

Upload service

▼ Install Node.js

Official Page - <https://nodejs.org/en/download>

Good resource to follow along - <https://www.digitalocean.com/community/tutorials/how-to-install-node-js-on-ubuntu-20-04>

▼ Initialise an empty typescript project

```
mkdir vercel
cd vercel
npm init -y
npx tsc --init
```

[Copy](#)

▼ Basic typescript configuration

1. Change `rootDir` to `src`
2. Change `outDir` to `dist` for the pro

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

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

[Copy](#)

▼ Initialize a simple express app in `index.ts` listening on port `3000`

▼ Initialise an endpoint that the user will hit and send the `repo url` as input

```
import express from "express";
import cors from "cors";
import { generate } from "../utils";

const app = express();
app.use(cors());
app.use(express.json());

// POSTMAN
app.post("/deploy", async (req, res) => {
  const repoUrl = req.body.repoUrl;
});

app.listen(3000);
```

[Copy](#)

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

```
function generate() {  
  const subset = "123456789qwertyuiopasdfghjklzxcvbnm";  
  const length = 5;  
  const id = "";  
  for (let i = 0; i < length; i++) {  
    id += subset[Math.floor(Math.random() * subset.length)];  
  }  
  return id;  
}
```

Copy

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

```
import express from "express";  
import cors from "cors";  
import simpleGit from "simple-git";  
import { generate } from "../utils";  
  
const app = express();  
app.use(cors());  
app.use(express.json());  
  
// POSTMAN  
app.post("/deploy", async (req, res) => {  
  const repoUrl = req.body.repoUrl;  
  const id = generate(); // asd12  
  await simpleGit().clone(repoUrl, `output/${id}`);  
  
  res.json({  
    id: id  
  })  
});  
  
app.listen(3000);
```

Copy

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

```
import fs from "fs";  
import path from "path";  
  
export const getAllFiles = (folderPath: string) => {  
  let response: string[] = [];  
  // ...  
}
```

Copy

```

    const allFilesAndFolders = fs.readdirSync(folderPath);allFilesAndF
    const fullFilePath = path.join(folderPath, file);
    if (fs.statSync(fullFilePath).isDirectory()) {
        response = response.concat(getAllFiles(fullFilePath))
    } else {
        response.push(fullFilePath);
    }
  });
  return response;
}

```

▼ Create an AWS account

<https://cloudflare.net/>

<https://aws.amazon.com/>

▼ Write a function that uploads a file given a `path` to S3

```

import { S3 } from "aws-sdk";
import fs from "fs";

// replace with your own credentials
const s3 = new S3({
  accessKeyId: "7ea9c3f8c7f0f26f0d21c5ce99d1ad6a",
  secretAccessKey: "b4df203781dd711223ce931a2d7ca269cdbf81bb530de454",
  endpoint: "https://e21220f4758c0870ba9c388712d42ef2.r2.cloudflare.s3.amazonaws.com",
});

// fileName => output/12312/src/App.jsx
// filePath => /Users/harkiratsingh/vercel/dist/output/12312/src/App.j
export const uploadFile = async (fileName: string, localFilePath: string) => {
  const fileContent = fs.readFileSync(localFilePath);
  const response = await s3.upload({
    Body: fileContent,
    Bucket: "vercel",
    Key: fileName,
  }).promise();
  console.log(response);
}

```

Copy

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

```

const files = getAllFiles(path.join(__dirname, `output/${id}`));
files.forEach(async file => {

```

Copy

```
    await uploadFile(file.slice(__dirname.length + 1), file);  
  })
```

▼ Start redis locally

<https://developer.redis.com/create/windows/>

▼ Initialize a redis publisher

```
import { createClient } from "redis";  
const publisher = createClient();  
publisher.connect();
```

Copy

▼ Use `redis queues` to push the `uploadId` in the queue

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

Copy

▼ Also store the current video id's status as `uploaded` .

```
publisher.hSet("status", id, "uploaded");
```

Copy

▼ 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.

```
app.get("/status", async (req, res) => {  
  const id = req.query.id;  
  const response = await subscriber.hGet("status", id as string);  
  res.json({  
    status: response  
  })  
})
```

Copy

Deploy service

▼ Initialise an empty typescript project.

```
npm init -y
```

Copy

▼ Configure the `tsconfig.json`.

```
npx tsc --init
```

Copy

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

```
import { createClient, commandOptions } from "redis";
import { copyFinalDist, downloadS3Folder } from "./aws";
import { buildProject } from "./utils";

async function main() {
  while(1) {
    const res = await subscriber.brPop(
      commandOptions({ isolated: true }),
      'build-queue',
      0
    );
    console.log(res.element)
  }
}
main();
```

