Back of the envelope Requirements **Constraints** estimation. Users will be able to upload images. 1000 follow requests per second. Users will be able to see the images of users they follow. Million users base. 500 uploads/ sec Users will be able to follow other users . Don't use AWS. 5k likes /sec. Users will be able to like images of the people they follow. Availability, Latency. 1000 follow requests/ sec 1000 new image reqs/ sec Application is read as well as write heavy. Amazon **EKS** WAF getImage / image-id Get image by id user facing UI microservice Image get image from image datastore datastore using image id and send to UI. **Image** datastore Angular **EKS** image uploadImage user who uploaded timestamp no of likes uploadlmage non relational datastore which fetches from object microservice store input: image (jpg/jpeg) output: 200 OK image will be stored in datastore also, an event will be triggered to followers to notify them that a new image has arrived. **Events queue** This is one transaction. Asynchronous · As soon as any user uploads image, an event is microservices, hosted on VMs, docker generated. containers or kubernetes. Fetch the list of followers from user datastore and send a notification to them using some Push mechanism. Send the notification to user followUser level datastore/cache which is of minimal size. **EKS** followUser Kafka / Kinesis / Separate SQS for different events . microservice input: {user you want to follow, your user id}. output: 200 OK. user datastore: update no of **USER** followers update no of following. **USER** datastore also, an event will be triggered to user to notify them that someone has name, followed them. id This is one transaction no of followers Asynchronous no of following list of follower id list of following id

profile pic

Relational datastore



ike image

- User will be able to like image using this microservice.
- Input: image id and user id of the person who liked it.
- increment the no. of likes for the image in image datastore.
- send an event to events queue to let the image owner know that someone liked their image.



Notification datastore at user level

app level service

Notif datastore

- user level
- events queue sends the respective events of a user to the datastore.
- · Can be in-memory.
- Follows subscription mechanism instead of polling.
- Once the user has read the notifications, clear them.
- Event is of form:
 - user id who generated event
 - event type
 - if it is a follow request, then user who sent the request.
 - if it is a upload notification, then user who uploaded image and the image uploaded location/url.
- in-memory datastore.

App level background process

- Subscribes to the notification datastore.
- After it gets an event, based on the type, it sends the UI response to be shown in notification window.