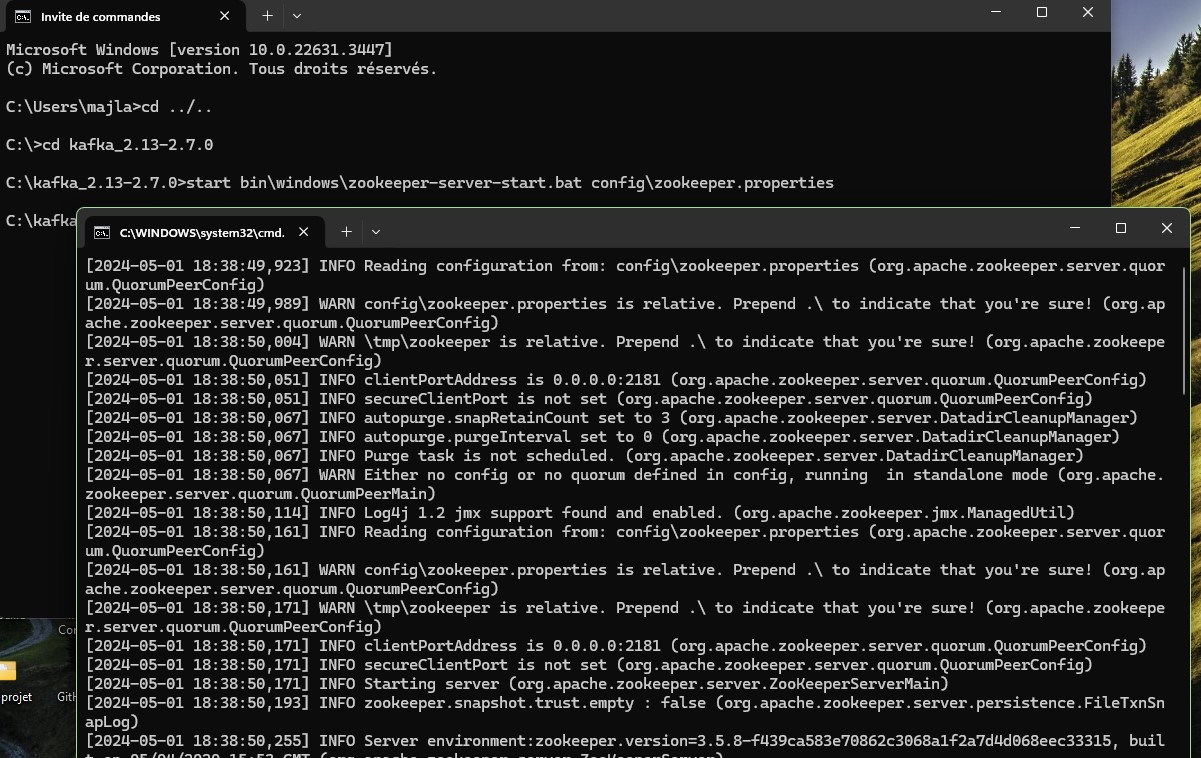
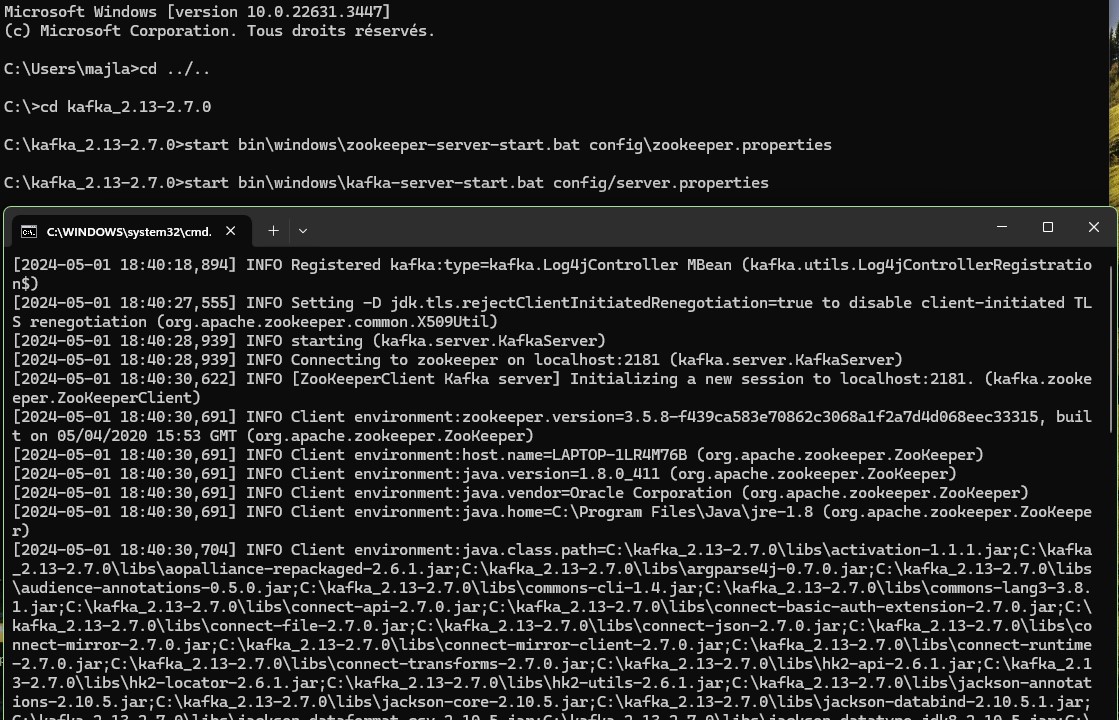
***Part 1 - Even Driven Distributed Processing With Spring Cloud Stream Functions - KAFKA Broker***

