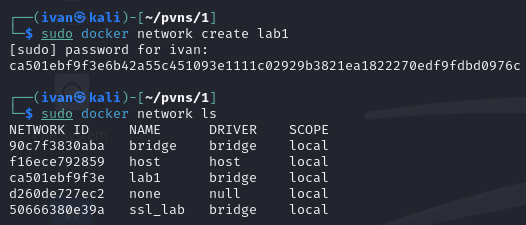
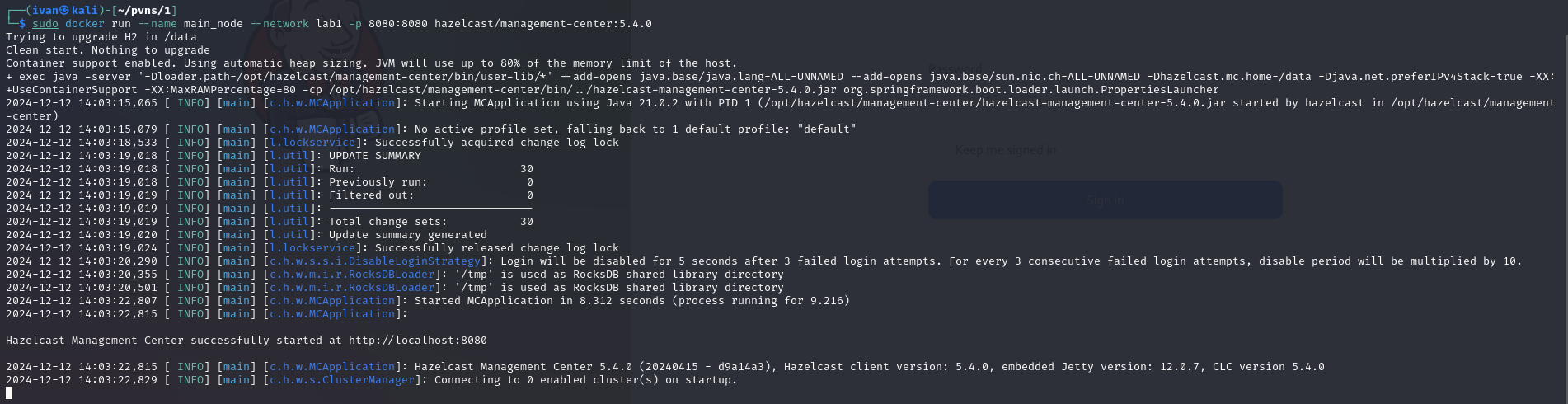
