Hibernate 2nd level cache with HazelCast, Spring Boot and Spring Data JPA

Pavan Jadda
Feb 14 · 3 min read

This blog explains the process to implement Hibernate 2nd level cache with HazelCast, Spring Boot and Spring Data JPA. Source code and config files are uploaded to **Github**

pavankjadda/Hazelcast-Hibernate2ndLevelCache-SpringBoot

Hibernate 2nd level cache with HazelCast, Spring Boot and Spring Data JPA. Please checkout my...

github.com

Technologies

1. Spring Boot 2.2.x
2. Spring Data JPA 2.2.x
3. HazelCast 4.0
4. Docker

Start HazelCast instance

HazelCast instances are setup as Docker containers using the docker-compose file shown below.

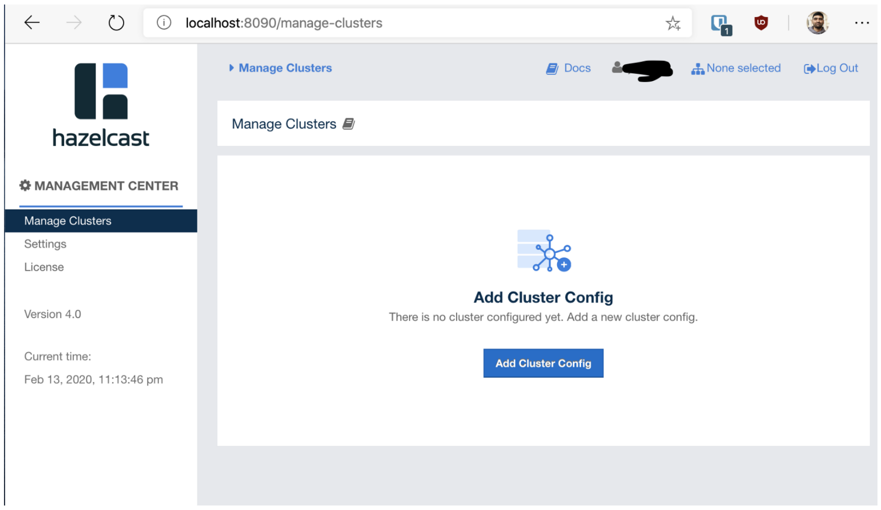
```yaml
version: '3'
services:
  hazelcast1:
    image: hazelcast/hazelcast
    environment:
      - 'JAVA_OPTS=-Dhazelcast.local.publicAddress=192
    ports:
```

docker-hazelcast.yml

Start the HazelCast instances and management center using the following command.

```
$ docker-compose -f <location of the docker-hazelcast.yml >
up
```

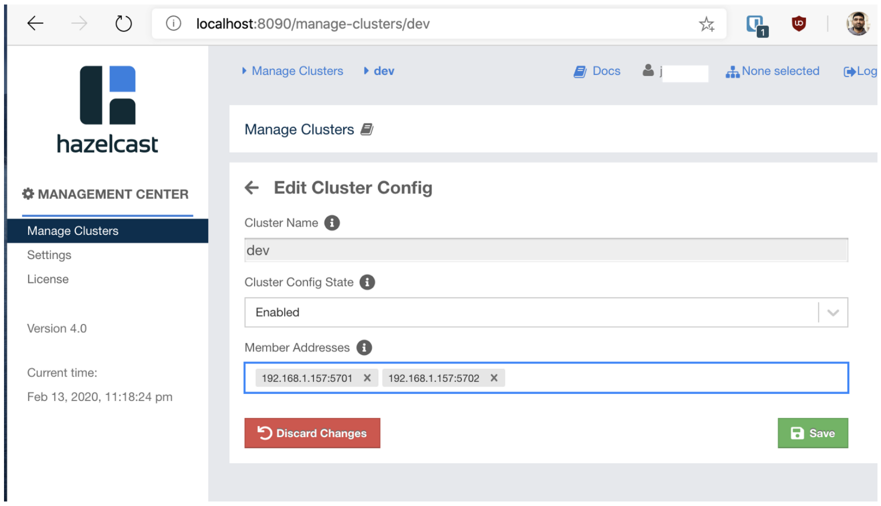
HazelCast management center should up and running at **http://localhost:8090/login.html**. Create a new account to see the management dashboard.



There is no cluster configured yet. Add a new cluster config.

Hazelcast default Dashboard

Click on **Add Cluster Config** to add newly created cluster config. Enter the cluster name as **dev**(default name) and replace **192.168.1.157** with your host IP Address as shown below.



