

# **Continuity Reactor 2.2.1**

## **Installation and Configuration Guide**

# Contents

<b>Introduction</b>	<b>3</b>
Conventions	3
<b>System Requirements</b>	<b>4</b>
Hardware Requirements	4
Network Requirements	4
Software Prerequisites	5
Java Runtime	5
Node.js Runtime	5
Hadoop/HBase Environment	5
Prepare the Cluster	6
ULIMIT Configuration	6
<b>Packaging</b>	<b>7</b>
RPM using Yum	7
Debian using APT	8
<b>Installation</b>	<b>9</b>
<b>Verification</b>	<b>10</b>
<b>Troubleshooting</b>	<b>11</b>
Application Won't Start	11
No Metrics/logs	11
Only the First Flowlet Showing Activity	11
YARN Application Shows ACCEPTED For Some Time But Then Fails	11
<b>Appendix: <code>continuity-site.xml</code></b>	<b>12</b>

# Introduction

This guide is to help you install and configure Continuity Reactor. It provides the [system](#), [network](#), and [software requirements](#), [packaging options](#), and instructions for [installation](#) and [verification](#) of the Continuity Reactor components so they work with your existing Hadoop cluster.

These are the Continuity Reactor components:

- **Continuity Web-App:** User interface—the *Dashboard*—for managing Continuity Reactor applications;
- **Continuity Gateway:** Service supporting REST endpoints for Continuity Reactor;
- **Continuity Reactor-Master:** Service for managing runtime, lifecycle and resources of Reactor applications; and
- **Continuity Kafka:** Metrics and logging transport service, using an embedded version of *Kafka*.

Before installing the Continuity Reactor components, you must first install a Hadoop cluster with *HDFS*, *YARN*, *HBase*, and *Zookeeper*. All Reactor components can be installed on the same boxes as your Hadoop cluster, or on separate boxes that can connect to the Hadoop services.

Our recommended installation is to use two boxes for the Reactor components; the [hardware requirements](#) are relatively modest, as most of the work is done by the Hadoop cluster. These two boxes provide high availability; at any one time, one of them is the leader providing services while the other is a follower providing failover support.

Some Reactor components run on YARN, while others orchestrate the Hadoop cluster. The Continuity Gateway service starts a router instance on each of the local boxes and instantiates one or more gateway instances on YARN as determined by the gateway service configuration.

We have specific [hardware](#), [network](#) and [prerequisite software](#) requirements detailed [below](#) that need to be met and completed before installation of the Continuity Reactor components.

## Conventions

In this document, *client* refers to an external application that is calling the Continuity Reactor using the HTTP interface.

*Application* refers to a user Application that has been deployed into the Continuity Reactor.

Text that are variables that you are to replace is indicated by a series of angle brackets (< >). For example:

```
https://<username>:<password>@repository.continuity.com
```

indicates that the texts <username> and <password> are variables and that you are to replace them with your values, perhaps username `john_doe` and password `BigData11`:

```
https://john_doe:BigData11@repository.continuity.com
```

# System Requirements

## Hardware Requirements

Systems hosting the Continuity Reactor components must meet these hardware specifications, in addition to having CPUs with a minimum speed of 2 GHz:

Continuity Component	Hardware Component	Specifications
<b>Continuity Web-App</b>	RAM	1 GB minimum, 2 GB recommended
<b>Continuity Gateway</b>	RAM	2 GB minimum, 4 GB recommended
<b>Continuity Reactor-Master</b>	RAM	2 GB minimum, 4 GB recommended
<b>Continuity Kafka</b>	RAM	1 GB minimum, 2 GB recommended
	Disk Space	<i>Continuity Kafka</i> maintains a data cache in a configurable data directory. Required space depends on the number of Continuity applications deployed and running in the Continuity Reactor and the quantity of logs and metrics that they generate.

## Network Requirements

Continuity components communicate over your network with *HBase*, *HDFS*, and *YARN*. For the best performance, Continuity components should be located on the same LAN, ideally running at 1 Gbps or faster. A good rule of thumb is to treat Continuity components as you would *Hadoop DataNodes*.

