# Get Started with Hazelcast Jet

In this hands-on exercise, you will:

- Download and install Hazelcast Jet on your local device
- Create a simple pipeline and submit it to Jet in embedded mode
- Start a Jet cluster
- Submit a simple pipeline to the cluster in client/server mode
- Use Jet CLI tools

## Before You Start

Hazelcast Jet is a Java application and requires the Java Runtime Environment (JRE), version 8.0 or later. Make sure you have JRE installed. Download JRE

## Part 1: Acquire Hazelcast Jet

Choose one of the following options:

### Maven

Open a new Maven project in your preferred IDE. Add the following lines to your `pom.xml`

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>org.example</groupId>
    <artifactId>first-jet-program</artifactId>
    <version>1.0-SNAPSHOT</version>

    <name>first-jet-program</name>
    <properties>
      <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
      <maven.compiler.target>1.8</maven.compiler.target>
      <maven.compiler.source>1.8</maven.compiler.source>
    </properties>

    <dependencies>
      <dependency>
        <groupId>com.hazelcast.jet</groupId>
        <artifactId>hazelcast-jet</artifactId>
        <version>4.0</version>
      </dependency>
    </dependencies>

    <build>
      <plugins>
        <plugin>
          <groupId>org.apache.maven.plugins</groupId>
```

```
          <artifactId>maven-jar-plugin</artifactId>
          <version>3.2.0</version>
          <configuration>
            <archive>
              <manifest>
                <mainClass>org.example.JetJob</mainClass>
              </manifest>
            </archive>
          </configuration>
        </plugin>
      </plugins>
    </build>
</project>
```

## TAR

1. Browse to https://jet-start.sh/download.
2. Download the current version.
3. Extract the downloaded file.
4. Open a new project in your IDE. Add the file `hazelcast-jet-<version>.jar` to your classpath.

## Docker (Client/Server mode only)

Launch the Hazelcast Docker container.

```
$ docker run hazelcast/hazelcast-jet
```

*Note: If you use Docker, skip the Embedded Mode portion of this lab exercise.*

## Part 2: Embedded Mode

From your IDE, run the following code:

```
import com.hazelcast.jet.Jet;
import com.hazelcast.jet.JetInstance;
import com.hazelcast.jet.pipeline.Pipeline;
import com.hazelcast.jet.pipeline.Sinks;
import com.hazelcast.jet.pipeline.test.TestSources;

public class firstJetProgram {
    public static void main(String[] args) {


        Pipeline p = Pipeline.create();

        // source specifies where the streaming data comes from
        // this source is a generated sequence of increasing numbers
        p.readFrom(TestSources.itemStream(10))
                .withoutTimestamps()

        // tell Jet what to do with the streaming data
                .filter(event -> event.sequence() % 2 == 0)
                .setName("filter out odd numbers")

        // sink specifies where to send output
                .writeTo(Sinks.logger());

        // start embedded instance of Jet
        JetInstance jet = Jet.newJetInstance();

        // keep Jet JVM alive while job lasts
        jet.newJob(p).join();
    }
}
```

From your IDE run window, observe the startup sequence of the Hazelcast Jet instance, as well as the output printed by your code. Your output will continue to increment as long as your code is running because we're using Jet in embedded mode.

```
23:20:14.756 [ INFO] [c.h.j.i.MasterJobContext] Start executing job '0441-8809-ed40-
0001', execution 0441-8809-ed41-0001, execution graph in DOT format:
digraph DAG {
       "itemStream" [localParallelism=1];
       "filter out odd numbers" [localParallelism=12];
       "loggerSink" [localParallelism=1];
       "itemStream" -> "filter out odd numbers" [queueSize=1024];
       "filter out odd numbers" -> "loggerSink" [queueSize=1024];
}
HINT: You can use graphviz or http://viz-js.com to visualize the printed graph.
23:20:14.811 [ INFO] [c.h.j.i.JobExecutionService] Execution plan for jobId=0441-8809-
ed40-0001, jobName='0441-8809-ed40-0001', executionId=0441-8809-ed41-0001 initialized
23:20:14.815 [ INFO] [c.h.j.i.JobExecutionService] Start execution of job '0441-8809-
ed40-0001', execution 0441-8809-ed41-0001 from coordinator [192.168.10.115]:5701
23:20:14.822 [ INFO] [c.h.j.i.c.W.loggerSink#0] (timestamp=23:20:14.800, sequence=0)
23:20:15.023 [ INFO] [c.h.j.i.c.W.loggerSink#0] (timestamp=23:20:15.000, sequence=2)
23:20:15.229 [ INFO] [c.h.j.i.c.W.loggerSink#0] (timestamp=23:20:15.200, sequence=4)
23:20:15.427 [ INFO] [c.h.j.i.c.W.loggerSink#0] (timestamp=23:20:15.400, sequence=6)
23:20:15.626 [ INFO] [c.h.j.i.c.W.loggerSink#0] (timestamp=23:20:15.600, sequence=8)
23:20:15.825 [ INFO] [c.h.j.i.c.W.loggerSink#0] (timestamp=23:20:15.800, sequence=10)
```

# Part 3: Client/Server Mode – Start and Validate Jet

Starting and validating the Jet cluster will depend on the method you used to acquire Hazelcast Jet.

## TAR

1. Go to the `bin` directory and use the shell script `jet-start` (or `jet-start.bat` if you are using Windows) script to start an instance of Jet.

2. Use the pre-packaged "hello world" JAR to verify that Jet is working.
   `bin/jet submit examples/hello-world.jar`

## Docker

1. Use `$docker run hazelcast/hazelcast-jet` to start an instance of Jet. Note the IP address used by Docker for the Jet container.