Edit Cluster Config

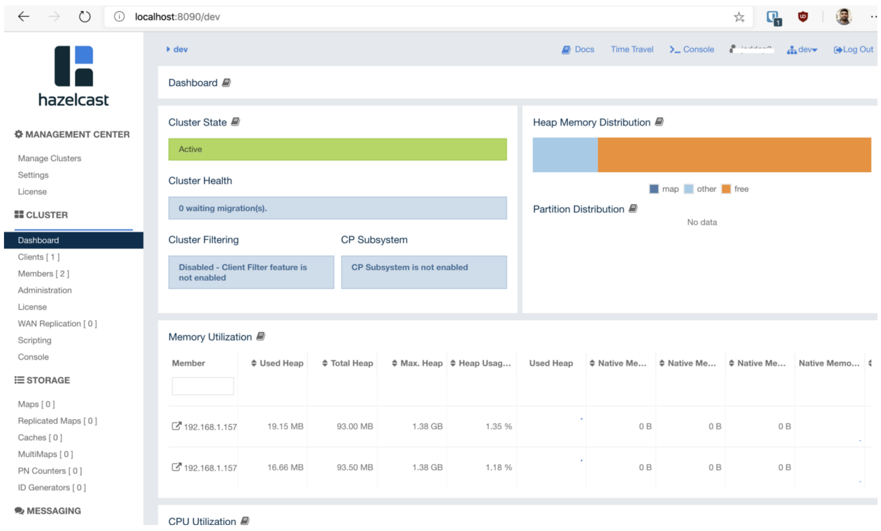
Cluster Name: dev

Cluster Config State: Enabled

Member Addresses: 192.168.1.157:5701, 192.168.1.157:5702

Add dev cluster config

Save the config and go to home page to see list of added clusters. Select **dev** cluster and you should see the following dashboard



Cluster State: Active

Cluster Health: 0 waiting migration(s).

Cluster Filtering: Disabled - Client Filter feature is not enabled

CP Subsystem: CP Subsystem is not enabled

Heap Memory Distribution

Memory Utilization

Member	Used Heap	Total Heap	Max. Heap	Heap Usag...	Used Heap	Native Me...	Native Me...	Native Me...	Native Memo...
192.168.1.157	19.15 MB	93.00 MB	1.38 GB	1.35 %		0 B	0 B	0 B	
192.168.1.157	16.66 MB	93.50 MB	1.38 GB	1.18 %		0 B	0 B	0 B	

Dev Dashboard

Spring Boot Application

1. **HazelCastConfig** class defines **Config** bean needed to connect Spring Boot app to HazelCast cluster as shown below

```
package com.pj.hazelcastdemo.config;

import com.hazelcast.config.Config;
import com.hazelcast.config.NetworkConfig;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configur

```

HazelCastConfig.java

2. The demo Spring Boot application has Employee domain object with EmployeeController and EmployeeRepository defined using Spring Data JPA. It uses H2 in memory data base to persist data

```
@Data
@Entity
@Cache(region = “employeeCache”, usage =
CacheConcurrencyStrategy.READ_WRITE)
public class Employee
{
   @Id
   private String empId;
   private String empName;
}
```

3. The annotation **@Cache** creates **employeeCache** map in HazelCast cache and takes care of **CachePut** and **CacheEvict** operations. By default **findAll(), create, update, delete** method calls hit database and **findById()** method hits HazelCast cache.