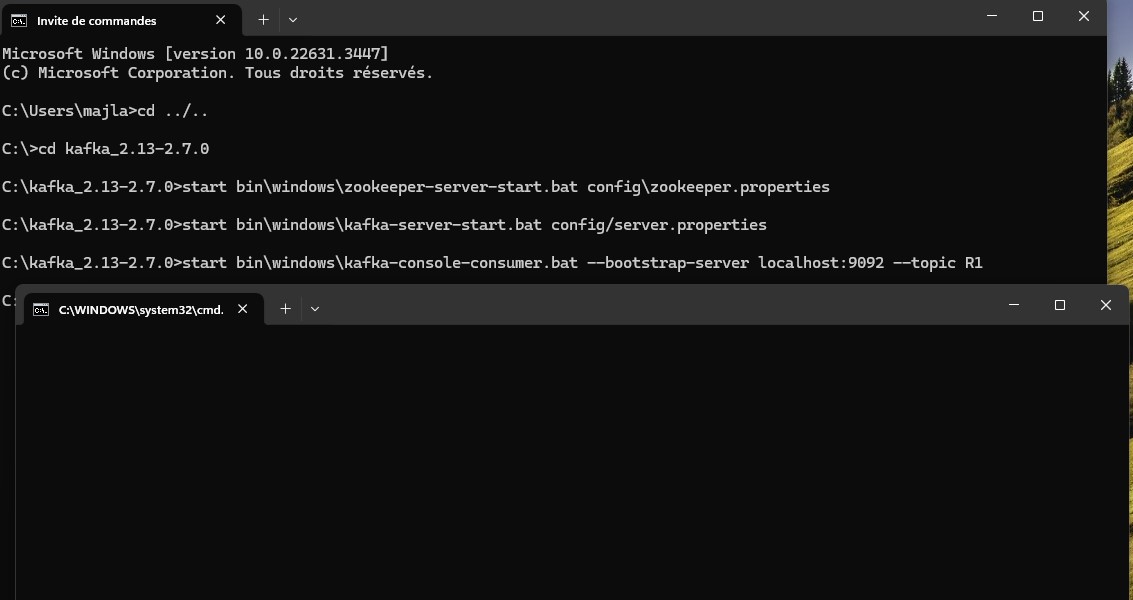
Tout dabord on commence par le démarrage de serveur zookeeper en ligne de commande (cmd) :



