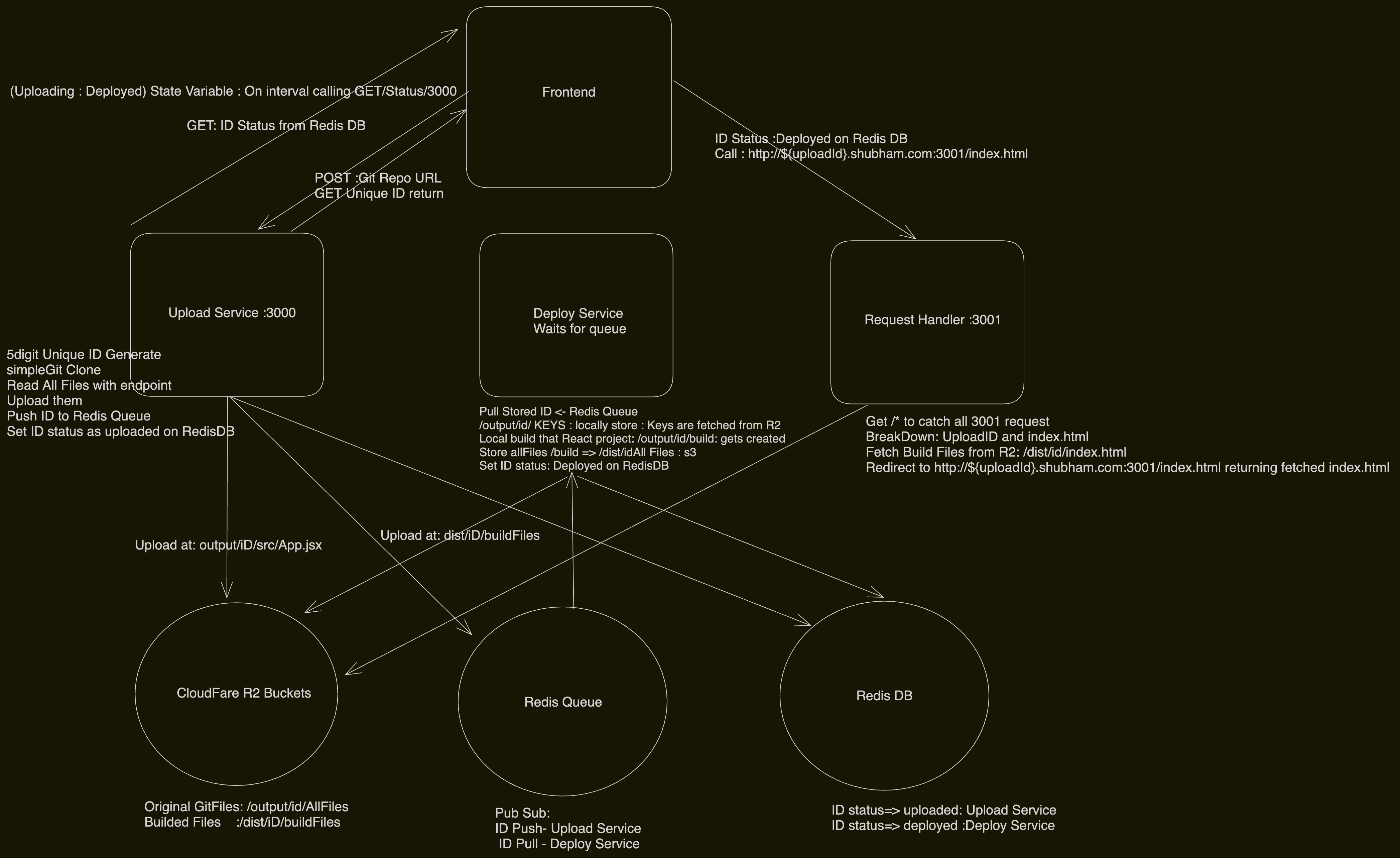
Vercel Deploying App:

This app takes your github repo as url, the app should be build in react. Uploads that app to R2 Cloudfare bucket using Upload Service by reading all files and using that endpoint of each file store in R2 in /output/UniqueIdGenerated/ All Files ending part sliced.  
Then this Id is passed to using Redis QUEUE Deploy Service whose work is to first fetch all uploaded git files and npm run build that file and then store builded file in R2. I used files endpoint by slicing those names and stored at R2 /dist/id/builded file which is mapped locally to /output/id/build/builded files  
  
Frontend first gives github url to upload Service which pass to Deploy Service. Upto here all our files ploading and building that file is done.  
We used Redis DB to keep track of ID status once upload service is over id:Uploaded once deploy server is over id:Deployed  
Frontend simply uses useState and when button is pressed we used promises to be in sync.  
  