## Software Prerequisites

You'll need this software installed:

- Java runtime (on Reactor and Hadoop nodes)
- Node.js runtime (on Reactor nodes)
- Hadoop/HBase environment to run against

### Java Runtime

The latest [JDK or JRE version 1.6.xx](#) for Linux and Solaris must be installed in your environment.

Once you have installed the JDK, you'll need to set the `JAVA_HOME` environment variable.

### Node.js Runtime

You can download the latest version of Node.js from [nodejs.org](http://nodejs.org), using any of the methods given.

Using Yum:

```
$ curl -O http://download-i2.fedoraproject.org/pub/epel/6/i386/epel-release-6-8.noarch.rpm
$ sudo rpm -ivh epel-release-6-8.noarch.rpm
$ sudo yum install npm
```

Using APT:

```
$ sudo apt-get install npm
```

### Hadoop/HBase Environment

For a distributed enterprise, you must install these Hadoop components:

Component	Distribution	Required Version
<b>HDFS</b>	Apache Hadoop DFS	2.0.2-alpha or later
	CDH	4.2.x or later
	HDP	2.0 or later
<b>YARN</b>	Apache Hadoop DFS	2.0.2-alpha or later
	CDH	4.2.x or later
	HDP	2.0 or later
<b>HBase</b>		0.94.2+ or 0.96.0+
<b>Zookeeper</b>		Version 3.4.3 or later

Reactor nodes require Hadoop and HBase client installation and configuration. No Hadoop services need to be running.

Certain Continuuity components need to reference your *Hadoop*, *HBase*, and *YARN* cluster configurations by adding your configuration to their classpaths.

## Prepare the Cluster

To prepare your cluster so that Continuity Reactor can write to its default namespace, create a top-level `/continuity` directory in HDFS, owned by an HDFS user `yarn`:

```
hadoop fs -mkdir /continuity && hadoop fs -chown yarn /continuity
```

In the Continuity Reactor packages, the default HDFS namespace is `/continuity` and the default HDFS user is `yarn`. If you set up your cluster as above, no further changes are required.

If you want to use an HDFS directory with a name other than `/continuity`:

- Create the HDFS directory you want to use, such as `/myhadoop/myspace`.
- Create an xml file `conf/continuity-site.xml` (see appendix) and include in it an `hdfs.namespace` property for the HDFS directory:

```
<configuration>
...
<property>
<name>hdfs.namespace</name>
<value>/myhadoop/myspace</value>
<description>Default HDFS namespace</description>
</property>
...
```

- Ensure that the default HDFS user `yarn` owns that HDFS directory.

If you want to use a different HDFS user than `yarn`:

- Check that there is—and create if necessary—a corresponding user on all machines in the cluster on which YARN is running (typically, all of the machines).
- Create an `hdfs.user` property for that user in `conf/continuity-site.xml`:

```
<configuration>
...
<property>
<name>hdfs.user</name>
<value>my_username</value>
<description>User for accessing HDFS</description>
</property>
...
```

- Check that the HDFS user owns the HDFS directory described by `hdfs.namespace` on all machines.

## ULIMIT Configuration

When you install the Continuity Reactor packages, the `ulimit` settings for the Continuity user are specified in the `/etc/security/limits.d/continuity.conf` file. On Ubuntu, they won't take effect unless you make changes to the `/etc/pam.d/common-session` file. For more information, refer to the [ulimit discussion in the Apache HBase Reference Guide](#).

# Packaging

Continuuity components are available as either Yum `.rpm` or APT `.deb` packages. There is one package for each Continuuity component, and each component may have multiple services. Additionally, there is a base Continuuity package with two utility packages installed which creates the base configuration and the `continuity` user. We provide packages for *Ubuntu 12* and *CentOS 6*.

Available packaging types:

- RPM: YUM repo
- Debian: APT repo
- Tar: For specialized installations only

Continuuity packages utilize a central configuration, stored by default in `/etc/continuity`.

When you install the Continuuity base package, a default configuration is placed in `/etc/continuity/conf.dist`. The `continuity-site.xml` file is a placeholder where you can define your specific configuration for all Continuuity components.

Similar to Hadoop, Continuuity utilizes the `alternatives` framework to allow you to easily switch between multiple configurations. The `alternatives` system is used for ease of management and allows you to choose between different directories to fulfill the same purpose.