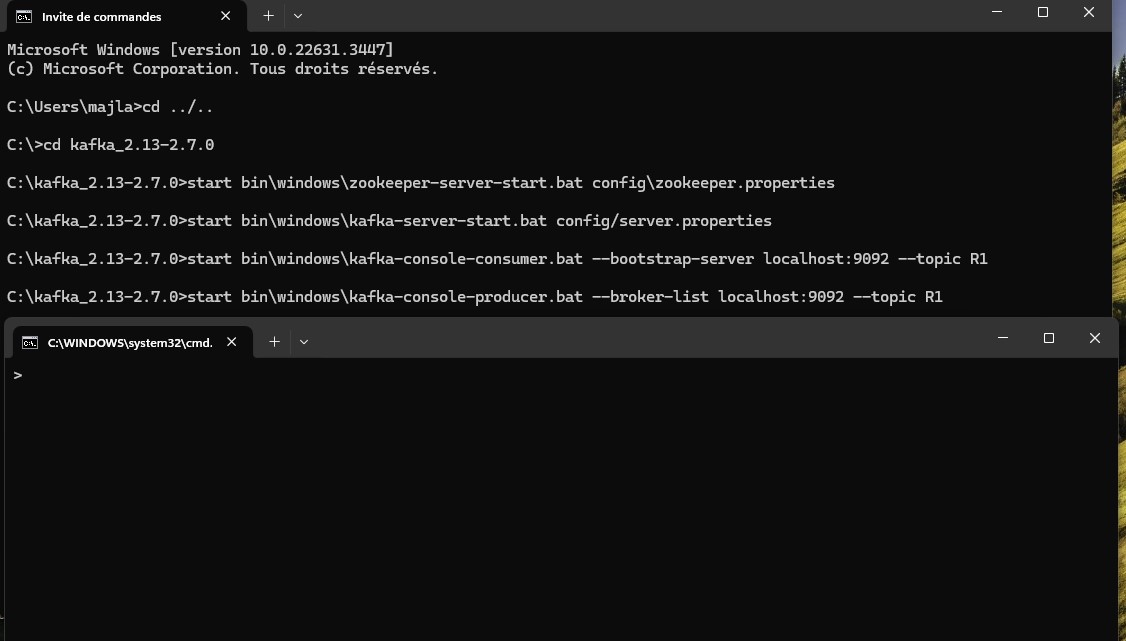
Une fois zookeeper est démarré on va démarrer le serveur KAFKA en ligne de commande (cmd) :



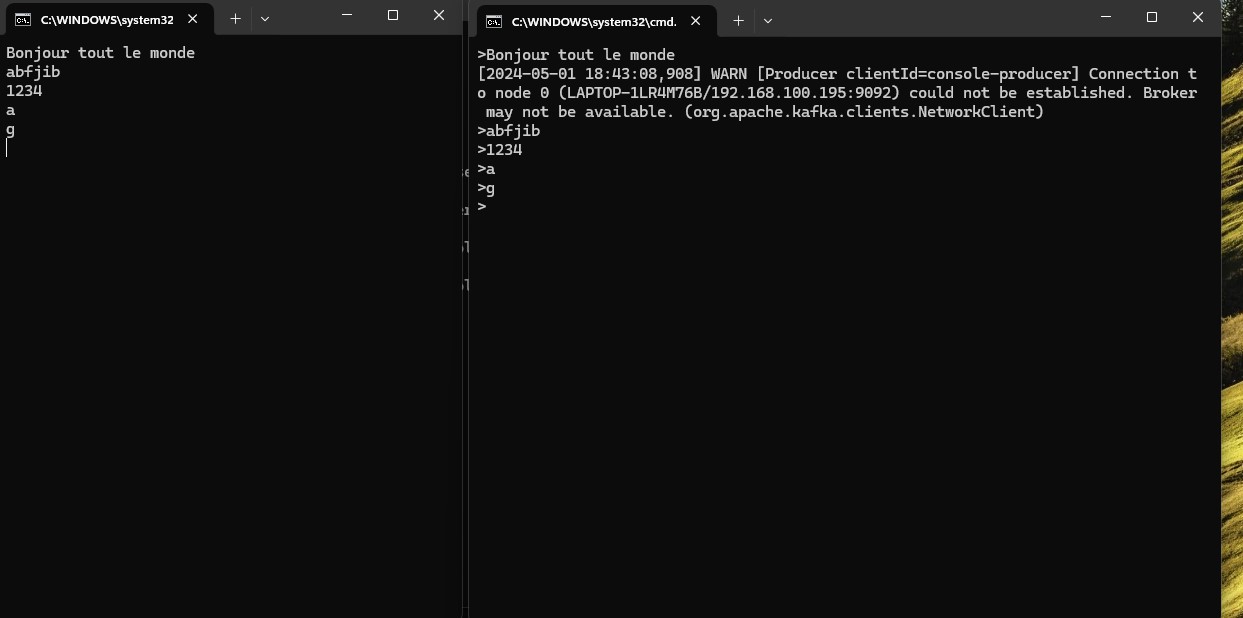
Puis on démarre l'outil kafka-console-consumer en ligne de commande (cmd) en spécifiant l'emplacement où il se trouve kafka avec l'option --bootstrap-server localhost:9092 et le nom de topic avec l'option --topic R1 :