4. Run the application and go to http://localhost:8080/api/v1/employee/find/all to see all the employees.

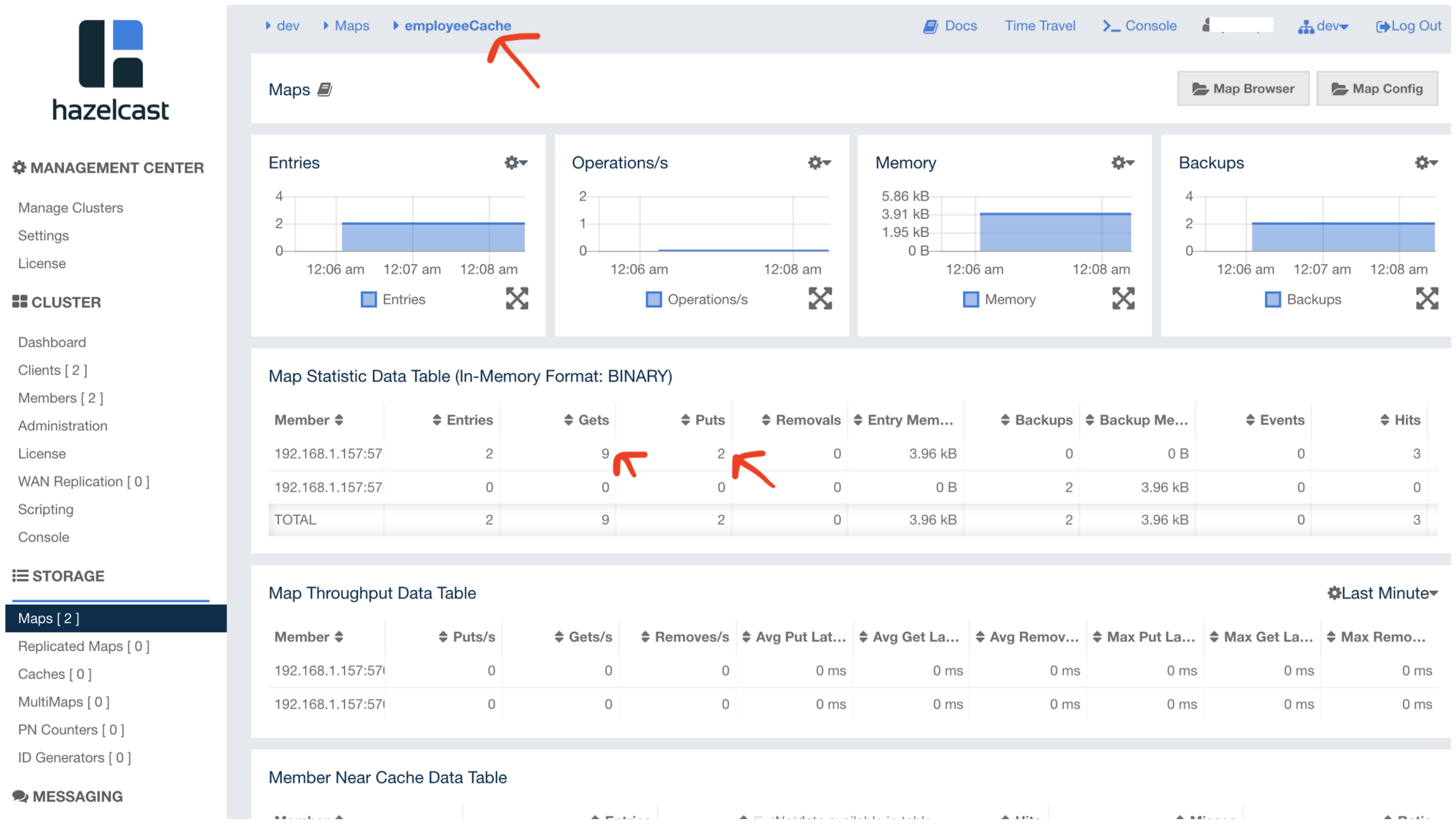
5. Create new employee by hitting the link http://localhost:8080/api/v1/employee/create

```
Hibernate Logs:


Hibernate: select employee0_.emp_id as emp_id1_0_0_,
employee0_.emp_name as emp_name2_0_0_ from employee
employee0_ where employee0_.emp_id=?


Hibernate: insert into employee (emp_name, emp_id) values
(?, ?)

Hibernate: select employee0_.emp_id as emp_id1_0_0_,
employee0_.emp_name as emp_name2_0_0_ from employee
employee0_ where employee0_.emp_id=?
```

6. If you look at hibernate logs, you should see **2 select statements** for 2 findAll() method calls and **1 insert statement** for findById() method call

5. Now try to get employee information by providing the ID created in step 2. **http://localhost:8080/api/v1/employee/find/<your_emp_id>**. If you look at the hibernate logs, you should see nothing. That means Spring Boot returned the employee record from HazelCast cache without hitting the database.

7. Look at the following employeeCache map from HazelCast management center, you should be see all the **Gets** and **Puts** to cache



Map Statistic Data Table (In-Memory Format: BINARY)

Member	Entries	Gets	Puts	Removals	Entry Mem...	Backups	Backup Me...	Events	Hits
192.168.1.157:57	2	9	2	0	3.96 kB	0	0 B	0	3
192.168.1.157:57	0	0	0	0	0 B	2	3.96 kB	0	0
TOTAL	2	9	2	0	3.96 kB	2	3.96 kB	0	3

Map Throughput Data Table

Member	Puts/s	Gets/s	Removes/s	Avg Put Lat...	Avg Get La...	Avg Remov...	Max Put La...	Max Get La...	Max Remo...
192.168.1.157:57	0	0	0	0 ms	0 ms	0 ms	0 ms	0 ms	0 ms
192.168.1.157:57	0	0	0	0 ms	0 ms	0 ms	0 ms	0 ms	0 ms

Member Near Cache Data Table

. . .

Source code is available in Github for reference and Happy Coding