Once we get deployed status we present a url http://${uploadId}.shubham.com:3001/index.html

Then a Request Service is made which is listening at 3001 this url is breaked down into UniqueID and Key: index.html. This service download index.html from R2 using this uniqueId by going into /dist/id/index.html into R2.  
and this service return index.html which is showed in browser.  
  
  
Since we don’t have external server we have mapped our localmachine IP to http://${uploadId}.shubham.com . Now whenever this ip is called it is our machine that is listening similar to localhost :3001.  
  
Learning- R2 Buckets, Redis Queue, DB, Microservices, Deployment  
  
Let me tell you how this app works it takes github repo link as input and according to that generates a deployed http url where that repo code is running Internally I am using microservices first service fetch git repo using third party library in react clone that to local backend and generate a unique id and push that complete code to R2 buckets on cloudfare Now this unique id is passed to Rabbit Queue used as pub sub model Who is talking to another service that once gets notified from queue fetch the unique id and copy that file into local system then it uses pseudo terminal build that file and the output of that build file is again stored in R2 since that build file directly contains html file which shows the net builded repo code Another service is where we give a Url to user which when click consisit of unique id of builded repo this url calls this service and take unique it fetched builded file from R2 and return the html to get displayed on the browser  
  
Reference->Harkirat

https://projects.100xdevs.com/tracks/ZSQI8YNE0iL6sT1hJpts/vercel-1

Each Service Steps.txt provided with the steps to build that

1-Upload Service

2-Deploy Service

3-RequestHandlerService

4-Frontend

Upload Service  
https://projects.100xdevs.com/tracks/ZSQI8YNE0iL6sT1hJpts/vercel-1

1-Install Node.js

2-Initialise an empty typescript project

npm init -y

npm install typescript

npx tsc --init

example : index.ts

npx tsc -> convert ts to js output to folder mentioned in tsconfig.json

3-Basic typescript configuration

Change rootDir to src

Change outDir to dist for the pro

4-Add express , redis , aws-sdk , simple-git , cors as dependencies

npm install express @types/express @types/cors redis aws-sdk simple-git cors

// types/express -> typescript express write help res:no need to tell type req:no need

aws-sdk-> s3 sqs queue simple-git -build wager help

5-Initialize a simple express app in index.ts listening on port 3000

// build any project npx tsc -b TS->JS DIST mein convert

6-Initialise an endpoint that the user will hit and send the repo url as input

app.post("",aysnc(){})

7-Create a function that randomly generates an id for this session. Call it generate

const MAX\_LEN = 5;

function generate() {

let ans = "";

const subset = "123456789qwertyuiopasdfghjklzxcvbnm";

for (let i = 0; i < MAX\_LEN; i++) {

ans += subset[Math.floor(Math.random() \* subset.length)];

}

return ans;

}

8-Use simple-git to clone the repo into a new folder (/out/id ).

import simpleGit from "simple-git";

await simpleGit().clone(repoUrl, path.join(\_\_dirname, `output/${id}`));

\_\_dirname looks like /user/shubham/from root to current folder

9-Write a function that gets the paths of all the files in the /out/id folder

since aws-sdk sdk.uploadFile() // need all the files in id folder

getAllFiles() calls recursively folder src/file1 src/file2 inko separately path nikalo

["path/id/file1","path/id/file2","path/id/file3"] uploadFile(1) then 2then 3

import fs from "fs";

import path from "path";

export const getAllFiles = (folderPath: string) => {

let response: string[] = [];

const allFilesAndFolders = fs.readdirSync(folderPath);allFilesAndFolders.forEach(file => {

const fullFilePath = path.join(folderPath, file);

if (fs.statSync(fullFilePath).isDirectory()) {

response = response.concat(getAllFiles(fullFilePath))

} else {

response.push(fullFilePath);

}

});

return response;

}

10-Create an AWS/CLOUDFARE account // aws-sdk works with CLOUDFARE account s3->r2 yaha simply

Account ID and API token create for accessing buckets

Token value

GJjjJpSSuADyWfwoxZjCccbgnp0wJ8e86anoSVFI

Access Key ID

50dffddf624ff1a6735ff7623596130e

Secret Access Key

88d02674da7cf5f101e95b0bf51eb69d280f9ec2fe6ac924d31710bd5bfea138

endpoint

https://d7a4c42a024029f72169ec6faf5cb79c.r2.cloudflarestorage.com

11-Write a function that uploads a file given a path to S3

Iterate over all the files and upload them to S3 one by one (or together)

s3.upload({

Body: fileContent,

Bucket: "vercel",

Key: fileName, fileName => output/12312/src/App.jsx

// output->12312->src->file

//this fileName will be in folder structure at cloudfare automatically

}).promise();

await uploadFile(file.slice(\_\_dirname.length + 1), file);

