   0           99s
ericsson-usage-feed-84865f9b95-gws5v        1/1     Running   0           99s
recharge-feed-c8bf6557-jfzbv               1/1     Running   0           99s
segment-updater-feed-84d6f6b48b-m7bs5       1/1     Running   0           99s
sharedpod                                  1/1     Running   0           14d
tomcat-84c8f8b4c8-ntbvv                    1/1     Running   0           99s
topup-demo-feed-54549545bf-78znh           1/1     Running   0           99s
troubleshoot-5bd649cd5c-ztbzf              1/1     Running   0           99s

```

# Chapter 3. Running HCL Detect

## Starting and stopping HCL Detect

Starting HCL Detect requires two steps.

First, we must ensure that the environment of the current shell is properly configured. This is accomplished by *invoking* the script that performs this task:

```
$ ./<installation-location>/drive/bin/environment_setter
```



### Note:

*Modifying the PATH environment variable in the user's .bashrc file*

The `environment_setter` script loads the user's `.bashrc` file (if it exists) and, subsequently, validates that the execution environment is not modified inadvertently with respect to the needs of HCL Detect.

Specifically, the `environment_setter` validates that the executable for the Java Virtual Machine (`java`) and the Python interpreter (`python`) are consistent with the ones needed by HCL Detect.

In case an inconsistency is flagged due to a modification of the user's `PATH` environment variable in `.bashrc`, an error message will be emitted and two corrective alternatives are possible.

If it happens to be an irrelevant/unintended misconfiguration, you can simply eliminate it from your `.bashrc` file.

If the original behavior needs to be preserved, this can be accomplished by conditionally making the following modification to the `PATH` environment variable:

```
if [[ -z ${HCL_PRODUCT:-} ]]; then
    export PATH=<path_to_custom_python>:<path_to_custom_jdk>:$PATH
fi
```

`HCL_PRODUCT` is an environment variable defined by HCL Detect in the new instance of the Bash shell the `environment_setter` starts.

By checking for this environment variable and conditionally updating the `PATH` variable only when it is not defined, the conflicting settings will not affect HCL Detect's runtime environment. Yet, when a regular shell is instantiated, those `PATH` settings will still remain active.

To ensure HCL Detect services are running on IPv4, in the `runtime_environment.json` file available in the `drive/etc` folder, update the network section as shown below:

- "ipV6Available": false,
- "usingIpV4": true

Next, we start all of the HCL Detect services:

```
$ <installation-location>/drive/bin/services_manager start
```

At this point, it is possible to open a browser on any computer with access to the server where HCL Detect is installed and access its web user interface by pointing the browser to the appropriate URL: `https://HCL.acme.com/drive`.

If everything has worked correctly, you will be presented with the initial setup workflow, whereby an administration password will be configured as well as other initialization steps.

If you cannot access the HCL Detect UI, the [Testing and troubleshooting the web proxy installation on page 3](#) section provides a few troubleshooting instructions that will help you ensure that your configuration is correct.

Finally, we can stop the HCL Detect services:

```
$ <installation-location>/drive/bin/services_manager stop
```

## Advanced options when starting and stopping Detect

HCL Detect can be started using the `services_manager` script located at `$HCL_HOME/bin`.

This script can be used to configure various parameters associated with the execution of the HCL Detect platform. A list of available parameters can be obtained as follows:

```
$ cd $HCL_HOME
$ ./bin/services_manager --help

usage: services_manager [-h] [-a] [-c path] [-i id] [-k] [-o [app [app ...]]]
                        [-r] [-p] [-s] [-t] [-u uri]
                        operation

The 'services_manager' application is part of HCL Detect 1.2.0 (build id:
HCLadmin@ubdev.HCL.com - commit id:
465f0374ffc53c20139e54ed9575ef81505c8ca3)

positional arguments:
  operation              specifies the operation to be performed, one of:
                        'start_applications', 'start', 'stop', 'check',
                        'stop_applications'

optional arguments:
  -h, --help            show this help message and exit
  -a, --disable-fastpast
                        indicates that FastPast should not be checked (when
                        checking the status of Detect services), started (when
                        starting the Detect services), or stopped (when
                        stopping the Detect services) (default: False)
  -c path, --drive-config-file path
                        specifies the path of the HCL Detect
                        configuration file (default: None)
  -i id, --instance-id id
                        specifies the instance identifier to use (default:
                        None)
  -k, --disable-kafka
                        indicates that Kafka should not be checked (when
                        checking the status of the services), started (when
                        starting the services), or stopped (when stopping the
                        services) (default: False)
  -o [app [app ...]], --applications-to-run [app [app ...]]
```

