Scenario 1: Logging

In this scenario, you are tasked with creating a logging server for any number of other arbitrary pieces of technologies.
Your logs should have some common fields, but support any number of customizeable fields for an individual log entry. You should be able to effectively query them based on any of these fields.
How would you store your log entries? How would you allow users to submit log entries? How would you allow them to query log entries? How would you allow them to see their log entries? What would be your web server?

Answer:

1. Traditionally, we either store log in our database or in file system. The advantages of storing in database is that it is easy to set up and it has build-in query tool. But the drawback it is not as flexible as file. So, in this case, as a logging service, I would like to store my log as a file in a file system or distributed virtual file system.
2. If it supports instant log, I will choose scribe. (https://github.com/facebookarchive/scribe/wiki). And I will write a package(including IO buffer and log formatter), set up a scribe service and allocate a category. Users just import the package and use the instance I offer to log.

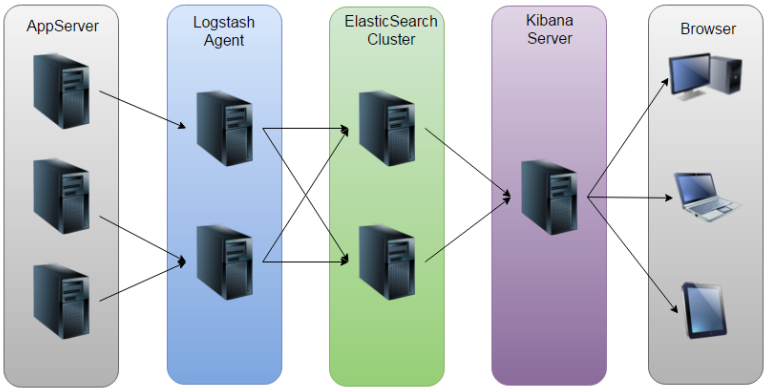
   If it doesn't require instant, we can just use the Linux syslog, like error_log in PHP. And then we can use Rsync tool to collect the local data. In this case, users can dump their log to local Rsync directory, and I will do synconize job every several hour.

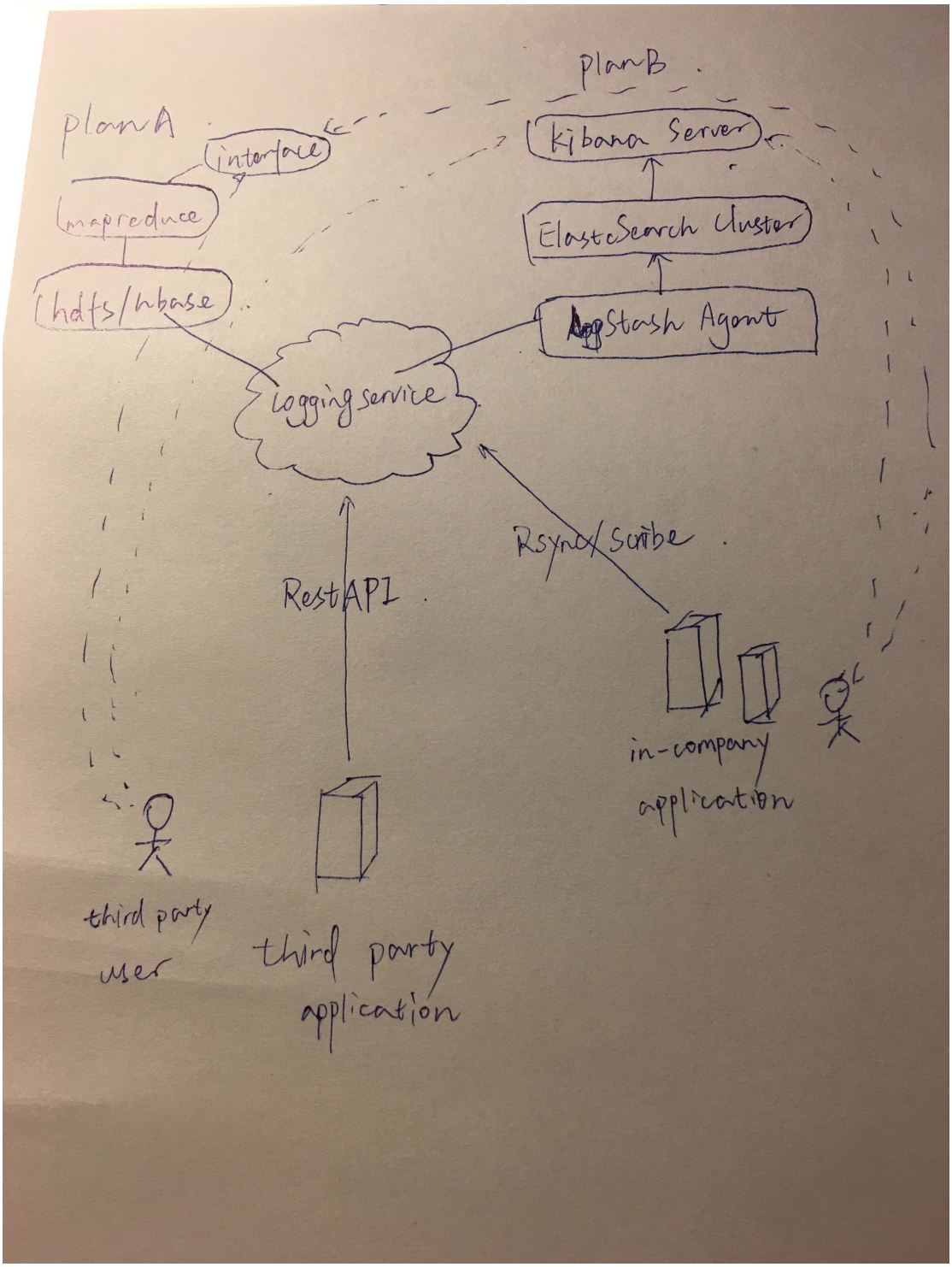
   If our service is a third-party service, we can offer some logging rest API. Users can call our API to transfer their log to us.
3. The easiest case is we store our log in database and we can