Simply copy the contents of `/etc/continuity/conf.dist` into a directory of your choice (such as `/etc/continuity/conf.myreactor`) and make all of your customizations there. Then run the `alternatives` command to point the `/etc/continuity/conf` symlink to your custom directory.

## RPM using Yum

Create a file `continuity.repo` at the location:

```
/etc/yum.repos.d/continuity.repo
```

The RPM packages are accessible using Yum at this authenticated URL:

```
[continuity]
name=Continuity Reactor Packages
baseurl=https://<username>:<password>@repository.continuity.com/content/groups/restricted
enabled=1
protect=0
gpgcheck=0
metadata_expire=30s
autorefresh=1
type=rpm-md
```

**where:**

- <username>:** Username provided by your Continuuity.com representative
- <password>:** Password provided by your Continuuity.com representative

## Debian using APT

Debian packages are accessible via APT on *Ubuntu 12*.

Create a file `continuity.list` at the location:

```
/etc/apt/sources.list.d/continuity.list
```

Use this authenticated URL (one line):

```
deb [ arch=amd64 ] https://<username>:<password>@repository.continuity.com/content/sites/apt  
precise release
```

**where:**

**<username>:** Username provided by your Continuity.com representative

**<password>:** Password provided by your Continuity.com representative



# Installation

Install the Continuuity Reactor packages by using either of these methods:

Using Yum (on one line):

```
sudo yum install continuuity-gateway continuuity-kafka continuuity-reactor-master  
continuity-web-app
```

Using APT (on one line):

```
sudo apt-get install continuuity-gateway continuuity-kafka continuuity-reactor-master  
continuity-web-app
```

Do this on each of the boxes that are being used for the Reactor components; our recommended installation is a minimum of two boxes.

This will download and install the latest version of Continuuity Reactor with all of its dependencies. When all the packages and dependencies have been installed, you can start the services on each of the Reactor boxes by running this command:

```
for i in `ls /etc/init.d/ | grep continuuity` ; do service $i restart ; done
```

When all the services have completed starting, the Continuuity Web-App should then be accessible through a browser at port 9999. The URL will be `http://<app-fabric-ip>:9999` where `<app-fabric-ip>` is the IP address of one of the machine where you installed the packages and started the services.

## Verification

To verify that the Continuity software is successfully installed and you are able to use your Hadoop cluster, run an example application. We provide in our SDK pre-built .JAR files for convenience:

1. Download and install the latest Continuity Developer Suite from <http://accounts.continuity.com>.
2. Extract to a folder (CONTINUITY\_HOME).
3. Open a command prompt and navigate to CONTINUITY\_HOME/examples.
4. Each example folder has in its target directory a .JAR file. For verification, we will use the TrafficAnalytics example.
5. Open a web browser to the Continuity Reactor Web-App ("Dashboard"). It will be located on port 9999 of the box where you installed Reactor.
6. On the Dashboard, click the button *Load an App*.
7. Find the pre-built JAR (*TrafficAnalytics-1.0.jar*) by using the dialog box to navigate to CONTINUITY\_HOME/examples/TrafficAnalytics/target/TrafficAnalytics-1.0.jar
8. Once the application is deployed, instructions on running the example can be found at the [TrafficAnalytics example](#).
9. You should be able to start the application, inject log entries, run the MapReduce job and see results.
- 10 When finished, stop and remove the application as described in the [TrafficAnalytics example](#).

# Troubleshooting

Here are some selected examples of potential problems and possible resolutions.

## Application Won't Start

Check HDFS write permissions. It should show an obvious exception in the YARN logs.

## No Metrics/logs

Make sure the *Kafka* server is running, and make sure local the logs directory is created and accessible. On the initial startup, the number of available seed brokers must be greater than or equal to the *Kafka* default replication factor.

In a two-box setup with a replication factor of two, if one box fails to startup, metrics will not show up though the application will still run:

```
[2013-10-10 20:48:46,160] ERROR [KafkaApi-1511941310]  
    Error while retrieving topic metadata (kafka.server.KafkaApis)  
    kafka.admin.AdministrationException:  
        replication factor: 2 larger than available brokers: 1
```

## Only the First Flowlet Showing Activity

Check that YARN has the capacity to start any of the remaining containers.