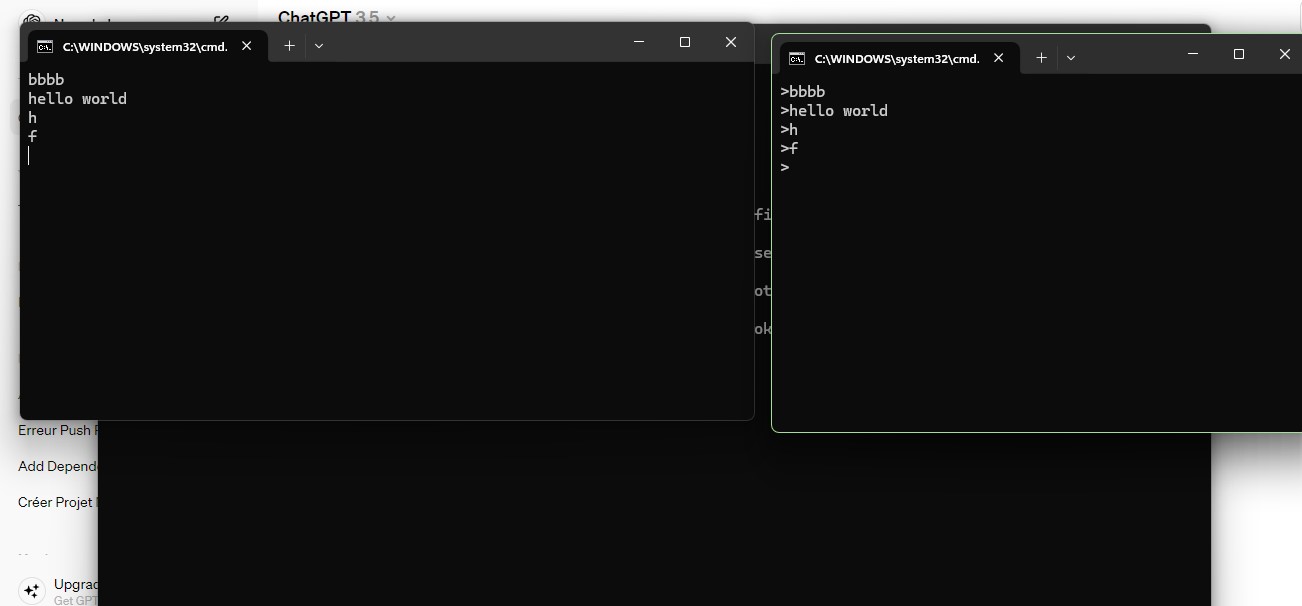


Pour envoyer des messages vers ce topic on va lancer kafka-console-producer :