query using database build-in query. If we store in file, we can use some tools like Elasticserch + Logstash + Kibana or we can evaluate by our own.

4. They can login to the admin back-end and see the data by chart or by other form.
5. Either Nginx or Apache is OK. Nginx is better on Dynamic request, Apache is better on static resource. Again if we are an inner-service, not to have a web server is fine. We just get the data using Rsync or scribe service.

Logging service can be just like:





At the end, maybe we can just simply use Logentries lol.
(https://logentries.com)

Scenario 2: Expense Reports

In this scenario, you are tasked with making an expense reporting web application.
Users should be able to submit expenses, which are always of the same data structure: `id`, `user`, `isReimbursed`, `reimbursedBy`, `submittedOn`, `paidOn`, and `amount`.
When an expense is reimbursed you will generate a PDF and email it to the user who submitted the expense.

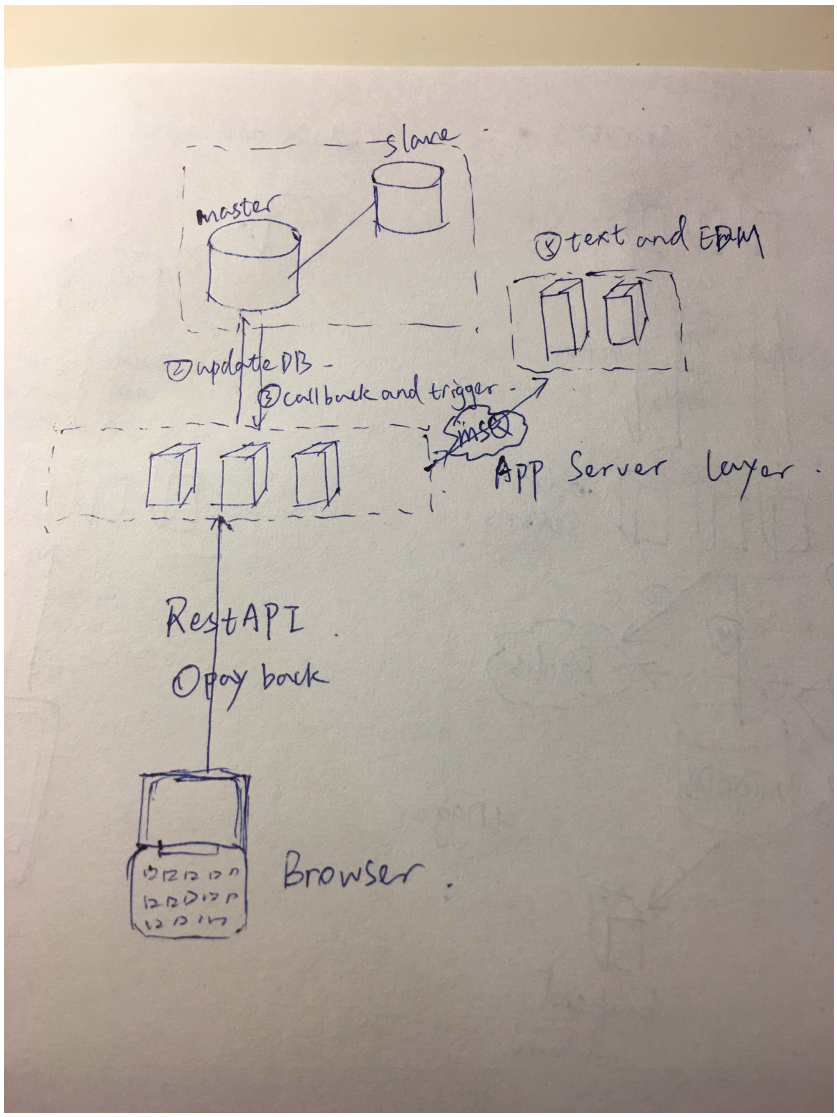
How would you store your expenses? What web server would you choose, and why? How would you handle the emails? How would you handle the PDF generation? How are you going to handle all the templating for the web application?

Answer:

1. I will store these data to relational database like mariaDB or postgreDB. If the write operation happens frequently, means allow people continuously expense in a short time. We can firstly save this data to Memcache and a MSQ, then save it to database. Or we can just simply store to redis.
2. I would like to use Nginx, because it is good at handling concurrent dynamic request.
3. When user pay back the money, our hook will send a message to MSQ. Our worker script receive the message and get the customer email inform from DB. And then using SMTP proxy to send mail. If your SMTP server using SSL or TSL, you should change your default config, like these two line in PHPMailer
   ```
   $mail->SMTPSecure = 'tls';
   $mail->Port = 587;
   Plus, if you want to generate a beautiful email, you should use MVC structure and a template engine.
   ```
4. Actually PHP has many tools to generate a PDF from html or xml

(https://packagist.org/search/?tags=pdf). We can also use LaTex or wkhtmltopdf.

5. We can use the Template Engine like smarty in PHP or blade in Nodejs. If we use XML in our template, we can use PHP XML Parser.



