HAZEL CAST

What are memory Grids:

In memory grid is a data structure that resides entirely in RAM and it is distributed among multiple servers.

Memory grids are:

- **Distributed**

  Suppose for example there are 3 nodes. They are Node1, Node2 and Node3

  Node1 has A,C,F

  Node2 has B

  Node3 has D,E,G

  A look up call comes for Node1 looking for data A. Data is present in Node1 itself. It finds and gives it back.

  A look up call comes on Node3 for data B. Data B is Node 2. So, Node3 will make a network call for Node 2. It looks up for B in Node2. It finds B. It gets back result and Node3 passes it back to the caller. It means that the data is distributed everywhere but memory grid takes care of looking up the data and giving back to the client.

- **Resilient(Fault tolerance)**

  Memory grids are capable of recovering from Node crashes. The way it does is by maintaining a copy of back up data into another Node.

  For example Node1 has A. Node 3 is keeping a backup of A as well. Node3 is undergoing out of memory exception and it crashed.

  Node1 and Node2 has the backup copy of the data present in Node3, we can rebuilt the cluster and whole cluster can be restored back.

- **Elastic**

  Suppose there are 2 nodes Node1 and Node2. They are nearing in terms of memory.

  We can start a new Node. Node 3 joins the cluster along with Node1 and Node2. It undergoes a process of rebalancing where the data is distributed into Node3 as well. This makes memory grid scalable. When the existing nodes reach the capacity, fire the new nodes and data is automatically distributed into the new nodes which we have added into the cluster

Hazel cast

It is an in-memory distributed caching mechanism. Hazel cast is for the jvms. It can be used in java.

Hazel cast has the persistence, scalability reliability. It has a data grid.

It has a lot of nodes connected in the grid and then the data is shared across the nodes and all these nodes and all these data is in memory persistence so all these are in the memory itself

It follows master less architecture. Every nodes which are a master for the particular request. If a node receives the request from the client that particular node will be the master. So, if a new request comes in there will be a different master.

For example, if one of the node goes down, the hazel cast framework or the architecture it automatically rebalances the grid. So, new nodes will be spawned or the other nodes will be synced up automatically.

We can add or remove nodes anytime from in the cluster on the grid. So, data partitions is done across the cluster so data will be stored in all the nodes in the cluster. We can have collections framework implemented

Backups are also distributed among the nodes to protect against the failure of any single node.

It provides central, predictable scaling of the applications through in memory access to frequently used data. These techniques reduce the querying load on the database and improve speed.

The major advantage is we can share data of a hash map across the JVM.

HOW?

We can create hazelcastInstance and we can access the hash map, put the hash map, get the hash map.

Hazel cast provides the data across the JVMs. So we can have one JVM with single hash map and other jvm can use

Messaging Channel (publisher and subscriber model):

We can publish messages from the hazel cast instance and then subscriber can consume messages by other.

Cluster Level Locks:

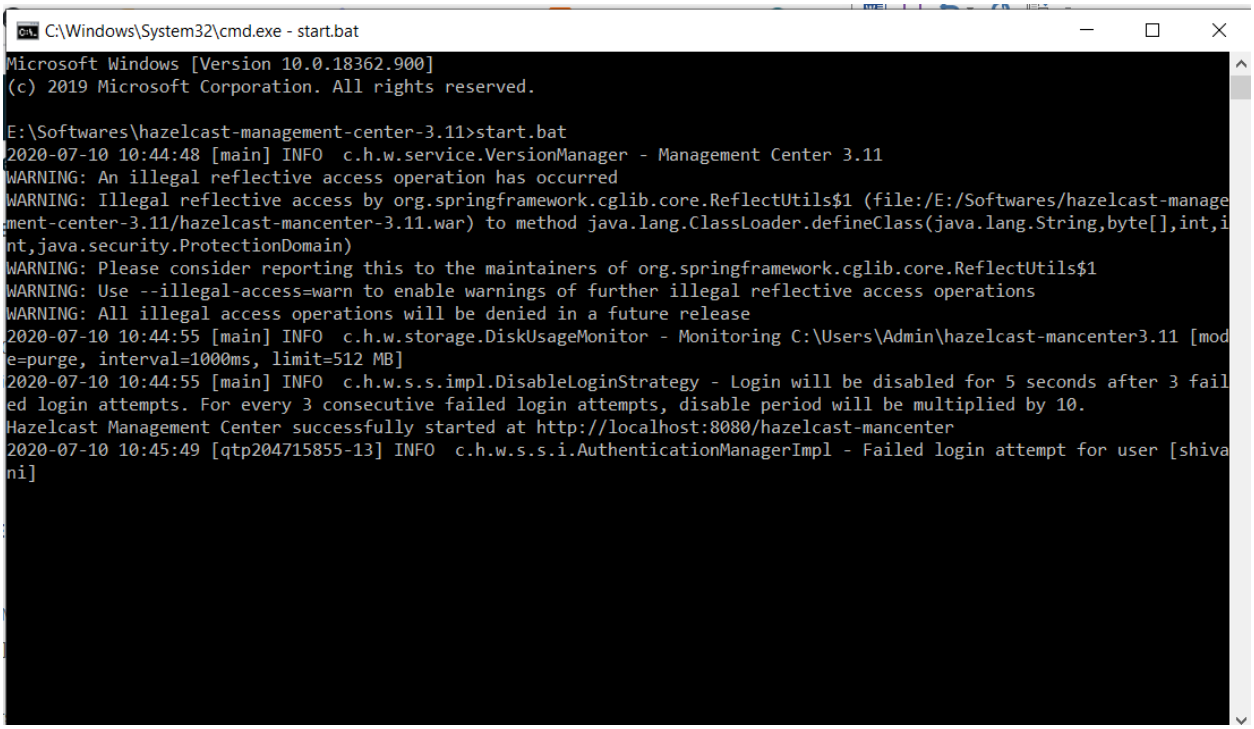
If we are writing data in a specific cluster then we can have cluster level lock so we can lock the particular cluster

Management Center:

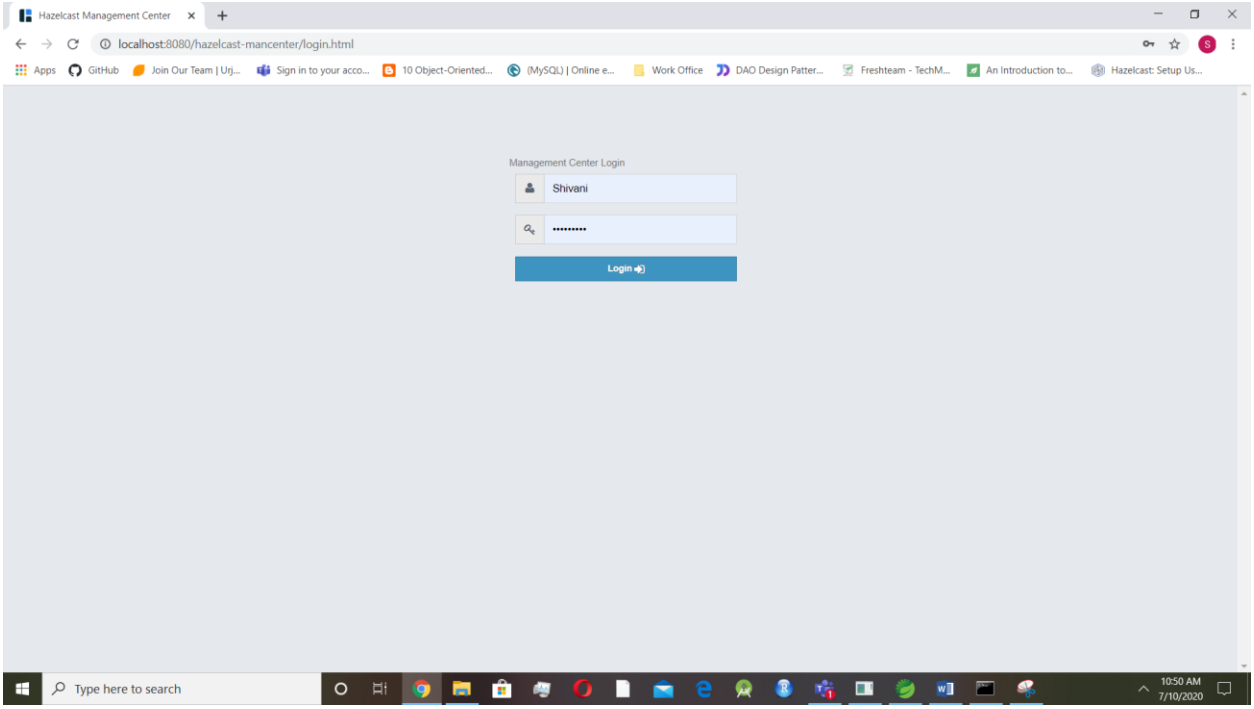
We can download it from hazelcast.

