**Week 12**

**12.1 Deploying Frontends on AWS**

* Objects are anything that are not stored on the database like mp4, html, css, jpeg files etc.
* We store objects in object stores. S3 is one such object store. Simple Storage Service – S3
* At scale, deploying frontend on AWS is much cheaper than Vercel.
* CDNs – Content Delivery Networks. For distribution. Cloudfront.
* CDNs manage a lot of pops (point of presence) so that when a request goes for a file, the files get cached in that specific pop for a certain time period. Pops are live smaller servers.
* All requests to object stores go through CDN.
* Usually distribution cost is higher, so you ask for discount on CDNs.
* For backends, edge networks are used which is deploying backend applications over multiple servers. The do not use caching as each user requests a unique data.
* We do not use edge networks for storage as the files are very large.
* npm run build to get HTML, CSS and JS files from the react project.
* serve serves all of your file over a port.
* We want to block public access because we don’t want our files to be distributed through S3 as it is expensive. We want it be distributed through Cloudfront.
* Generate the policy from Cloudfront and edit the bucket policy in S3 to allow Cloundfront to access S3.
* Find cheap domains on namecheap.com
* You can add alternate domain names through Cloudfront.
* You can also generate free SSL certificate for your new domain name through AWS or put one bought from GoDaddy etc. Amazon would verify by giving you a specific domain to put in your DNS
* Finally you need to put you new domain in the DNS pointing to the Cloudfront URL.
* You can also add error pages if the user tries to hit an invalid page through Cloudfront.
* To remove all caches from the Cloudfront pops, you can go to Invalidations and add /\*
* CI/CD pipeline whenever you push your code to GitHub it automatically deploys to S3.
* **Summary:** Deployed frontend on AWS and connected custom domain – Only HTML/CSS/JS files were uploaded to S3. Distribution through CDN – Cloudfront. SSL for custom domain requested through AWS, verified, new domain pointed to Cloudfront URL, added error page.

**12.2 Advance Typescript APIs**

* Pick – lets you select a set of properties from an existing type or interface.
* Partial
* Readonly and readonly.
* Record and Map.
* Exclude.
* Zod - export type finalUserSchema = z.infer<typeof userProfileSchema>;
* Objects and maps are two ways to do key value pairs.

**Deploying NPM Packages and Intro to Monorepos**

* Zod inference so that we can verify inputs at the frontend but the types is inferred from the backend.
* You never publish the src folder and you exclude it by adding “src” to .npmignore using vi.
* .d.ts file just contains the declarations, inferences, and types. No JS logic.
* Whenever you publish your package, you also need to publish .d.ts file so that the user using your package can get the types. Add “declaration”: true in tsconfig.json and compile again to get the .d.ts file. Imagine this file as a header file that shows all the return types, the input types, interfaces etc.
* No one know publish npm packages to get code from backend to frontend or vice versa. We use mono repos now.
* Turborepo is one mono repo. Just create components in ui folder and you can import them in your frontend because frontend’s package.json has “ui”: “\*” as a dependency.