Gemfire Quick Start

Tim Dalsing

# Install Java

Gemfire requires at least JDK 1.7, but the latest 1.8 is recommended. There are some bug fixes in JDK 1.8 that affect Gemfire (especially leaking file descriptors in the NIO module). The JDK is recommended over the JRE. The JDK version used in development should match the version that will be used in other environments, including production.

The <jdk\_home>/bin directory must be in the PATH. The JAVA\_HOME environment variable is not required but it is recommended.

# Install Gemfire

Gemfire can be downloaded from the Pivotal Network: <https://network.pivotal.io/products/pivotal-gemfire>. For development the JARs are available via the Gemfire Maven repo: http://dist.gemstone.com/maven/release/

There are several ways to install Gemfire:

1. For Windows platforms the zip distribution can be downloaded and extracted to some directory. The <gemfire\_home>/bin directory should be in the PATH.
2. On OS/X and Linux hosts the tar distribution can be used. The <gemfire\_home>/bin directory should be in the PATH.
3. On Linux hosts an RPM distribution is available. The RPM will place the Gemfire shell scripts on the PATH.
4. On OS/X Gemfire can be installed via HomeBrew:

$ brew untap pivotal/tap

$ brew tap pivotal/tap

$ brew install gemfire

Verify that Gemfire is installed correctly:

$ gfsh version

v8.2.0

# Create a Working Directory

Create a directory somewhere that will contain the Gemfire locator and server files. Any directory that is writable by the current user is fine.

Start gfsh in interactive mode:

$ gfsh

gfsh>

# Start Locator

To start a locator from within gfsh interactive mode:

gfsh> start locator --name=locator

This will start a locator named “locator” that listens on port 10334. The logs and other files will reside in the ./locator directory.

# Start Server

To start a server:

gfsh> start server --name=server --locators=localhost[10334]

This will start a server named “server” that connects to the locator started earlier and listens on port 40404. The logs and data files reside in the ./server sub-directory.

# Configure Server

This section describes the typical configuration for a Gemfire cluster. This may vary depending on the exact needs of the application. In general PDX will always be used and should have it’s own disk store. Other settings depend on whether there is server-side code and whether domain classes are used.

When the PDX configuration is changed the server(s) must be restarted.

To configure Gemfire, first create a disk store for PDX:

gfsh> create disk-store --name=pdx --dir=./

Now configure PDX:

gfsh>configure pdx --disk-store=pdx --read-serialized=true

If there will be server-side code that uses domain classes, read-serialized should be false and the --auto-serializable-classes option must be used.

Now restart the server:

gfsh>shutdown

gfsh>start server --name=server --locators=localhost[10334]

Now verify that Gemfire is functioning properly:

gfsh>list members

Member Count : 2

Coordinator : locator (tdalsing-mbp(locator:54151:locator)<v0>:42289)

Name | Id

------- | ---------------------------------------------

locator | tdalsing-mbp(locator:54151:locator)<v0>:42289

server | tdalsing-mbp(server:54184)<v4>:55839

gfsh>describe config --member=server

Configuration of member : "server"

JVM command line arguments

---------------------------------------------------

-Dgemfire.locators=localhost[10334]

-Dgemfire.use-cluster-configuration=true

-XX:OnOutOfMemoryError=kill -KILL %p

-Dgemfire.launcher.registerSignalHandlers=true

-Djava.awt.headless=true

-Dsun.rmi.dgc.server.gcInterval=9223372036854775806

GemFire properties defined using the API

...........................................................

name : server

Cache attributes

...........................................................

is-server : true

pdx-disk-store : pdx

pdx-persistent : true

pdx-read-serialized : true

Cache-server attributes

. tcp-no-delay : true

Note that all of these commands can be put into a file and run all at once, via:

gfsh run --file=my\_quick\_start.txt

The only caveat is that sleep commands must be placed after the create disk-store and shutdown commands to allow the commands to fully complete.

# Spring Boot Client

The quickest and easiest way to create a Gemfire client application is via Spring Boot. Using Spring Boot is also the preferred way to build micro-services for Pivotal Cloud Foundry, so migrating a Spring Boot app to PCF is very simple. Built-in support for Gemfire micro-service apps is provided via the Spring Cloud connector for Gemfire/PCF.

This github repo has several examples of how to use Spring Boot with the Gemfire client: https://github.com/Pivotal-Field-Engineering/gemfire-fe