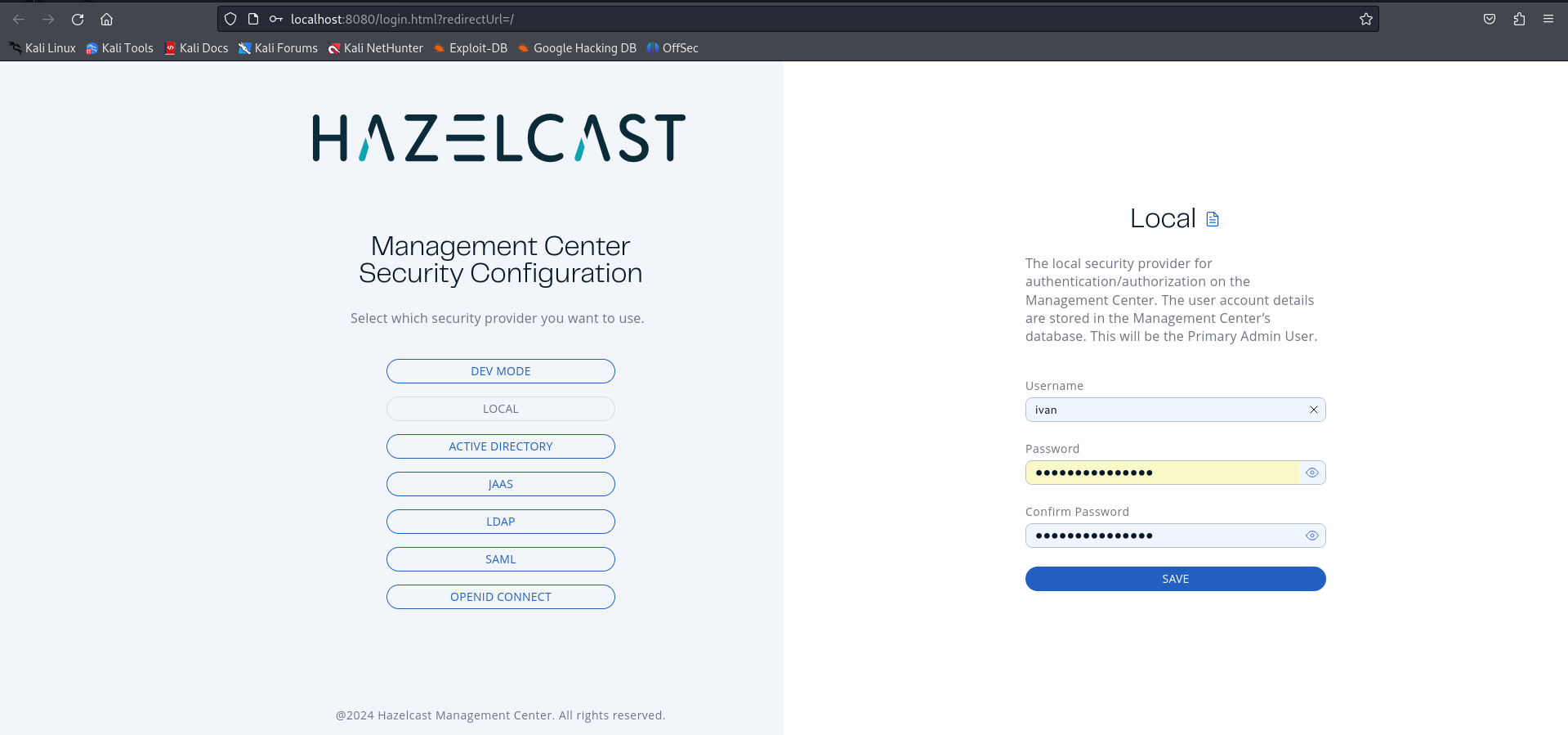
**Завдання:**