```

specifies the list of Detect applications to use (by
default, all of them are used)
-r, --disable-pinpoint
    indicates that PinPoint should not be checked (when
    checking the status of the services), started (when
    starting the services), or stopped (when stopping the
    services) (default: False)
-p, --disable-campaign-actuator
    indicates that Campaign Actuator should not be checked
    (when checking the status of the services), started
    (when starting the services), or stopped (when
    stopping the services) (default: True)
-s, --disable-segment-updater
    indicates that Segment Updater should not be checked
    (when checking the status of the services), started
    (when starting the services), or stopped (when
    stopping the services) (default: False)
-t, --disable-tomcat
    indicates that Tomcat should not be checked (when
    checking the status of the services), started (when
    starting the services), or stopped (when stopping the
    services) (default: False)
-u uri, --instance-uri uri
    specifies the instance URI to use (default: None,
    indicating that a self-managed Name Service will be
    started and used)

```

The user responsible for managing HCL Detect (typically `HCLadmin`) can start HCL Detect using the following command:

```

$ ./bin/services_manager start
Starting Name Service
--- running as process id 27981
--- instance URI is 'redis://10.0.2.15:34549/HCLadmin'
Starting Kafka
--- running Kafka as process id 27991
--- running Zookeeper as process id 27983
Starting FastPast
--- running as process id 27994
--- waiting for FastPast to initialize (done)
Starting PinPoint
--- running as process id 27995
--- waiting for PinPoint to initialize (done)
Starting Tomcat
--- running as process id 28051
--- waiting for HCL Drive to initialize..... (done)
Starting application 'Segment Updater'
--- running as process id 28181
Starting feed applications: 'Topup Demo', 'Mobile Usage'
Starting application 'Topup Demo'
--- running as process id 28193
Starting application 'Mobile Usage'
--- running as process id 28209

```

After its execution, the HCL Detect web-based user interface is accessible by opening the `http://<hostname>:8080/Detect` URL in a browser.

HCL Detect can be stopped by invoking the following command:



```

$ ./bin/services_manager stop
Stopping feed applications: 'Topup Demo', 'Mobile Usage'
Stopping application 'Topup Demo'
--- waiting for application 'Topup Demo' to terminate.
--- stopped process id 28193
Stopping application 'Mobile Usage'
--- waiting for application 'Mobile Usage' to terminate.
--- stopped process id 28209
Stopping application 'Segment Updater'
--- waiting for application 'Segment Updater' to terminate.
--- stopped process id 28181
Stopping Tomcat
--- waiting for Tomcat to terminate.
--- stopped process id 28051
Stopping PinPoint
--- waiting for PinPoint to shutdown its servers.
--- waiting for PinPoint to terminate.
--- stopped process id 27995
Stopping FastPast
--- waiting for FastPast to shutdown its servers.
--- waiting for FastPast to terminate.
--- stopped process id 27994
Stopping Kafka
--- waiting for Kafka to terminate.
--- stopped process id 27991
--- waiting for Zookeeper to terminate.
--- stopped process id 27983
Stopping the Name Service
--- stopped process id 27981

```

Once the stop sequence is complete, the web-based user interface is not available anymore.

The sample `services_manager` invocation we used earlier starts up all applications available as part of your installation. The HCL Detect administrator user can, of course, make use of the other parameters associated with the script. For instance, in order to start the services only with a single application named `Topup Demo` (assuming it is available as part of your installation), the following command can be used:

```

$ ./bin/services_manager start -o 'Topup Demo'
Starting Name Service
--- running as process id 28730
--- instance URI is 'redis://10.0.2.15:57173/HCLadmin'
Starting Kafka
--- running Kafka as process id 28741
--- running Zookeeper as process id 28735
Starting FastPast
--- running as process id 28743
--- waiting for FastPast to initialize (done)
Starting PinPoint
--- running as process id 28744
--- waiting for PinPoint to initialize (done)
Starting Tomcat
--- running as process id 28798
--- waiting for HCL Drive to initialize..... (done)
Starting application 'Segment Updater'
--- running as process id 28896
Starting feed applications: 'Topup Demo'

```

```
Starting application 'Topup Demo'
--- running as process id 28903
```

While the detailed status of the HCL Detect services can be examined via the web-based user interface, the `services_manager` script can also be used to get a brief status of all services. An example invocation is as follows:

```
$ ./bin/services_manager check
Kafka is UP
Zookeeper is UP
FastPast is UP
PinPoint is UP
Tomcat is UP
application 'Segment Updater' is UP
application 'Topup Demo' is UP
application 'Mobile Usage' is DOWN
```

While the services are up, the `services_manager` script can be used to start and stop individual applications. For instance, the following command can be used to start the 'Mobile Usage' application:

```
$ ./bin/services_manager start_applications -o 'Mobile Usage'
Starting feed applications: 'Mobile Usage'
Starting application 'Mobile Usage'
--- running as process id 29058
```

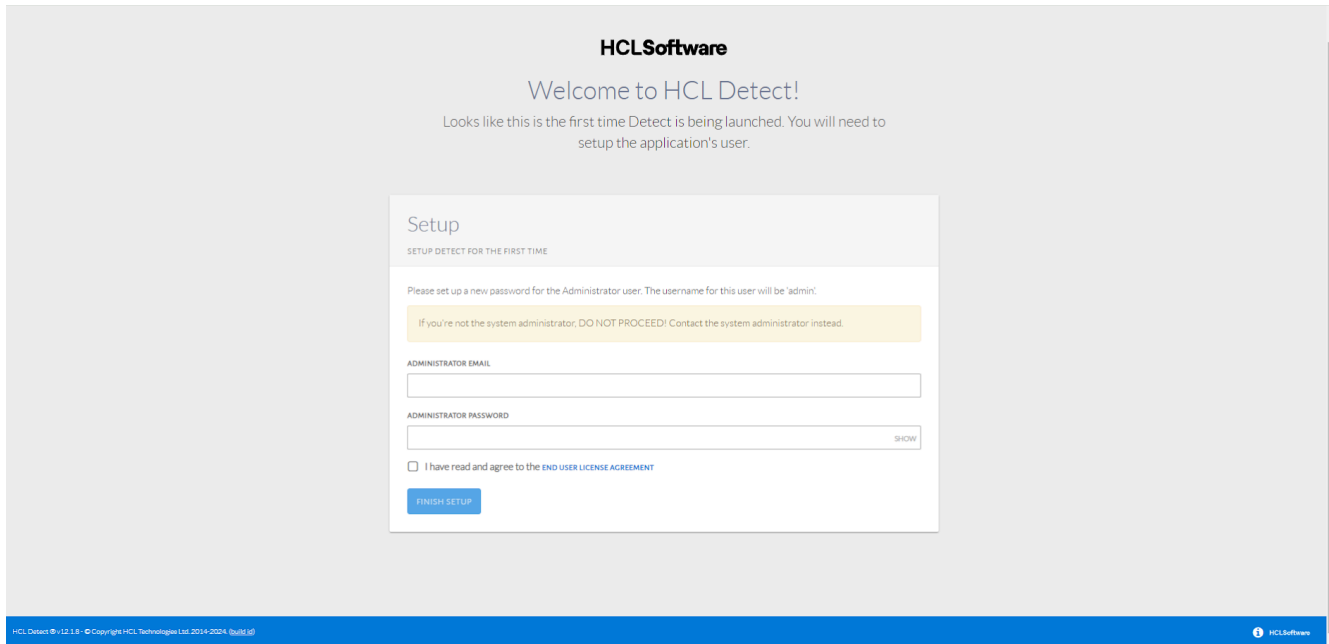
A running application can be stopped in a similar way. For instance, the following command can be used to stop the 'Mobile Usage' application:

```
$ ./bin/services_manager stop_applications -o 'Mobile Usage'
Stopping feed applications: 'Mobile Usage'
Stopping application 'Mobile Usage'
--- waiting for application 'Mobile Usage' to terminate.
--- stopped process id 29058
```

## Chapter 4. Accessing the interface for the first time

The preferred browser for accessing the user interface is [Google's Chrome](#).

When accessing the user interface for the first time after the software installation, the user will be presented with a screen to setup the `admin` password as seen below:



The screenshot shows the HCL Detect Setup interface. At the top, it says "HCLSoftware" and "Welcome to HCL Detect!". Below that, a message states: "Looks like this is the first time Detect is being launched. You will need to setup the application's user." The main section is titled "Setup" and "SETUP DETECT FOR THE FIRST TIME". It instructs the user to "Please set up a new password for the Administrator user. The username for this user will be 'admin'." A yellow warning box says: "If you're not the system administrator, DO NOT PROCEED! Contact the system administrator instead." There are two input fields: "ADMINISTRATOR EMAIL" and "ADMINISTRATOR PASSWORD". The password field has a "SHOW" button next to it. Below the fields is a checkbox labeled "I have read and agree to the END USER LICENSE AGREEMENT". At the bottom of the form is a blue "FINISH SETUP" button. The footer of the page shows "HCL Detect © v1.2.1.5 - © Copyright HCL Technologies Ltd. 2014-2024. (b) (6) (d)" and the HCLSoftware logo.

Setting up the `admin` password.

This should only be done by the user responsible for managing HCL Detect.

The `admin` login provides unrestricted administrative access to HCL Detect and, among other tasks, can be used to add additional users.