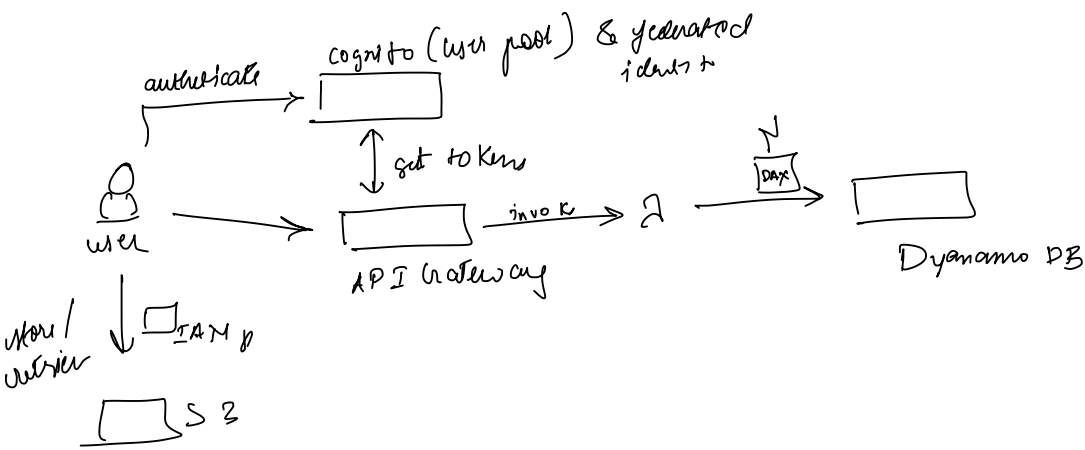
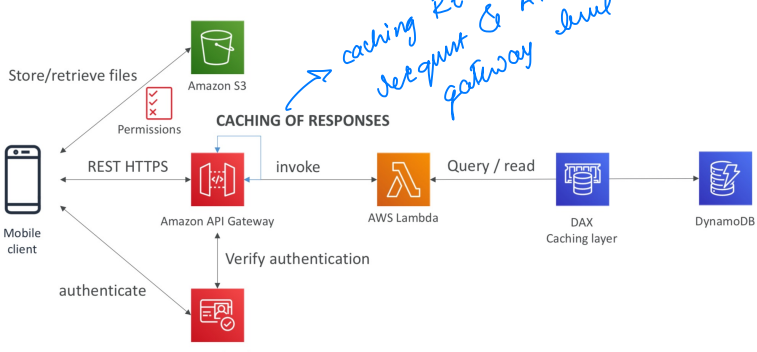
Serverless Solution Architectures

#1 My Todos List

- We want to create a mobile application with the following requirements

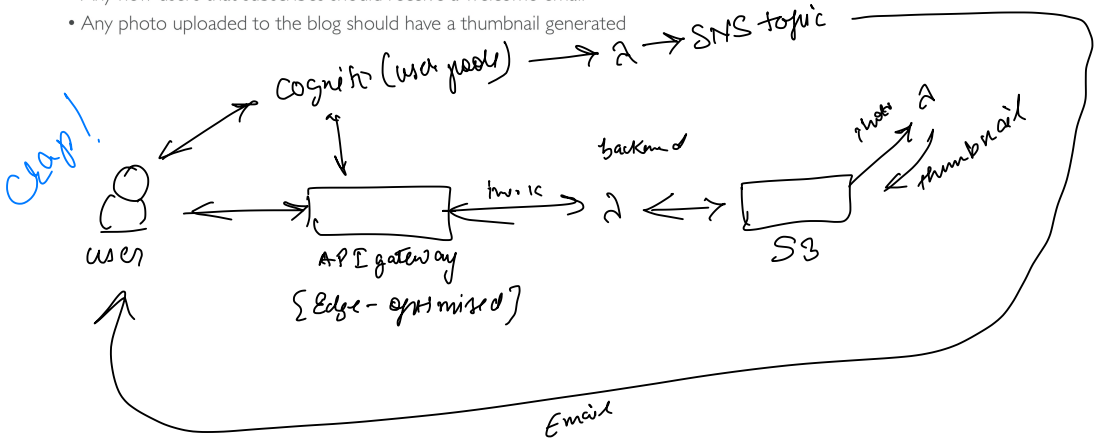
- Expose as REST API with HTTPS
- Serverless architecture
- Users should be able to directly interact with their own folder in S3
- Users should authenticate through a managed serverless service
- The users can write and read to-dos, but they mostly read them
- The database should scale, and have some high read throughput

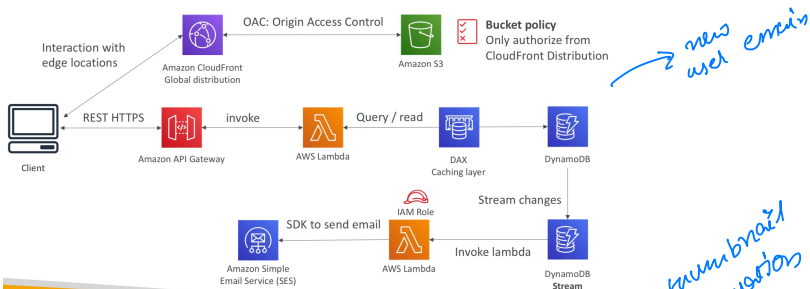


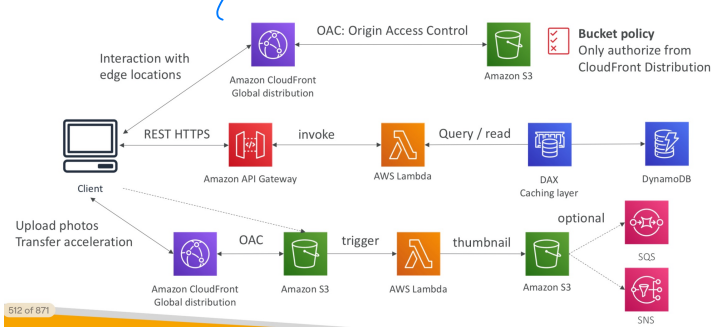


#2 My blogs.com

- This website should scale globally
- Blogs are rarely written, but often read
- Some of the website is purely static files, the rest is a dynamic REST API
- Caching must be implement where possible
- Any new users that subscribes should receive a welcome email
- Any photo uploaded to the blog should have a thumbnail generated







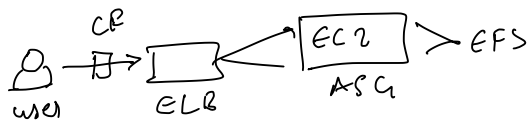
#3

- We want to switch to a micro service architecture
- Many services interact with each other directly using a REST API
- Each architecture for each micro service may vary in form and shape

- We want a micro-service architecture so we can have a leaner development lifecycle for each service

#4 Software update offloading

- We have an application running on EC2, that distributes software updates once in a while
- When a new software update is out, we get a lot of request and the content is distributed in mass over the network. It's very costly
- We don't want to change our application, but want to optimize our cost and CPU, how can we do it?



Why CloudFront?

- No changes to architecture
- Will cache software update files at the edge
- Software update files are not dynamic, they're static (never changing)
- Our EC2 instances aren't serverless
- But CloudFront is, and will scale for us
- Our ASG will not scale as much, and we'll save tremendously in EC2
- We'll also save in availability, network bandwidth cost, etc
- Easy way to make an existing application more scalable and cheaper!

cloudfront makes it super easy if we have static content to be distributed globally & can be cached for heavy reads!