## YARN Application Shows ACCEPTED For Some Time But Then Fails

It's possible that YARN can't extract the .JARs to the `/tmp`, either due to a lack of disk space or permissions.

## Appendix: `continuity-site.xml`

Here are the parameters that can be defined in the `continuity-site.xml` file, their default values, descriptions and notes.

Parameter name	Default Value	Description
<code>app.bind.address</code>	<code>127.0.0.1</code>	App-Fabric server host address
<code>app.bind.port</code>	<code>45000</code>	App-Fabric server port
<code>app.command.port</code>	<code>45010</code>	App-Fabric command port
<code>app.output.dir</code>	<code>/programs</code>	Directory where all archives are stored
<code>app.program.jvm.opts</code>	<code>\${weave.jvm.gc.opts}</code>	Java options for all program containers
<code>app.temp.dir</code>	<code>/tmp</code>	Temp directory
<code>dashboard.bind.port</code>	<code>9999</code>	Dashboard bind port
<code>data.local.storage</code>	<code>\${local.data.dir}/ldb</code>	Database directory
<code>data.local.storage.blocks.ize</code>	<code>1024</code>	Block size in bytes
<code>data.local.storage.caches.ize</code>	<code>104857600</code>	Cache size in bytes
<code>data.queue.config.update.interval</code>	<code>5</code>	Frequency, in seconds, of updates to the queue consumer
<code>data.queue.table.name</code>	<code>queues</code>	Tablename for queues
<code>data.tx.bind.address</code>	<code>127.0.0.1</code>	Transaction Inet address
<code>data.tx.bind.port</code>	<code>15165</code>	Transaction bind port
<code>data.tx.client.count</code>	<code>5</code>	Number of pooled transaction instances
<code>data.tx.client.provider</code>	<code>thread-local</code>	Provider strategy for transaction clients
<code>data.tx.command.port</code>	<code>15175</code>	Transaction command port number
<code>data.tx.janitor.enable</code>	<code>True</code>	Whether or not the TransactionDataJanitor coprocessor
<code>data.tx.server.io.threads</code>	<code>2</code>	Number of transaction IO threads
<code>data.tx.server.threads</code>	<code>25</code>	Number of transaction threads
<code>data.tx.snapshot.dir</code>	<code>\${hdfs.namespace}/tx.snapshot</code>	Directory in HDFS used to store snapshots and transaction logs
<code>data.tx.snapshot.interval</code>	<code>300</code>	Frequency of transaction snapshots in seconds
<code>data.tx.snapshot.local.dir</code>	<code>\${local.data.dir}/tx.snapshot</code>	Snapshot storage directory on the local filesystem

data.tx.snapshot.retain	10	Number of retained transaction snapshot files
enable.unrecoverable.reset	False	<b>WARNING: Enabling this option makes it possible to delete all applications and data; no recovery is possible!</b>
gateway.boss.threads	1	Number of Netty server boss threads
gateway.connection.backlog	20000	Maximum connection backlog of Gateway
gateway.exec.threads	20	Number of Netty server executor threads
gateway.max.cached.events.per.stream.num	5000	Maximum number of a single stream's events cached before flushing
gateway.max.cached.stream.events.bytes	52428800	Maximum size (in bytes) of stream events cached before flushing
gateway.max.cached.stream.events.num	10000	Maximum number of stream events cached before flushing
gateway.memory.mb	2048	Memory in MB for Gateway process in YARN
gateway.num.cores	2	Cores requested per Gateway container in YARN
gateway.num.instances	1	Number of Gateway instances in YARN
gateway.server.address	localhost	Router address to which Dashboard connects
gateway.server.port	10000	Router port to which Dashboard connects
gateway.stream.callback.executor.num.threads	5	Number of threads in stream events callback executor
gateway.stream.events.flush.interval.ms	150	Interval at which cached stream events get flushed
gateway.worker.threads	10	Number of Netty server worker threads
hdfs.lib.dir	\${hdfs.namespace}/lib	Common directory in HDFS for JAR files for coprocessors
hdfs.namespace	/\${reactor.namespace}	Namespace for files written by Reactor
hdfs.user	yarn	User name for accessing HDFS
kafka.bind.address	0.0.0.0	Kafka server hostname
kafka.bind.port	9092	Kafka server port

