Prerequisites:

1. To build the source code, must install golang and gin framework.

   The source was developed with go 1.12.5.

   go version go1.12.5 linux/amd64

   To install gin, run the below command:

   ```
   go get github.com/gin-gonic/gin
   ```

2. Related source code is in https://github.com/leimark/fibonacci.

   To clone and build the code:

   Run the build.sh script in <REPO_BASE>/fibonacci/scripts/build.sh.

   Notes: If git has been installed and SSL public key has been configured to access github, then can use "build.sh <SOURCE_DIR> git@github.com:leimark/fibonacci.git". Otherwise, use the HTTPS clone, "build.sh <SOURCE_DIR> https://github.com/leimark/fibonacci.git".

   To deploy the service:

   After ran build.sh and build success, run "<REPO_BASE>/fibonacci/scripts/deploy.sh".

Further considerations for production maintenance, robust, and performance etc.

As this is only a simple demo project, many things had not been implemented. If need to put the service in production for long term running, then must consider the improvement from the below aspects.

1. Overflow and pagination.
   If the inputted number is very huge, then the Fibonacci result may overflow. In this case, the normal "plus" operation is not suitable for this. May consider to use String operation to implement the sum of two huge numbers.

   Another consequence for a huge input number is the size of the result. The size of the sequence may consume large mount of memory and CPU time. Need to consider pagination the calculation of the Fibonacci sequence, calculate and return the result part by part in a reasonable size.

2. Log rotation.

In current code, logs are directly outputted to one log file. For production deployment, logs must be rotated in a certain interval.

3. Performance.

   The code uses the default http server provided by gin. This is ok for testing and demo. For production deployment, must consider to address high throughput, like using Nginx, LVS in front the gin for proxy and load balance.

4. Robustness.

   The code implements a monitoring crontab job to verify if the Fibonacci service is still running and returns the correct result. If deploy in production environment, must consider to add the CPU, memory monitoring etc.

5. Upgrade.
   If put the service in production environment, must consider how to upgrade the service with or without downtime. The simple deploy.sh could not cover the upgrade smoothly.