The link is **https://hazelcast.org/imdg/download/#hazelcast-imdg**

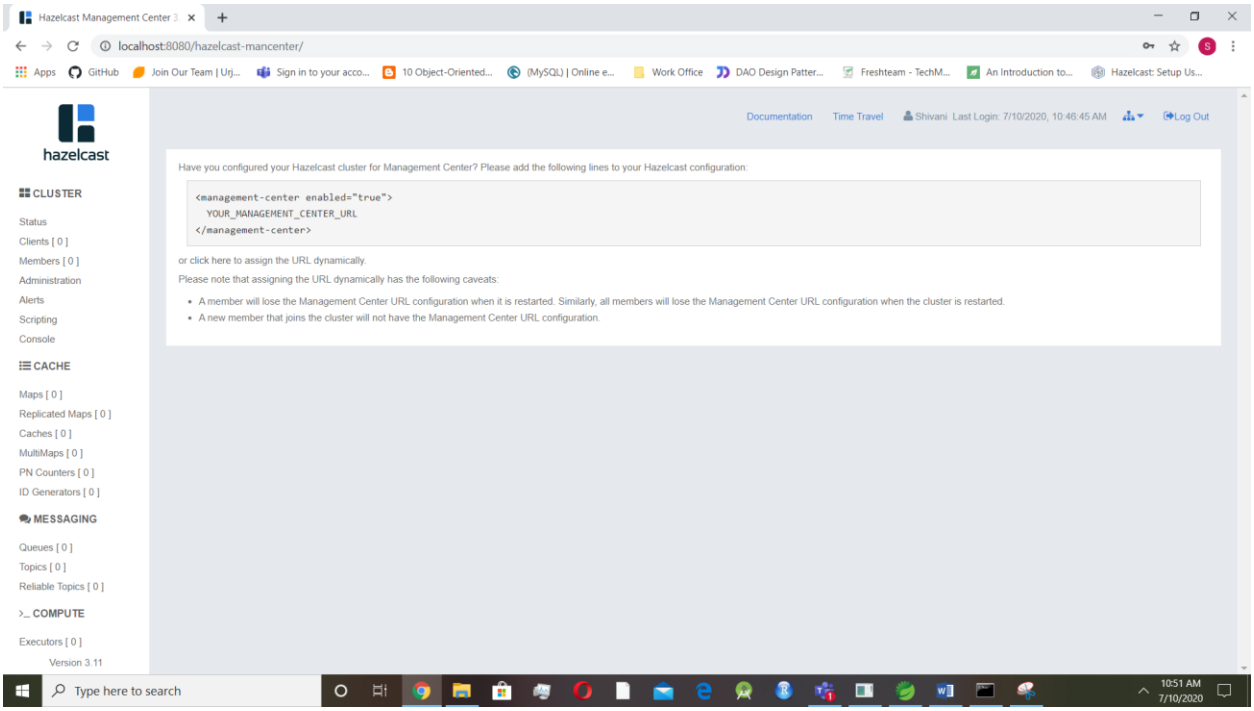
There will be two files. If its windows we can start with start.bat file on the command prompt.



This is how it looks. And then opens a new web console.



For the first time it asks to create user name and password. Then when we open it looks like below where in the left side we can see clusters, cache, and status.



SPRING BOOT APPLICATION:

In this application, writing from one port and reading from another port.