kafka.default.replication.factor	1	Kafka replication factor <a href="#">[Note 1]</a>
kafka.log.dir	/tmp/kafka-logs	Kafka log storage directory
kafka.num.partitions	10	Default number of partitions for a topic
kafka.seed.brokers	127.0.0.1:9092	Kafka brokers list (comma separated)
kafka.zookeeper.namespace	continuity_kafka	Kafka Zookeeper namespace
local.data.dir	data	Data directory for local mode
log.base.dir	/logs/avro	Base log directory
log.cleanup.run.interval.mins	1440	Log cleanup interval in minutes
log.publish.num.partitions	10	Number of Kafka partitions to publish the logs to
log.retention.duration.days	7	Log file HDFS retention duration in days
log.run.account	continuity	Logging service account
log.saver.num.instances	1	Log saver instances to run in YARN
metadata.bind.address	127.0.0.1	Metadata server address
metadata.bind.port	45004	Metadata server port
metadata.program.run.history.keepdays	30	Number of days to keep metadata run history
metrics.data.table.retention.resolution.1.seconds	7200	Retention resolution of the 1 second table in seconds
metrics.kafka.partition.size	10	Number of partitions for metrics topic
metrics.query.bind.address	127.0.0.1	Metrics query server host address
metrics.query.bind.port	45005	Metrics query server port
reactor.namespace	continuity	Namespace for this Reactor instance
router.bind.address	0.0.0.0	Router server address
router.client.boss.threads	1	Number of router client boss threads
router.client.worker.threads	10	Number of router client worker threads
router.connection.backlog	20000	Maximum router connection backlog
router.forward.rule	10000:gateway,20000:webapp/\$HOST	Router forward rules <a href="#">[Note 2]</a>

<code>router.server.boss.threads</code>	1	Number of router server boss threads
<code>router.server.worker.threads</code>	10	Number of router server worker threads
<code>scheduler.max.thread.pool.size</code>	30	Size of the scheduler thread pool
<code>stream.flume.port</code>	10004	
<code>stream.flume.threads</code>	20	
<code>thrift.max.read.buffer</code>	16777216	Maximum read buffer size in bytes used by the Thrift server <a href="#">[Note 3]</a>
<code>weave.java.reserved.memory.mb</code>	250	Reserved non-heap memory in MB for Weave container
<code>weave.jvm.gc.opts</code>	-verbose:gc -Xloggc:<log-dir>/gc.log -XX:+PrintGCDetails -XX:+PrintGCTimeStamps -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=10 -XX:GCLogFileSize=1M	Java garbage collection options for all Weave containers; <log-dir> is the location of the log directory on each machine
<code>weave.no.container.timeout</code>	120000	Amount of time in milliseconds to wait for at least one container for Weave runnable
<code>weave.zookeeper.namespace</code>	/weave	Weave Zookeeper namespace prefix
<code>yarn.user</code>	yarn	User name for running applications in YARN
<code>zookeeper.quorum</code>	127.0.0.1:2181/\${reactor.namespace}	Zookeeper address host:port
<code>zookeeper.session.timeout.millis</code>	40000	Zookeeper session time out in milliseconds

**Note 1:** `kafka.default.replication.factor` is used to replicate *Kafka* messages across multiple machines to prevent data loss in the event of a hardware failure. The recommended setting is to run at least two *Kafka* servers. If you are running two *Kafka* servers, set this value to 2; otherwise, set it to the number of *Kafka* servers

**Note 2:** This configuration has two rules:

1. Forward anything that comes on port 10000 to the service Gateway.
2. Forward anything that comes on port 20000 to `webapp/$HOST`, where `$HOST` is the host that the webapp wants to impersonate.

Example: `webapp/streamy.com` points to a webapp container running in YARN, with DNS set to point *streamy.com* to the router host. The router then forwards it to the webapp container in YARN.

**Note 3:** Maximum read buffer size in bytes used by the Thrift server: this value should be set to greater than the maximum frame sent on the RPC channel.