1. Встановити і налаштувати **Hazelcast 5.4.0** (у новіших версіях частина необхідного для виконання завдань функціоналу є платною)

<https://docs.hazelcast.com/hazelcast/5.4/getting-started/install-hazelcast>  <https://hazelcast.com/community-edition-projects/downloads/archives/#hazelcast-platform-5-4-0>

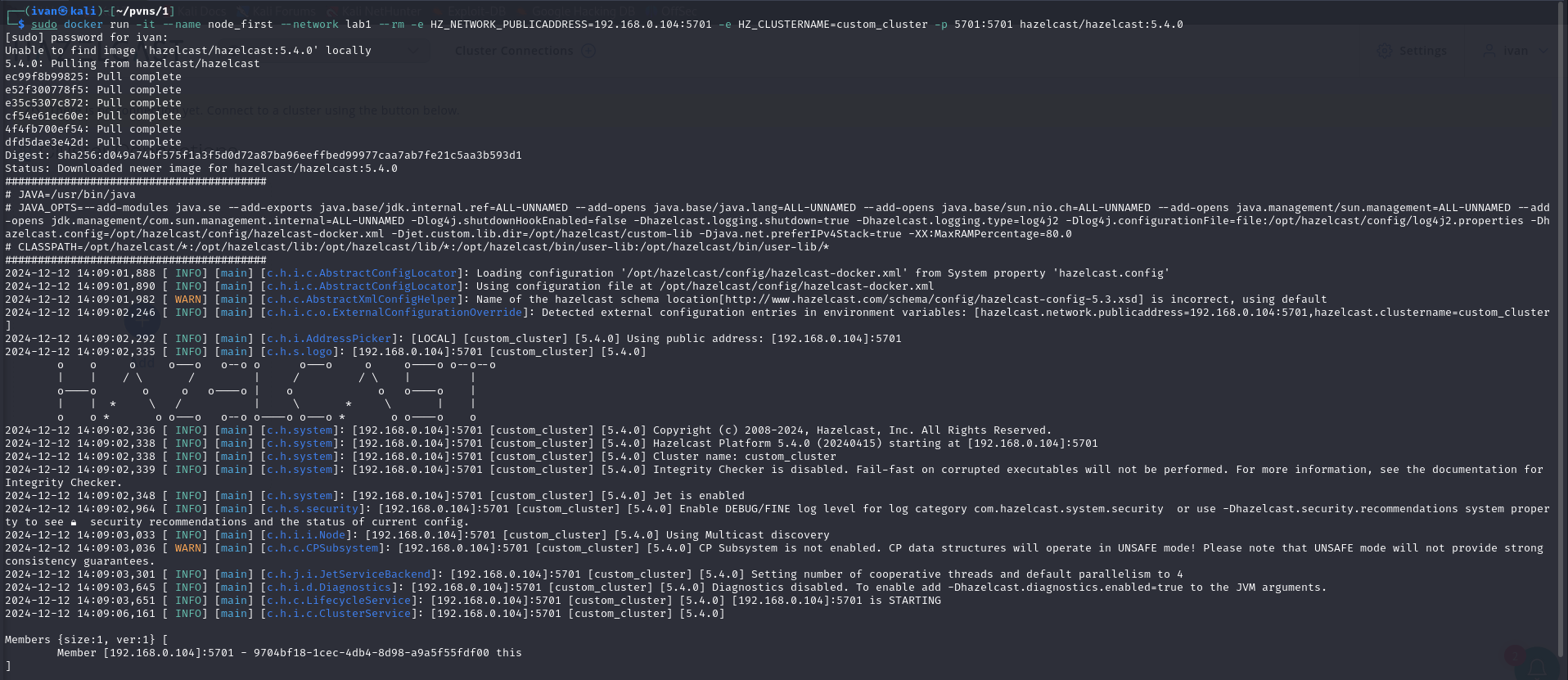


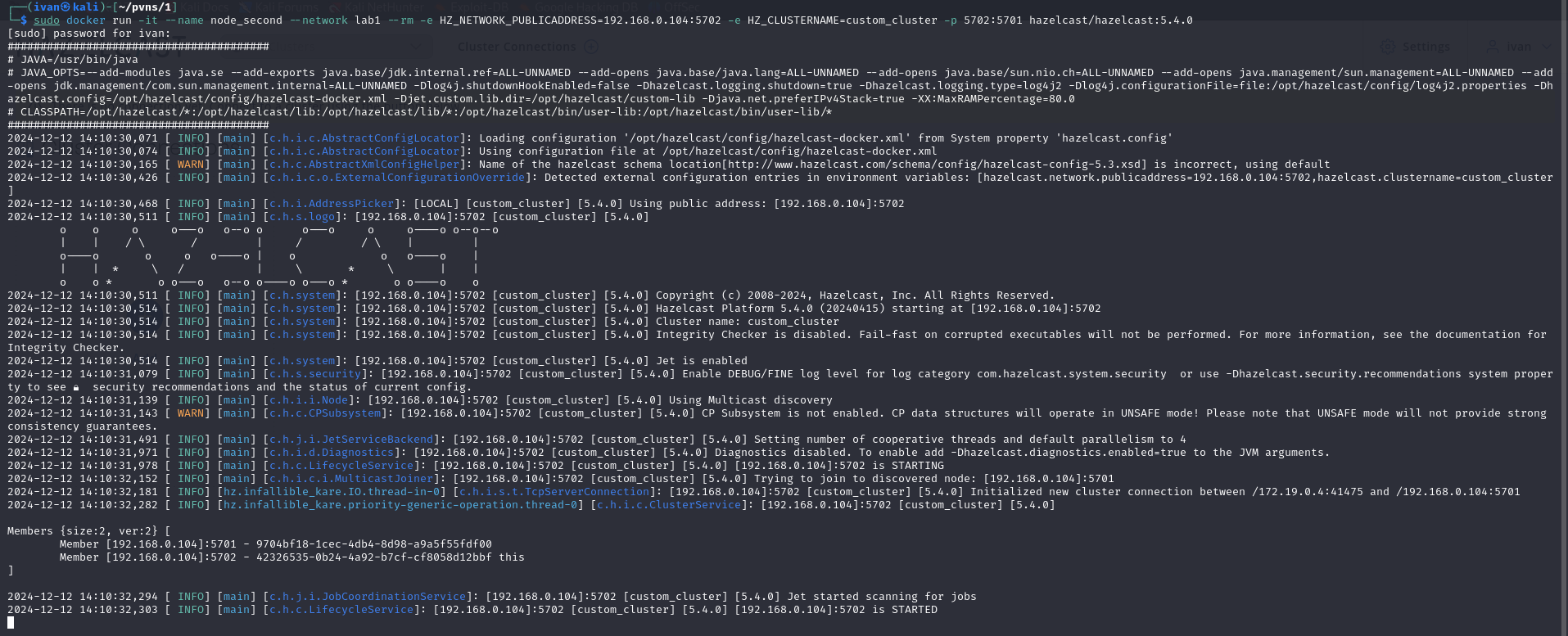


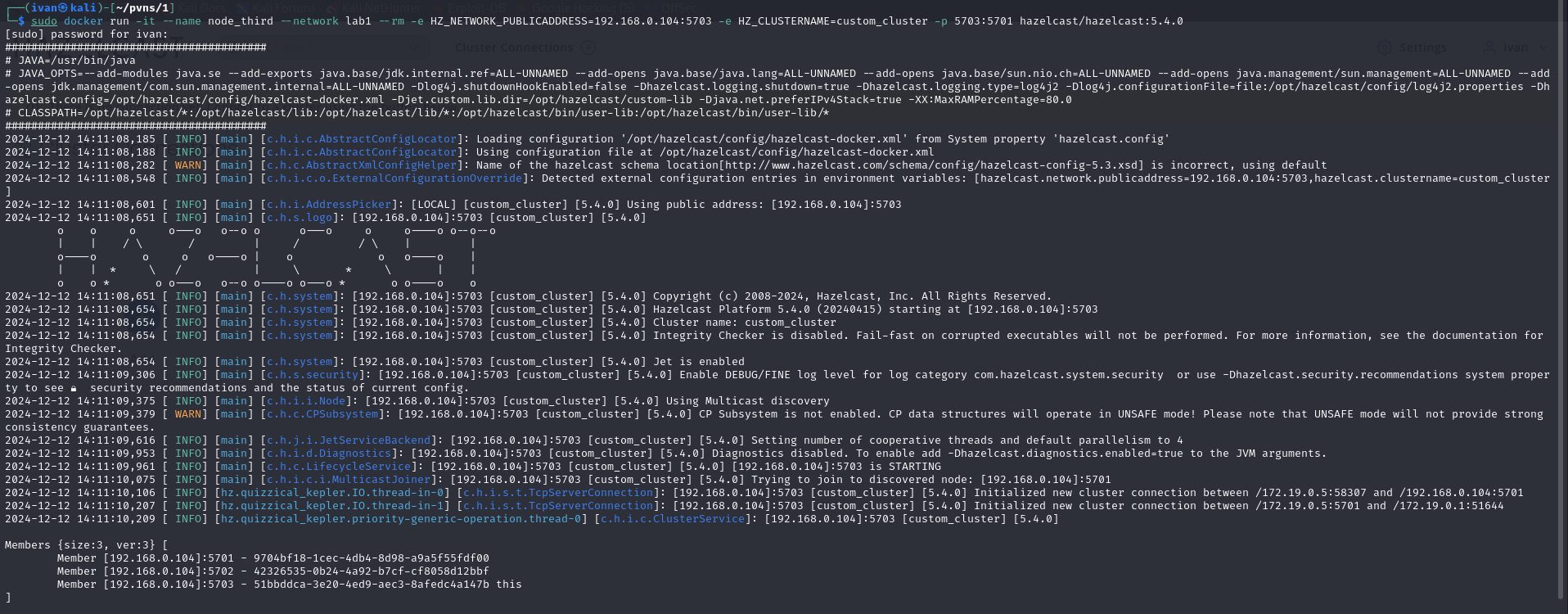


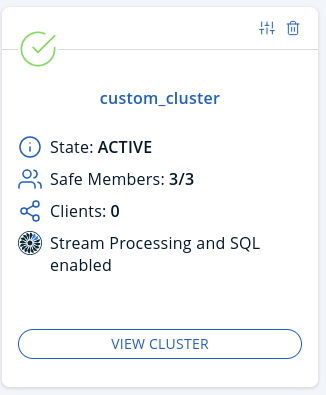
1. Сконфігурувати і запустити 3 ноди (інстанси) об'єднані в кластер або як частину Java-застосування, або як окремі застосування

У справжній системі кожна нода має запускатись на окремому сервері.









1. Далі, на основі прикладу з Distributed Map, напишіть код який буде емулювати інкремент значення для одного й того самого ключа у циклі до 10К. Це необхідно робити у 10 потоках.

