Offsite​ Test​ for​ DevOps

1. This is a technical test to understand our candidate better. Please try at your best efforts.
2. Please​ prepare​ document​ for​ ​the code​ / test​ case​ and​ detail​ setup​ instruction​ as a Git repository. Please use a random/meaningless project name for the git repository (e.g. 5848cf8a7dd2/wahaha)
3. The ​code​ needs​ to ​work​ after ​we​ clone ​it.
4. This test may contain bugs. Any bug fixing is appreciated.
5. Happy hacking!
6. We expect candidates can finish within 2 weeks for a fast feedback loop. For special scenarios please feel free and reach out to us.

Q1.​ Rate-limiting

**Background**    
Crypto.com is a public service, and is subjected to attacks. Some attackers might repeatedly make requests to Crypto.com, in an attempt to deny service (DoS) or brute-force passwords. One way to defend against such attacks is rate-limiting​.​

**Requirements**    
Write a program that will process a web server access log file and perform IP-based rate-limiting. The program outputs instructions for the load balancer to ban offending IP addresses.

Implement rate limiting using the following rules:

| **Condition** | **Result** |
| --- | --- |
| More than 40 requests from one IP address, in the past 1 minute | IP address banned for the next 10 minutes |
| More than 100 requests from one IP address, in the past 10 minutes | IP address banned for the next 1 hour |
| More than 20 requests to ​/login​ from one IP address, in the past 10 minutes | IP address banned for the next 2 hours |

The program reads web server logs in the Apache “Combined Log Format”​ . You can assume one IP address represents exactly one client.

Example input format:

| 222.219.188.0 - - [15/May/2019:17:12:55 +0800] "HEAD /settings HTTP/1.1" 200 5054 "" ""  202.158.64.0 - - [15/May/2019:17:20:44 +0800] "GET /fresh HTTP/1.1" 200 4990 "" ""  222.88.170.255 - - [15/May/2019:17:23:23 +0800] "GET /search HTTP/1.1" 200 4914 "" ""  88.52.98.255 - - [15/May/2019:17:26:52 +0800] "POST /login HTTP/1.1" 401 4916 "" ""  88.52.98.255 - - [15/May/2019:17:26:52 +0800] "POST /login HTTP/1.1" 401 4916 "" ""  88.52.98.255 - - [15/May/2019:17:26:53 +0800] "POST /login HTTP/1.1" 401 4916 "" ""  88.52.98.255 - - [15/May/2019:17:26:54 +0800] "POST /login HTTP/1.1" 401 4916 "" ""  62.41.64.0 - - [15/May/2019:17:27:01 +0800] "GET /fresh HTTP/1.1" 200 4990 "" ""  88.52.98.255 - - [15/May/2019:17:27:01 +0800] "POST /login HTTP/1.1" 401 4916 "" ""  ... |
| --- |

The program produces output in CSV format with 3 columns: (1)timestamp of action, (2)the action (​BAN​ or ​UNBAN​), and (3)IP address in question.

You can download an example input log file for testing, at ​<https://drive.google.com/file/d/1_Shlh5rX5eBbLC-VEhCI8_WIKuCOailS/view?usp=sharing>

Processing which must produce exactly the following output:

| 1546271816,BAN,58.236.203.13  1546277422,BAN,221.17.254.20  1546281160,UNBAN,221.17.254.20  1546285801,BAN,210.133.208.189  1546293587,UNBAN,210.133.208.189  1546297454,BAN,221.17.254.20  1546301070,UNBAN,221.17.254.20  1546310858,UNBAN,58.236.203.13 |
| --- |

Q2.​ AWS​ API​ programming

Please​ help​ to​ prepare​ a script​ to​ query​ AWS​ API​ and​ look up the public​ IP​ of the instance​ with the​ specific​ EC2​ Name​ tag.​ Then​ the script​ should​ execute​ the ssh​ command

ssh​ ec2-user@EC2\_PUBLIC\_IP​ .

Example​ interaction​ of​ the​ script​ (say​ the executable​ script​ is​ awssh​ ):

| $​awssh​api-server-002 ssh​ec2-user@120.45.256.57 [ec2-user@ip-120-45-256-57​~]$ $​awssh​not-existing-host Host​not​found |
| --- |

You​ may​ test​ against​ your​ code​ with​ real​ AWS​ account​ OR​ AWS​ EC2​ mock​ API​ moto (​ ​https://github.com/spulec/moto​ ​)

Q3. System design and Implementation

Please​ implement​ a URL​ shortener​ service (e.g. ​https://bitly.com/​)​

Implement​ a simple​ bit.ly​ like​ service,​ name​ the following​ 2 RESTful​ endpoints

1.​ RESTful​ endpoint​ for​ url​ submission

| POST /newurl   - Request: { "url": "​https://www.google.com​" }   - Response: { "url": "​https://www.google.com​", "shortenUrl":"https://shortenurl.org/g20hi3k9"} |
| --- |

2.​ Shorten​ redirect​ URL

| GET​ /[a-zA-Z0-9]{9}​ (regex,​ eg.​ g20hi3k9)  - HTTP​ 304​ to​ saved​ link​ (eg.​ https://www.google.com​​ according previous​ example) |
| --- |

- No​ update​ on the shorten​ link​ once​ created

System​ design​ concern

● High availability​:​ Please​ make​ it​ highly available and​ no​ single​ point​ ​of failure.  
  
● Scalability​ :​ Please​ make​ it​ scalable.  
   
● Scaling​ target​:​ 1000+​ req/s,​ after​ scaling-up/out without​ major​ code​ change

Tech stack

● You​ could choose​ any​ technology​ / programming​ language​ / database​ / cache​ / AWS​ services,​ as​ long​ as​ ​the application​ code​ is​ executable​ on​ generic​ Linux EC2​ node​ and​ ​the whole​ setup​ fits​ in​ AWS​ environment.

Deliverable​ & Documentation

● The​ application​ deliverable​ should​ be​ self-contained,​ preferably​ an​ automated  
   
deployment​ package / container image,​ such​ that​ we​ can​ deploy​ it​ easily  
   
● The​ system/infrastructure​ should​ be​ also​ documented​ OR​ automated​ (via​ e.g. shell​ script)  
   
● Both​ ​the system​ design​ and ​the actual​ system​ infrastructure​ configuration​ are the key​ measurement​ ​of this​ question.   
● Please​ briefly​ explain​ your​ system​ and​ say​ why​ you​ are going​ to​ implement like​ that.  
   
● Please​ state​ any​ assumption​ and​ limitation​ of​ ​the system​ implemented.