```
Members {size:1, ver:1} [
     Member [172.17.0.2]:5701 - 87266884-c08b-4a58-a2db-0ea43cd6199f this
]

2020-04-27 20:45:13,191  INFO [c.h.c.LifecycleService] [main] - [172.17.0.2]:5701
[jet] [4.0] [172.17.0.2]:5701 is STARTED
```

2. Use the pre-packaged "hello world" JAR to verify that Jet is working.
   `docker run -it -v "$(pwd)"/examples:/examples hazelcast/hazelcast-jet jet -a 172.17.0.2 submit /examples/hello-world.jar`

For both methods, your output should look like this:

```
Submitting JAR 'examples/hello-world.jar' with arguments []

Top 10 random numbers in the latest window:
       1. 8,218,635,360,648,282,334
       2. 8,116,903,806,942,984,946
       3. 6,537,960,752,096,543,810
       4. 4,491,342,259,218,956,732
       5. 3,810,471,447,482,334,816
       6. 2,613,354,662,915,080,364
       7. 1,503,881,168,272,164,712
       8. 966,309,147,306,304,520
       9. 235,854,650,329,198,139
       10. -2,796,834,573,279,258,052

Top 10 random numbers in the latest window:
       1. 9,219,160,880,783,222,642
```

## Part 4: Submitting Your Job

1. Change your pipeline code as shown below.

```
        // start embedded instance of Jet
        //JetInstance jet = Jet.newJetInstance();
        // submit job to existing Jet instance (or start if no instance found)
        JetInstance jet = Jet.bootstrappedInstance();
```

2. Create a JAR for your code. Make sure you include the main class information in your build.

3. In a separate terminal window, use `jet submit` to submit your JAR to Jet. As we are using the logger as the sink, the output will appear in the server log file.

   local install: `$jet submit <path_to_JAR_file>`
   Docker: `$ docker run -it -v <path_to_JAR_file>:/jars hazelcast/hazelcast-jet jet -a 172.17.0.2 submit /jars/<name_of_the_JAR_file>`

   If you did not include the main class in your build, include it in the command line.

   local install: `$jet submit -c <main_class_name> <path_to_JAR_file>`
   Docker: `$ docker run -it -v <path_to_JAR_file>:/jars hazelcast/hazelcast-jet jet -c <main_class_name> -a 172.17.0.2 submit /jars/<name_of_the_JAR_file>`

   As we are using logger for our sink, the output will display in the server log window.

4. Use ^C in the client window to return to the prompt. Note that the output is still progressing in the server window. Once a job is submitted to Jet, it keeps running even if the client application stops.

5. Use the Jet cli commands to observe the cluster, job, suspend, and restart the job.

```
Hazelcast Jet 4.0
Usage: jet [-hvV] [-f=<config>] [-n=<clusterName>] [-a=<hostname>:<port>[,
           <hostname>:<port>...]...]... [COMMAND]
Utility to perform operations on a Hazelcast Jet cluster.
By default it uses the file config/hazelcast-client.yaml to configure the
client connection.

Global options are:

  -h, --help              Show this help message and exit.
  -V, --version           Print version information and exit.
  -f, --config=<config>   Optional path to a client config XML/YAML file. The
                            default is to use config/hazelcast-client.yaml. If you
                            set this option, Jet will ignore the --addresses and
                            --group options.
  -a, --addresses=<hostname>:<port>[,<hostname>:<port>...]...
                          Optional comma-separated list of Jet node addresses in
the
                            format <hostname>:<port>, if you want to connect to a
                            cluster other than the one configured in the
                            configuration file
  -v, --verbosity         Show logs from Jet client and full stack trace of errors
```

```
  -n, --cluster-name=<clusterName>
                          The cluster name to use when connecting to the cluster
                            specified by the <addresses> parameter.
                            Default: jet
Commands:
  help              Displays help information about the specified command
  cancel            Cancels a running job
  cluster           Shows current cluster state and information about members
  delete-snapshot   Deletes a named snapshot
  list-jobs         Lists running jobs on the cluster
  list-snapshots    Lists exported snapshots on the cluster
  restart           Restarts a running job
  resume            Resumes a suspended job
  save-snapshot     Exports a named snapshot from a job and optionally cancels it
  submit            Submits a job to the cluster
  suspend           Suspends a running job
```

## Part 5: Client/Server – Cluster Elasticity

1. With your job running, open another terminal window and launch another instance of Jet.

2. Use the Jet CLI commands to verify that the cluster is now running two nodes.

   *Note: The output only appears on one server because our code does not have any instructions for data rebalancing or output distribution. This topic is covered in the Stream Processing courses.*

3. Kill the first server. Note that your job output begins appearing on the second server automatically; the job is not disrupted by the server failure.

   *Note: The output restarts at 0 because we are not using a fault-tolerant source for this Getting Started exercise. The *source* resets to 0, but not the job. This topic is covered in the Stream Processing courses.*