//all files upload with key name

// x=12345 then x.slice(2) return 345

// for key /Users/shubham/vercel/dist/output/12312/src/App.jsx => output/12312/src/App.jsx trim kar rahe

12-Start redis locally

nodejs->(lpush(1))->redisQueue->rpop()->AnotherNodejs

docker run --name some-redis -p 6379:6379 -d redis //pull image

13-Initialize a redis publisher

import { createClient } from "redis";

const publisher = createClient();

publisher.connect(); //by defualt locally running redis connect // otherwise need to give credentials

14- Use redis queues to push the uploadId in the queue

publisher.lPush("build-queue", id);

//check toh redis-cli open RPOP build-queue // same queue name

15-Also store the current video id’s status as uploaded .

publisher.hSet("status", id, "uploaded"); // set the status of the id

16-Expose a status endpoint that the frontend will poll to get back the status of a video. It needs to check redis for the current value.

const subscriber = createClient(); //to get data from redis

subscriber.connect();

// In redis pub can only set data and subscirber can only get data

// dono get set alone ek nahi karega

app.get("/status", async (req, res) => {

const id = req.query.id;

const response = await subscriber.hGet("status", id as string);

res.json({

status: response

})

})

publisher.connect();

Deploy service

1-Initialise an empty typescript project.

npm init -y

npm install typescript

npx tsc --init

2-Configure the tsconfig.json.

Change rootDir to src

Change outDir to dist for the pro

3-Installing dependencies

npm install express @types/express @types/cors redis aws-sdk simple-git cors

4-Initialize a redis subscriber

import { createClient, commandOptions } from "redis";

const subscriber = createClient();

subscriber.connect(); //by defualt locally running redis connect // otherwise need to give credentials

// localhost:6379

5-Create an infinitely running for loop that pulls values from the redis queue.

async function main() {

while(1) {

const res = await subscriber.brPop( //right pop //pushing from left tha

commandOptions({ isolated: true }),

'build-queue',

0 //wait infinitely untill something is coming // can put timeout if needed

);

console.log(res.element)

}

}

main();

//CHECK redis-cli jao LPUSH queuename value

6-Write a function called downloadS3Folder that downloads all the files from a given location in S3.

@ts-ignore // this tells typesciript to ignore below error aage badh sakta

// check aws.ts commented\*\*

//1st list down all keys

const allFiles = await s3.listObjectsV2({

Bucket: "vercel",

Prefix: prefix

}).promise();

//2nd keys se ouput folder mein stream karo donwload karo

const outputFile = fs.createWriteStream(finalOutputPath); //downloading from internet

s3.getObject({ //reading file and output to output file which is capable to write

Bucket: "vercel",

Key

}).createReadStream().pipe(outputFile).on("finish", () => {

resolve("");

})

Theory

A promise is an object that represents the eventual completion (or failure)

of an asynchronous operation and its resulting value.

Promises provide a cleaner and more manageable way to handle asynchronous code

compared to traditional callback-based approaches.

7-Run npm run build to convert the React code into HTML/CSS files. (Bonus if this is containerized).

utils.js

const child = exec(`cd ${path.join(\_\_dirname, `output/${id}`)} && npm install && npm run build`)

child.on('close', function(code) {

resolve("")

});

npm install npm run build ek child process banake kiye hai

8-Write a function that uploads a directory to S3 (you can copy it from the last module).

Same as upload service

waha output/id/ filesnames

all builded files uthao /output/id/build create hoga usko map karo

here finally store dist/id/ filenames

9- Store in the redis database that this specific upload has been processed.

const publisher = createClient();

publisher.hSet("status", id, "deployed")

Request handler

1-Initialize a Node.js Project, add TS configurations

npm init -y

npm install typescript

npx tsc --init

Configure the tsconfig.json.:

Change rootDir to src

Change outDir to dist for the pro

npm install express @types/express @types/cors redis aws-sdk simple-git cors

2-Initialize an express server running on port 3001

app.listen(3001,()=>{

console.log("Listening")

});

3-\*\* Making somewebsite points to your machine like localhost point to my machine

cd /e

vi /etc/hosts

add your website 127.0.0.1 mywebsite.com

if ssl error use http instead https use incognito

4-Add a global route catch (/\*) which handles all requests

app.get("/\*", async (req, res) => {

})

5-Extract the sub-domain the request is coming from (id.vercel.com ⇒ id)

{

const host = req.hostname; // hostname gives me complete host mywebsite.com

const id = host.split(".")[0]; // getting first dot before

}

6-Get the contents from S3 assuming the subdomain represents the id and forward it to the user. Add the correct content-type header to ensure the final file is parsed as a html file.

const type = filePath.endsWith("html") ? "text/html" : filePath.endsWith("css") ? "text/css" : "application/javascript"

res.set("Content-Type", type);