1. Viết chương trình nén dữ liệu lấy từ api bằng phương pháp kiểm tra biên độ giao động. Nếu data thay đổi lớn (vượt ngưỡng) thì sẽ ghi lại ngay giá trị đó, nhưng nếu data không có sự thay đổi lớn nào thì định kỳ sau một thời gian nhất định vẫn phải ghi data đó xuống.
   - Lưu ý: mỗi lần lấy data mới thì so sánh với dữ liệu cuối cùng được ghi nhận gần nhất (vượt ngưỡng hoặc sau thời gian), không phải dữ liệu query của lần trước đó.
   - Tạo maven project, sau đó add thêm 2 thư viện logback, và json để parse một dữ liệu json từ URL http://news.admicro.vn:10002/api/realtime?domain=kenh14.vn. Lưu ý, dữ liệu query từ URL là dữ liệu thay đổi liên tục. Do đó dữ liệu nhận được sẽ là dạng time-series với độ phân giải là 2s. Thực hiện phương pháp nén dữ liệu time-series trên với ngưỡng biến đổi 0.5% theo phương pháp sau:
      a. Lấy dữ liệu mỗi 2s một lần
      b. Nếu số user trong field user ở json trả ra mà lớn hơn 0.5% so với **số user lần cuối cùng ghi vào file log** thì sử dụng logback ghi số user vào file log với level là INFO, nếu không thì sau 12s ghi lại số user đó vào file log với level DEBUG.
      c. Setup file log rolling sau mỗi 1 phút ghi ra một file mới. Một file dung lượng tối đa là 1MB.

0s	2s	4s	6s	8s	10s	12s	14s	16s	18s	20s	22s
1032	1030	1092	1091	1092	1090	1089	1090	1132	1032	1232	1232
Info	Ko ghi	Info	Ko ghi	Ko ghi	Ko ghi	Ko ghi	Ko ghi	Debug	Ko ghi	Info	Ko ghi

2. Viết chương trình đơn giản để crawl data từ một url. Tạo maven project với project encode là utf-8. Sử dụng thư viện jsoup lấy nội dung html của url "http://dantri.com.vn" và nội dung sau khi đã loại bỏ các thẻ, ghi ra một file với tên file là thời gian hiện tại theo format yyyy_MM_dd_mm:ss.txt

Tiếp theo build toàn bộ project thành một file jar duy nhất chứa tất cả các thư viện đi kèm (sử dụng plug-in *jar-with-dependencies* của maven).

Tạo một máy ảo linux sử dụng Docker hoặc subsystem (lưu ý không cài các bài linux có giao diện). Tiến hành cài openssh trên máy ảo, Sử dụng một tool SSP (như xmanager, PuTTY…) trên máy thật và remote ssh vào máy ảo sử dụng ssh-key mà **không sử dụng phương pháp remote bằng password**.

Copy file jar vừa build vào trong máy ảo sử dụng câu lệnh SCP. Sau đó dùng shell script của linux (file .sh) để chạy file jar mỗi 5 giây một lần.

3. Viết một webservice đơn giản với việc trả lại danh sách số nguyên tố từ 0 đến số truyền vào.
Tạo một project viết một restful webservice (có thể sử dụng sparkjava.com) trả lại dãy số nguyên tố từ một đến n nhập vào từ param trên url của request (Ví dụ: http://localhost:8080/prime?n=10000 sẽ trả lại danh sách các số nguyên tố từ 1 đến 10000).

Hãy chạy chương trình trong chế độ debug, giả sử nhập vào số n=12571, sử dụng đặt điều kiện vào breakpoint để dừng chương trình tại các số nguyên tố tìm ra mà giá trị của nó nằm trong khoảng 1.000 đến 3.000. **Lưu ý:** không thêm các câu lệnh rẽ nhánh vào code để thực hiện debug.

4. Hãy viết cache cho webservice ở bài 3 để giúp giảm số lần tính toán nếu người dùng thường xuyên truy vấn các số n giống nhau. Sử dụng cache bằng Hashmap để trả lại ngay kết quả cho số muốn n nhập vào nếu số n đã tồn tại trong cache. Ngoài ra, Cache này có thêm TTL (thời gian sống cho từng phần tử):

- sau khi ghi một phần tử mới vào Cache thì sau *m* giây thì sẽ tự xóa phần tử này đi
- nếu sau *n* giây không có request đọc phần tử nào đó trong Cache (sử dụng hàm get vào phần tử đó) thì Cache cũng tự xóa đi.

Viết cache này bằng cách tạo ra một Class mới là CacheTTL<K, V> implements Map<K, V>, trong đó đảm bảo những phương thức sau:

- CacheTTL(int n, int m) : hàm khởi tạo với 2 giá trị tham số *n, m*
- V get(K key) : hàm trả lại value tương ứng với key
- void put(K key, V value): hàm đẩy cặp giá trị tương ứng vào cache
- Map<K,V> getMap(): hàm trả lại tất cả các thành phần còn lại trong cache

5. Thay thế cache trên bài 4 bằng thư viện Guava cache, vẫn xóa sau 10s không có request và 20s sau khi ghi vào cache. Upload project lên github.com, đồng thời cùng lúc làm 2 việc sau để tạo confict code:

- Vào project trên github, mở một file trên trình duyệt, sửa nội dung file thêm dòng /* Chinh sua tren server github */, sau đó lưu lại
- Vào project trên máy, cũng vào file đó nhưng thêm dòng /* Chinh sua tren server may client */, sau đó Commit và Push lên git server

Hãy merge và edit conflict 2 đoạn code trên và sửa 2 dòng trên thành dòng comment sau: /* Chinh sua hop nhat giua server & client */ NOTE: PROJECT ĐẨY LÊN GIT KHÔNG NÊN ĐỂ PUBLIC.

6. Deploy project bài 5 thông qua git clone và build file jar bằng lệnh của maven trên máy ảo docker. Hãy set heap size tối đa khi chạy chương trình là 512MB, và heap size khi mới khởi tạo là 125MB. Build image mới để có thể tạo 2 instance khác giống hệt như vậy mà không cần phải vào config cho từng container.

7. Viết một ứng dụng client – server để đồng bộ dữ liệu giữa các client với nhau dựa vào http restful.

Mỗi client khi tạo ra sẽ cài đặt để monitor một folder chỉ định từ lúc start client. Nếu folder đó có bất kỳ khi thay đổi nào thì thay đổi đó sẽ được đồng bộ cho tất cả các client khác đang connect đến Server.

Server có trách nhiệm truyền tải thông tin giữa các client với nhau. Các client cần đăng ký với server để đồng bộ hóa dữ liệu. Khi client ngắt kết nối đến server trong một khoảng thời gian, khi nó connect lại thì phải tự đồng bộ các thay đổi mới nhất.

Lưu ý: Client ko được mở port hay chứa một webserver của riêng nó. Webserver chỉ được tạo trên server. Folder mà client monitor không chứa các thư mục con khác. Và mỗi file trong thư mục không quá 1MB.

Sau khi viết code xong, thì build ra 2 file jar: server.jar và client.jar. các file này sẽ được chạy lên trực tiếp từ dòng lệnh từ hệ điều hành với đối số truyền vào của server.jar là server_name và đối số truyền vào của client.jar là client_name (tên task client) và đường dẫn đến folder mà client đó cần phải đồng bộ.

Lưu ý việc build để khi khi chạy server và client lần lượt như sau:

- Server: java –jar server.jar server_name
- Client 1: java –jar client.jar client1_name path/folder1
- Client 2: java –jar client.jar client2_name path/folder2
- Client 3: java –jar client.jar client3_name path/folder3