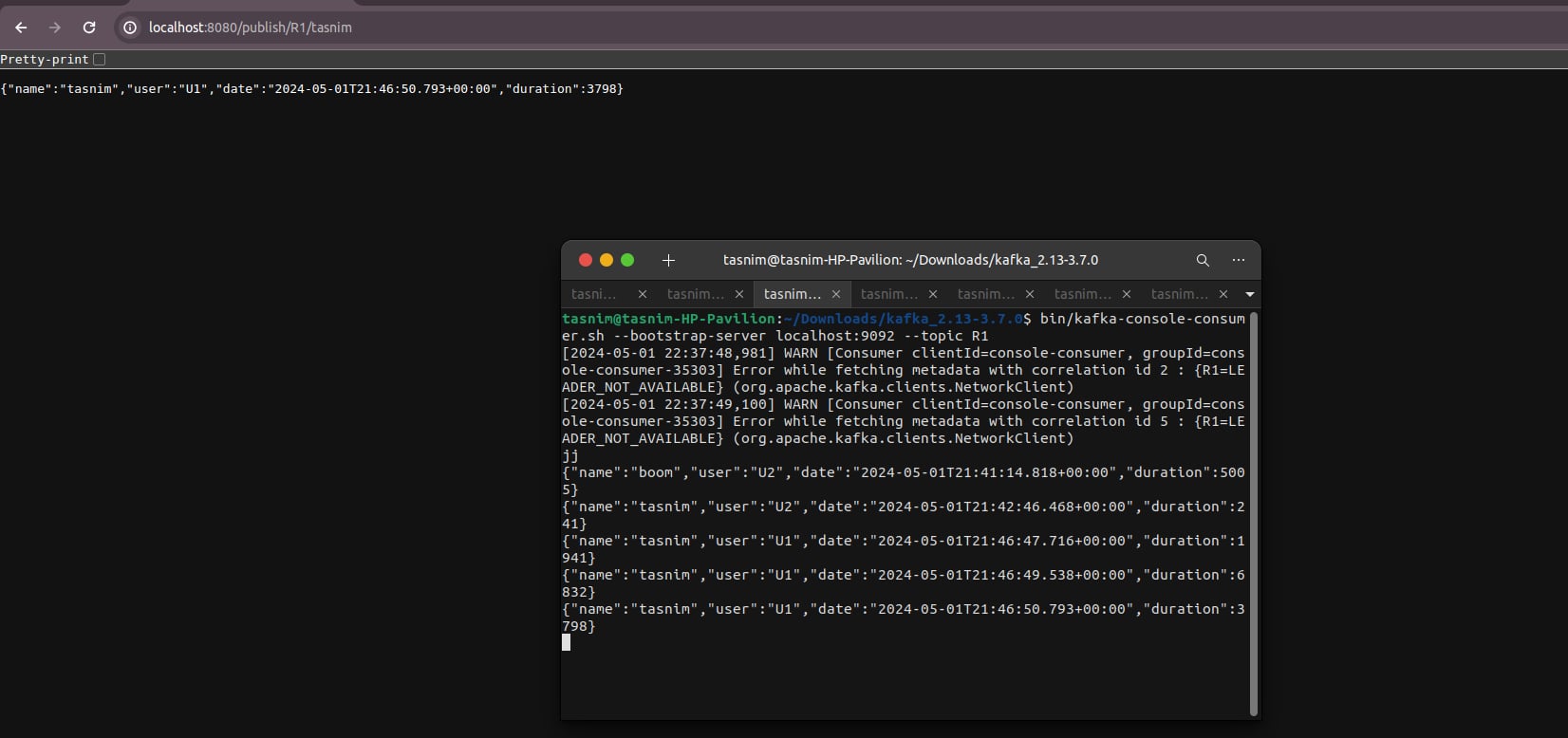
Maintenant on envoie des messages par le producer et on teste s'ils vont arrivés au consommateur via le broker kafka :



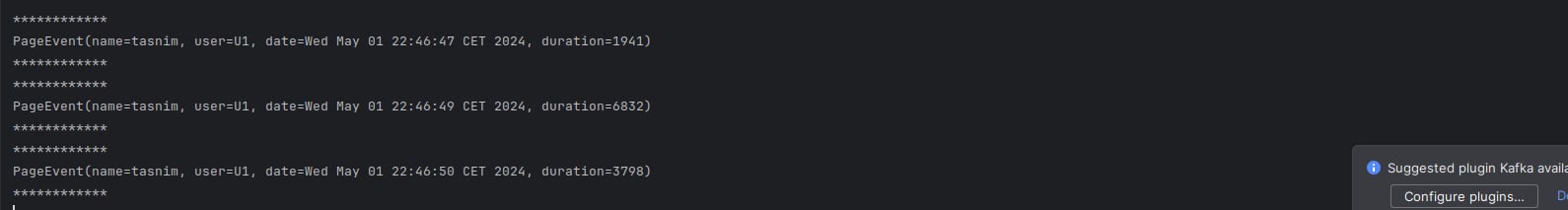


Après avoir tester le fonctionnement de broker kafka avec kafka-console-consumer et kafka-console-producer on passe à la création d'une application Spring Boot dans la quelle on va essayer de produire des messages dans un topic.