*for (t in 1..10) {*

*thread {*

*for (i in 1..10\_000) {*

*increment\_counter()*

*// distribMap.put(key, counter + 1)*

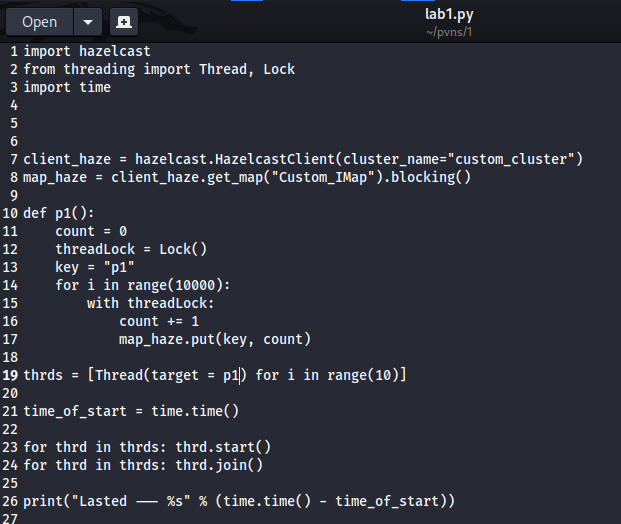
*}*

*}*

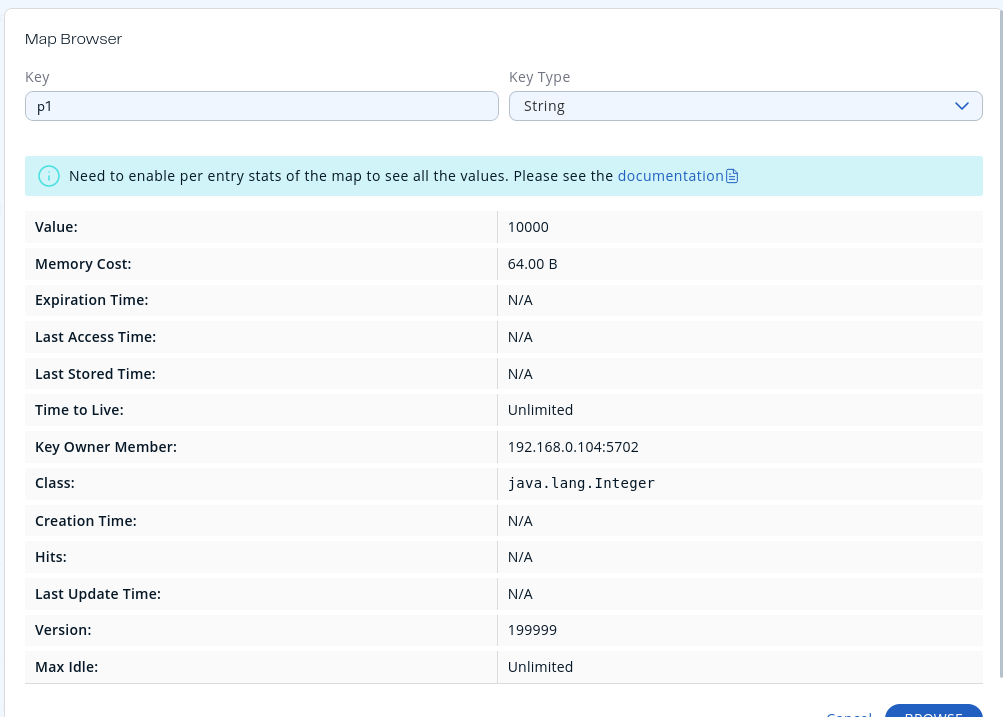
*}*

Виходячи з того, шо 10 потоків інкрементують каунтер 10К разів кожен, то остаточне значення каунтера має бути 10\*10\_000 = 100\_000

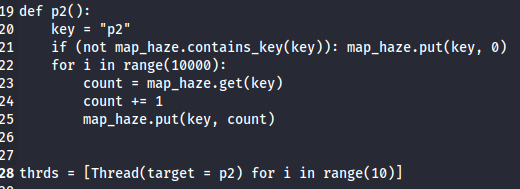
Для імплементації спочатку скористаємось Distributed Map у Hazelcast