Scenario 3: A Twitter Streaming Safety Service

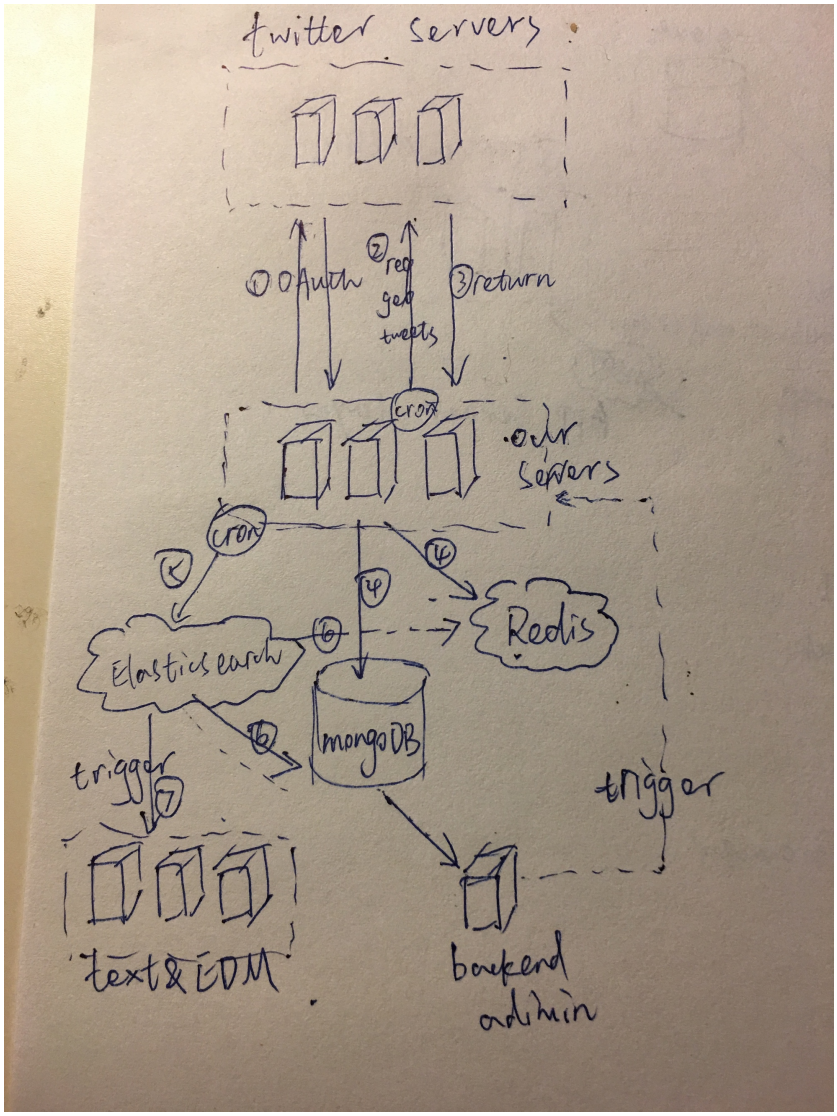
In this scenario, you are tasked with creating a service for your local Police Department that keeps track of Tweets within your area and scans for keywords to trigger an investigation.
This application comes with several parts:

- An online website to CRUD combinations of keywords to add to your trigger. For example, it would alert when a tweet contains the words (`fight` **or** `drugs`) AND (`SmallTown USA HS` or `SMUHS`).
- An email alerting system to alert different officers depending on the contents of the Tweet, who tweeted it, etc.
- A text alert system to inform officers for critical triggers (triggers that meet a combination that is marked as extremely important to note).
- A historical database to view possible incidents (tweets that triggered an alert) and to mark its investigation status.
- A historical log of *all* tweets to retroactively search through.
- A streaming, online incident report. This would allow you to see tweets as they are parsed and see their threat level. This updates in real time.
- A long term storage of all the media used by any tweets in your area (pictures, snapshots of the URL, etc).

Which Twitter API do you use? How would you build this so its expandable to beyond your local precinct? What would you do to make sure that this system is constantly stable? What would be your web server technology? What databases would you use for triggers? For the historical log of tweets? How would you handle the real time, streaming incident report? How would you handle storing all the media that you have to store as well? What web server technology

would you use?

Answer:
1.I use The Search API: Tweets by Place.
(https://dev.twitter.com/rest/public/search-by-place)
2.I will put the place into a configuration or insert into DB which can be update through the admin back-end interface.
3.We have roughly 5 components: (1) collect tweets (2) analyse tweets (3)email alert (4)test alert (5)backend. Our core program is the collection cron task and analyze script. To sure system stable means to ensure these two program stable. We can have a daemon to monitor this script. What's more, we can print logs like error_log and run_status_log.
4.web server tech including (1) MSQ because analyzing needs time, so we can use MSQ to avoid stuck. (2) HTTP Client to call the tweet API (3) OAuth to authorization (4) Redis to store most recent tweets. (5) Elasticsearch to make a full text search
5.I will use MongoDB, because MongoDB is built to store document type data just like tweet.
6. I can reuse the analyze program and just change a little bit of collection program. Now we can say when we push the update button in admin backend, it trigger the collection program and analyze program.
7.Either store in our own file or database, or we can put it into some web service like google drive, Dropbox so on.
8. Media is static resource, so we can use CDN to accelerate access speed and backup. And maybe we should also map the new media link and the related tweet.