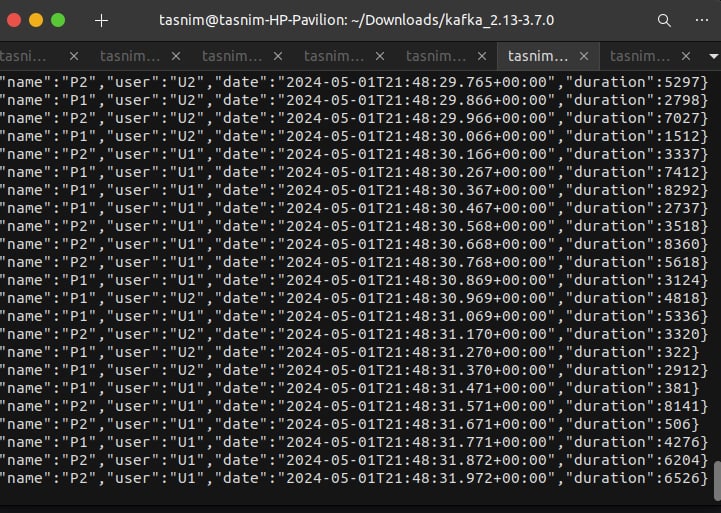
Dans le premièr cas on publie des messages vers le topic kafka à travers d'un RestController pour q'un consommateur kafka-console-consumer puisse les consommer :



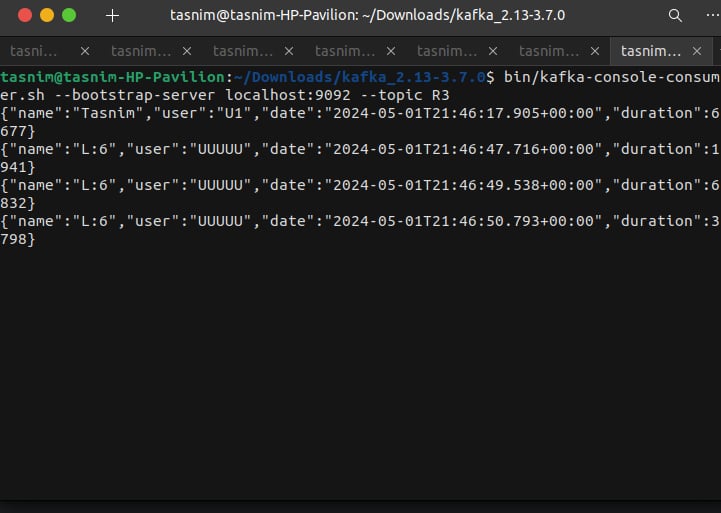
Dans le deuxième cas au lieu d'utiliser kafka-console-consumer on va créer un consumer qui va écouter le topic et de lire les messages qui arrivent sur le topic :

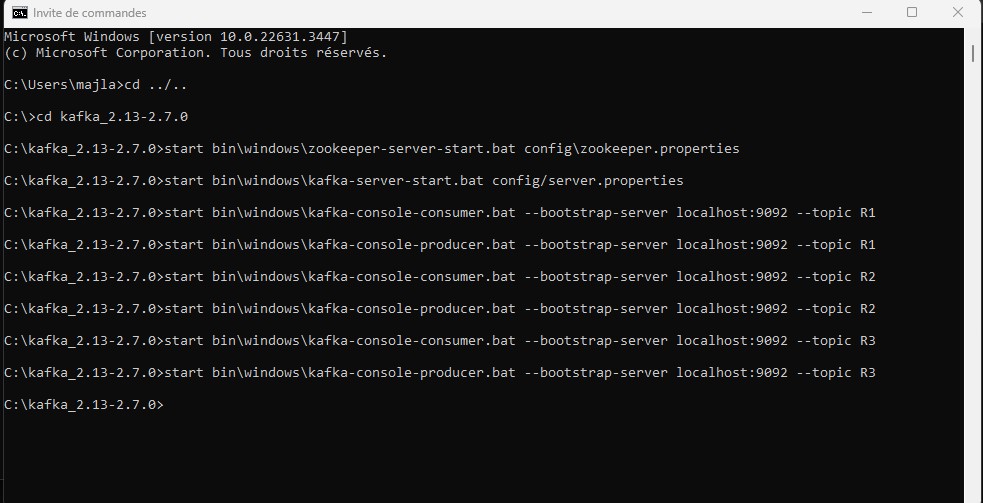


Pour le troisième cas au lieu d'utiliser kafka-console-producer on va créer un producer qui va produire des streams vers le topic chaque seconde :



Dans le quatrième cas on va créer à la fois un producer et un consumer :





***Exécution sur intelij***