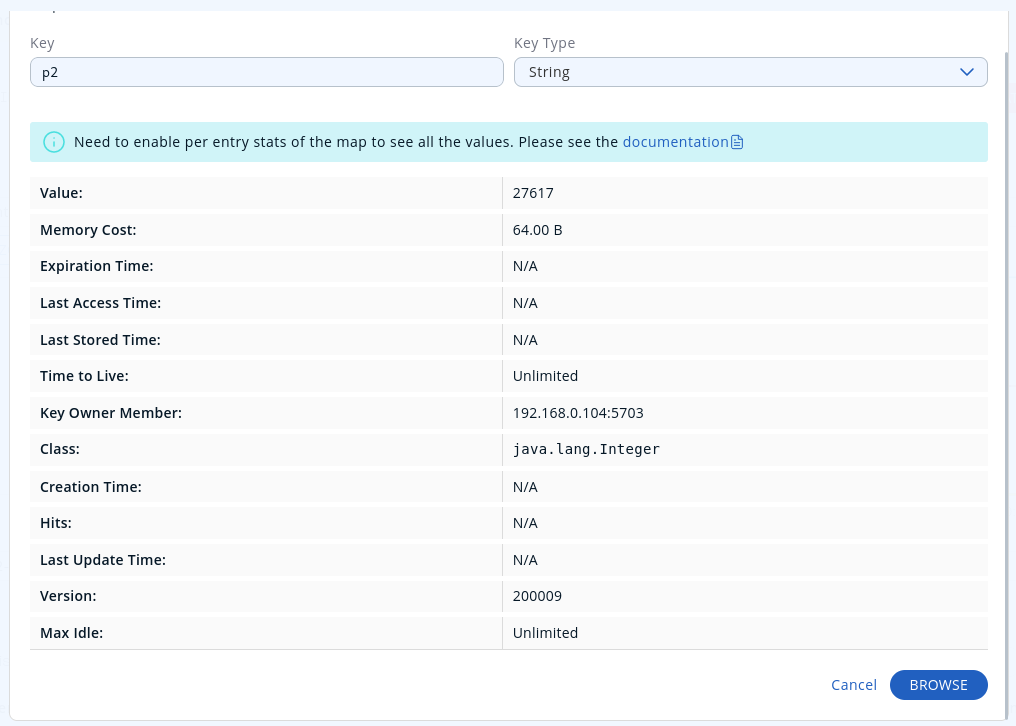




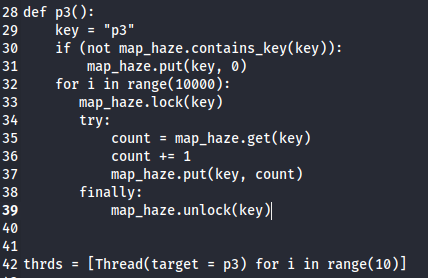
1. На основі прикладу <https://docs.hazelcast.com/imdg/latest/data-structures/map#locking-maps> реалізуйте каунтер без блокувань. Поміряйте час виконання, та подивиться чи коректне кінцеве значення каунтера ви отримаєте.



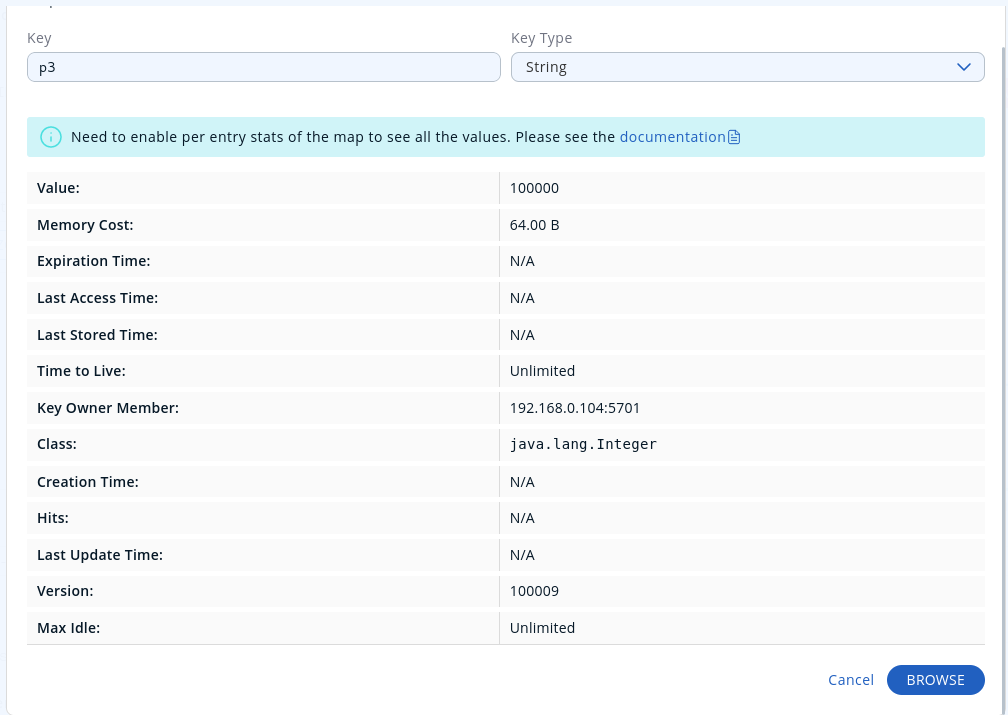




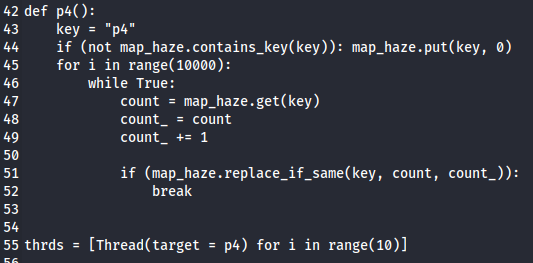
1. На основі прикладу <https://docs.hazelcast.com/imdg/latest/data-structures/map#pessimistic-locking>  реалізуйте каунтер з використанням песимістичного блокування. Поміряйте час виконання, та подивиться чи коректне кінцеве значення каунтера ви отримаєте.