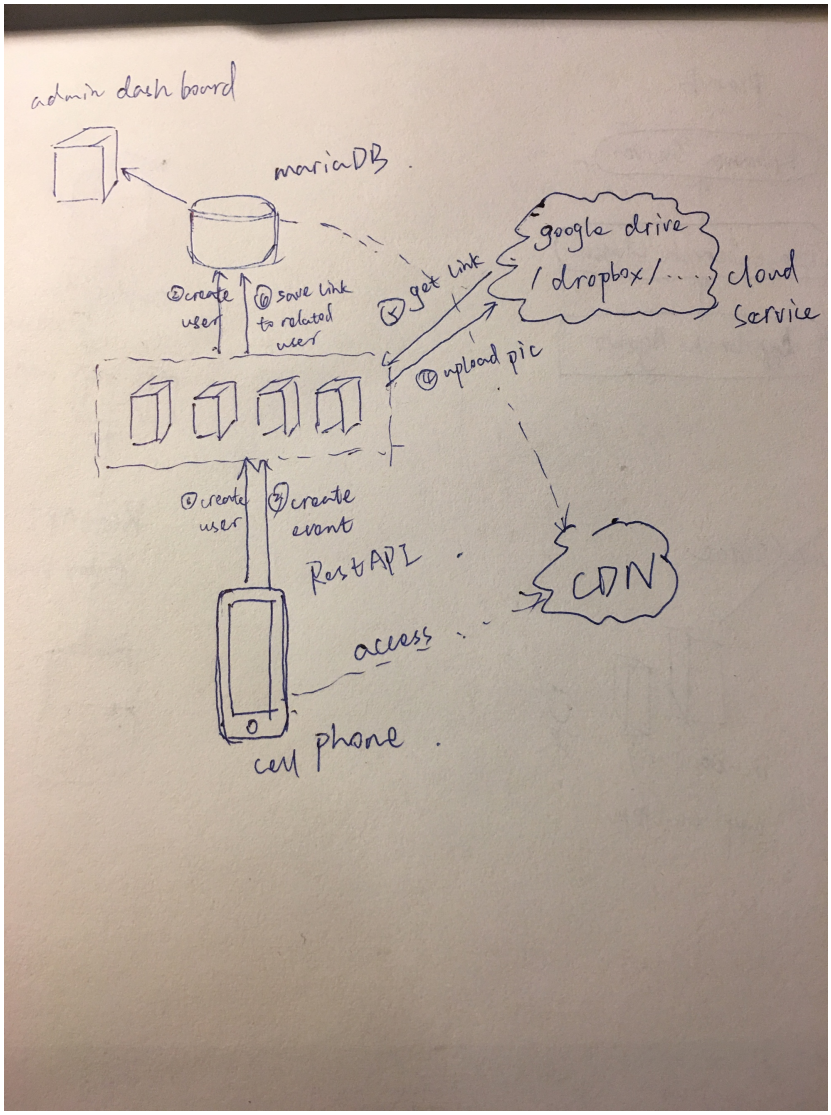

twitter servers
1 OAuth
2 req geo tweets
3 return
cron
our servers
cron
4
4
4
Elasticsearch
5
Redis
MongoDB
trigger
7
trigger
text&IDM
backend admin

Scenario 4: A Mildly Interesting Mobile Application

In this scenario, you are tasked with creating the web server side for a mobile application where people take pictures of mildly interesting things and upload them. The mobile application allows users to see mildly interesting pictures in their geographical location.
Users must have an account to use this service. Your backend will effectively amount to an API and a storage solution for CRUD users, CRUD 'interesting events', as well as an administrative dashboard for managing content.

How would you handle the geospatial nature of your data? How would you store images, both for long term, cheap storage and for short term, fast retrieval? What would you write your API in? What would be your database?
Answer:

1. We can use google location API and then tag our data. To accelerate access speed, we should use CDN.
2. For the long term storage, if I am in a large company, I will build my own Image Storage Service. If I am in a startup, I will choose cloud service like Dropbox to store. For the short term storage, pictures can be stored in Redis as a binary file.
3. First of all, Authorization. I will use password mode of Oauth2.0 to authorization. Also I will offer some RestfulAPIs like above say. Maybe some APIs as cropper or filter.
4. I would like to use relational database to store users inform and event inform, and put the pictures into clouds, just store the link of the resource.


admin dash board
mariaDB
google drive
/ dropbox / ...
cloud
service
get link
save link
to related
user
create
user
upload pic
create
user
create
event
Rest API
CDN
access
cell phone