Copy

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

```
import { S3 } from "aws-sdk";
import fs from "fs";
import path from "path";

const s3 = new S3({
  accessKeyId: "7ea9c3f8c7f0f26f0d21c5ce99d1ad6a",
  secretAccessKey: "b4df203781dd711223ce931a2d7ca269cdbf81bb530de454",
  endpoint: "https://e21220f4758c0870ba9c388712d42ef2.r2.cloudflares"
});

// output/asdasd
export async function downloadS3Folder(prefix: string) {
  const allFiles = await s3.listObjectsV2({
    Bucket: "vercel",
    Prefix: prefix
  });
```

Copy

```

    }).promise();

    //
    const allPromises = allFiles.Contents?.map(async ({Key}) => {
      return new Promise(async (resolve) => {
        if (!Key) {
          resolve("");
          return;
        }
        const finalOutputPath = path.join(__dirname, Key);
        const outputFile = fs.createWriteStream(finalOutputPath);
        const dirName = path.dirname(finalOutputPath);
        if (!fs.existsSync(dirName)){
          fs.mkdirSync(dirName, { recursive: true });
        }
        s3.getObject({
          Bucket: "vercel",
          Key
        }).createReadStream().pipe(outputFile).on("finish", () =>
          resolve(""));
        })
      }) || []
    console.log("awaiting");

    await Promise.all(allPromises?.filter(x => x !== undefined));
  }

```

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

```

import { exec, spawn } from "child_process";
import path from "path";

export function buildProject(id: string) {
  return new Promise((resolve) => {
    const child = exec(`cd ${path.join(__dirname, `output/${id}`)}`);

    child.stdout?.on('data', function(data) {
      console.log('stdout: ' + data);
    });
    child.stderr?.on('data', function(data) {
      console.log('stderr: ' + data);
    });
  });
}

```

Copy

```

        child.on('close', function(code) {
            resolve("")
        });
    })
}

```

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

Copy

```

export function copyFinalDist(id: string) {
    const folderPath = path.join(__dirname, `output/${id}/dist`);
    const allFiles = getAllFiles(folderPath);
    allFiles.forEach(file => {
        uploadFile(`dist/${id}/` + file.slice(folderPath.length + 1),
    })
}

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;
}

const uploadFile = async (fileName: string, localFilePath: string) => {
    const fileContent = fs.readFileSync(localFilePath);
    const response = await s3.upload({
        Body: fileContent,
        Bucket: "vercel",
        Key: fileName,
    }).promise();
    console.log(response);
}

```

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

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

Copy

Request handler

▼ Initialize a Node.js Project, add TS configurations

```
npm init -y  
npx tsc --init
```

Copy

▼ Initialize an express server running on port 3001

```
import express from "express";  
import { S3 } from "aws-sdk";  
  
const app = express();  
  
app.listen(3000)
```

Copy

▼ Add a global route catch (/*) which handles all requests

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

Copy

▼ Extract the sub-domain the request is coming from (id.vercel.com => id)

```
const host = req.hostname;  
const id = host.split(".")[0];
```

Copy

▼ 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.

Copy

```
import express from "express";
import { S3 } from "aws-sdk";

const s3 = new S3({
  accessKeyId: "7ea9c3f8c7f0f26f0d21c5ce99d1ad6a",
  secretAccessKey: "b4df203781dd711223ce931a2d7ca269cdbf81bb530de454",
  endpoint: "https://e21220f4758c0870ba9c388712d42ef2.r2.cloudflares.com"
});

const app = express();

app.get("/*", async (req, res) => {
  // id.100xdevs.com
  const host = req.hostname;

  const id = host.split(".")[0];
  const filePath = req.path;

  const contents = await s3.getObject({
    Bucket: "vercel",
    Key: `dist/${id}${filePath}`
  }).promise();

  const type = filePath.endsWith(".html") ? "text/html" : filePath.endsWith(".css") ? "text/css" : "application/javascript";
  res.set("Content-Type", type);

  res.send(contents.Body);
});

app.listen(3001);
```

Frontend

Code - <https://github.com/hkirat/vercel/tree/main/frontend>

The project involves building a simple form that let's users send requests to the services we made in the last points