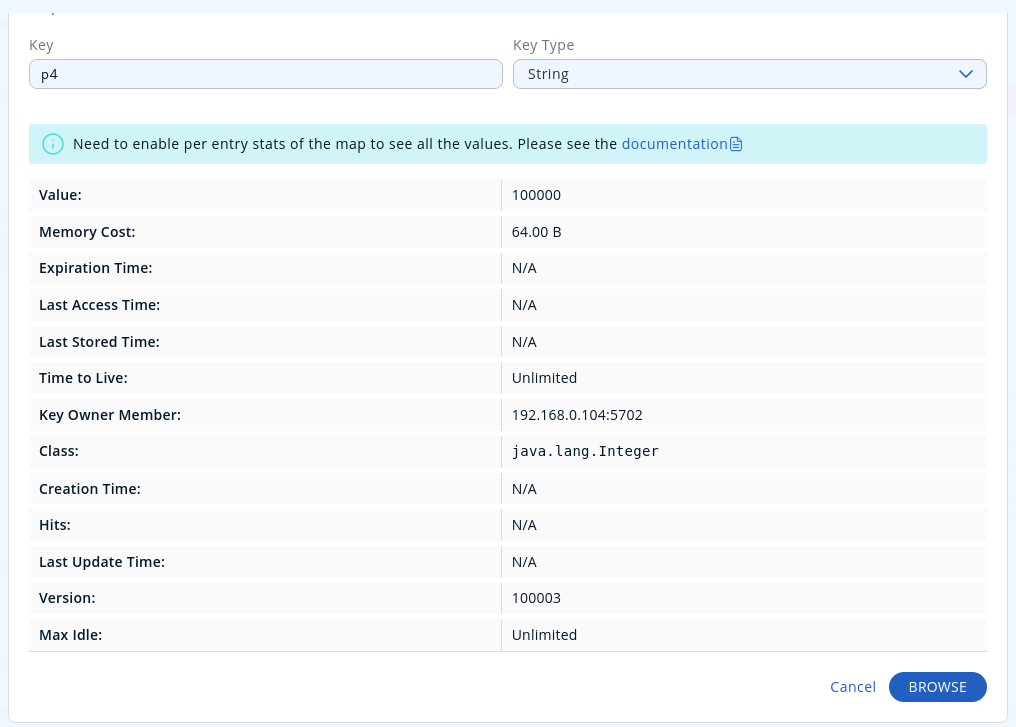




1. На основі прикладу <https://docs.hazelcast.com/imdg/latest/data-structures/map#optimistic-locking>  реалізуйте каунтер з використанням оптимістичного блокування. Поміряйте час виконання, та подивиться чи коректне кінцеве значення каунтера ви отримаєте.







1. На справді, в загальному випадку, Distributed Map у Hazelcast не гарантує, що у випадку падіння ноди на інші ноди встигне реплікуватись (скопіюватись) значення ключа у Distributed Map.  
   Тому використаємо іншу структуру даних, яка дає більші гарантії - *IAtomicLong* та використовує так званий протокол консенсусу Raft: <https://docs.hazelcast.com/hazelcast/5.4/data-structures/iatomiclong>

Реалізуйте каунтер з використанням *IAtomicLong* та увімкнувши підтимку *CP Sysbsystem* (<https://docs.hazelcast.com/hazelcast/5.4/cp-subsystem/configuration>) на основі трьох нод.   
**УВАГА! Без *CP Sysbsystem* не гарантується коректність результату**

Поміряйте час виконання, та подивиться чи коректне кінцеве значення каунтера ви отримаєте.

Попередньо змінив обмеження для cp-subsystem для членів і поставив обмеження на кількість членів та групи рівним 3.