STEPS:

1. Add dependencies in pom.xml file: - hazelcast from mvn repository
2. Hazelcast Configuration
3. Then use the Map of Hazel cast which is IMap which works similar to the Map
4. We can check from postman reading and writing using multiple instances

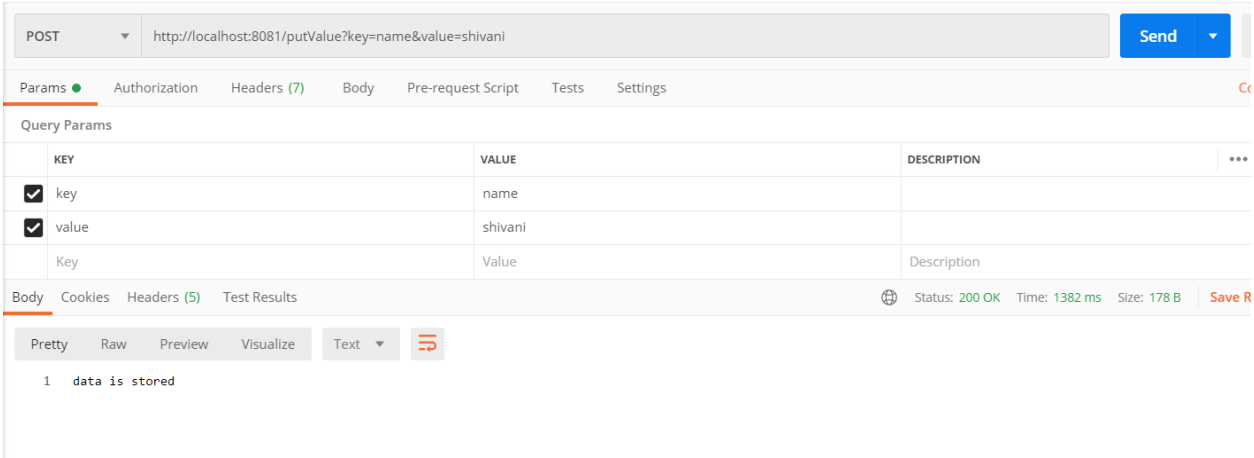
RESULT when we run for the first instance using port 8081:

```
Members {size:1, ver:1} [
        Member [192.168.245.1]:5701 - a778e7e2-a523-43fa-86ef-81ee1451573e this
]
```

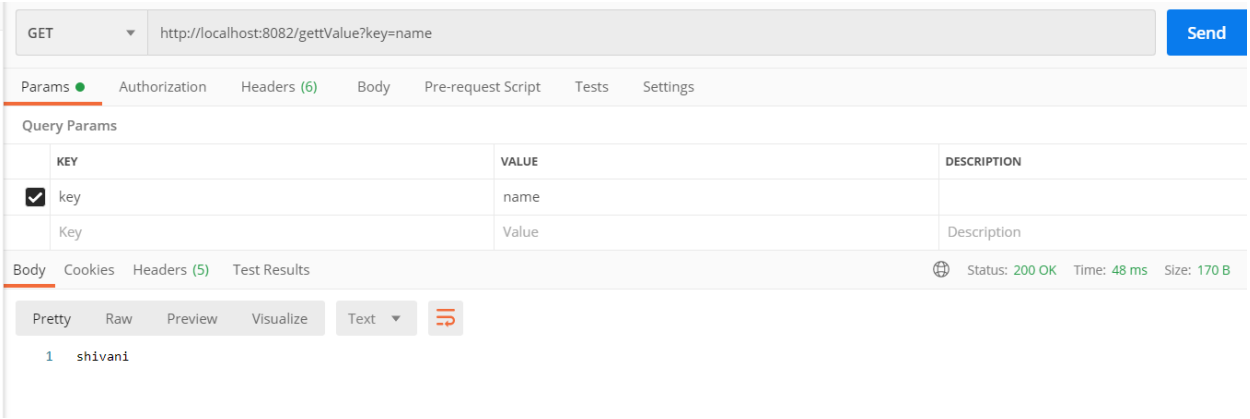
RESULT when we run for the first instance using port 8082:

```
Members {size:2, ver:2} [
        Member [192.168.245.1]:5701 - a778e7e2-a523-43fa-86ef-81ee1451573e
        Member [192.168.245.1]:5702 - 85c1cf55-0c0e-4bad-8e1d-a7d3e9e89379 this
]
```

Putting data using 8081:

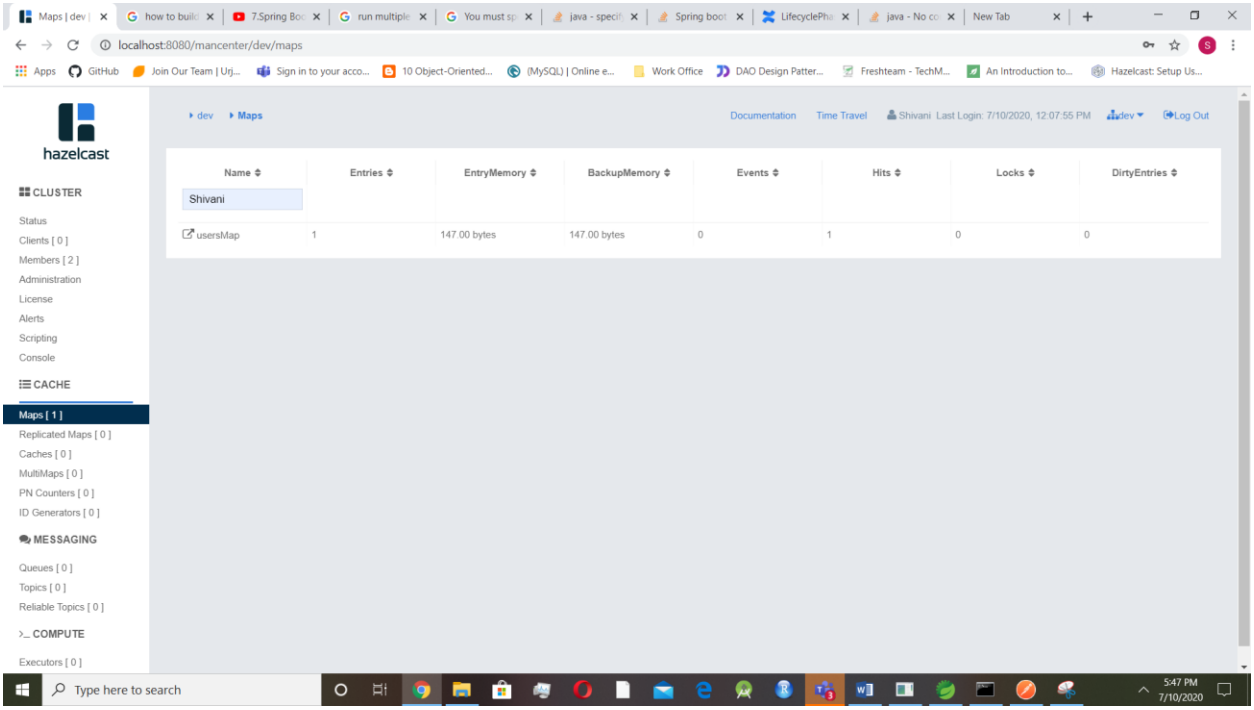


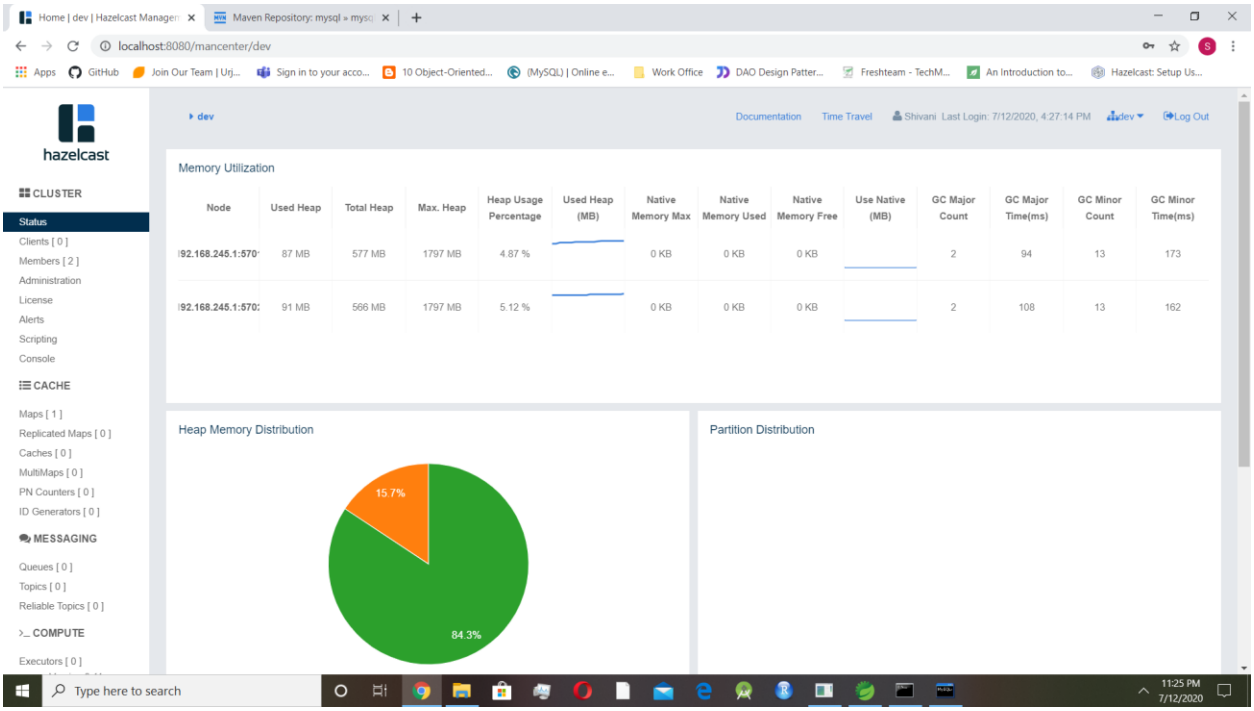
Reading data using another port 8082:



MANAGEMENT CENTER:

After running two instances, it looks like below.





When we stop one of the instance, we can retrieve the data from another instance.

Example:

I have killed the process ID of 8081.

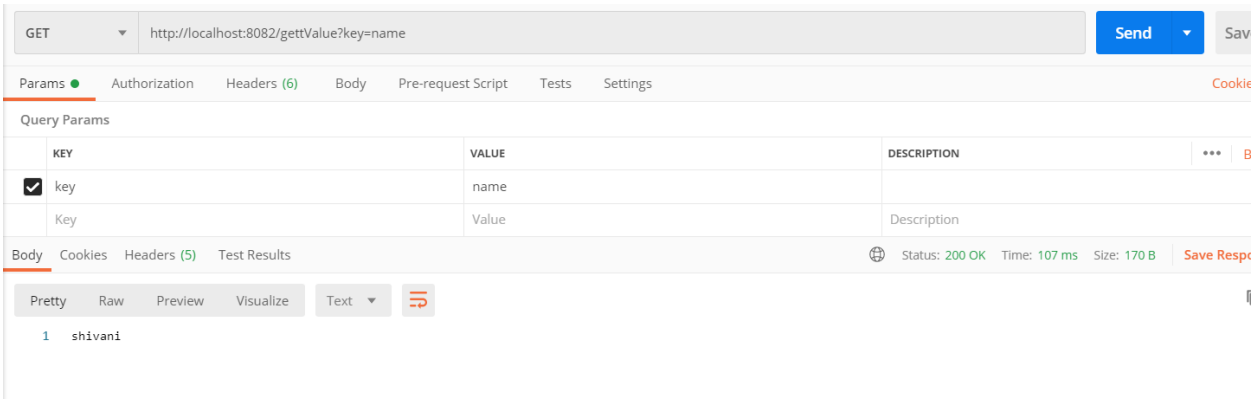
```
C:\Users\Admin>netstat -a | findstr "8081"
  TCP    0.0.0.0:8081           0:0                    LISTENING
^C^C
C:\Users\Admin>netstat -aon | findstr "8081"
  TCP    0.0.0.0:8081           0.0.0.0:0              LISTENING       5400
  TCP    [::]:8081              [::]:0                 LISTENING       5400

C:\Users\Admin>taskkill /f /PID 5400
SUCCESS: The process with PID 5400 has been terminated.

C:\Users\Admin>netstat -aon

Active Connections
```

We can retrieve the data using port 8082 still…



Management center looks like below: