Microservices

SWE 432, Fall 2019

Web Application Development

Quiz

Go to:
b.socrative.com, Click student login
Room name: SWE432
Student Name: Your G-number (Including the G)

Reminder: Survey can only be completed if you are in class. If you are not in class and do it you will be referred directly to the honor code board, no questions asked, no warning.

Blobs: Storing uploaded files

- Example: User uploads picture
  - … and then?
  - … somehow process the file?

How do we store our files?

- Dealing with text is easy - we already figured out firebase
  - Could use other databases too… but that's another class!
- But
  - What about pictures?
  - What about movies?
  - What about big huge text files?
- Aka…Binary Large OBject (BLOB)
  - Collection of binary data stored as a single entity
  - Generic terms for an entity that is array of bytes

Working with Blobs

- Module: multer

- Simplest case: take a file, save it on the server

```
app.post('/upload',upload.single("upload"), function(req, res) {
        var sampleFile = req.file.filename;
      //sampleFile is the name of the file that now is living on our server
        res.send('File uploaded!');
    });
});
```

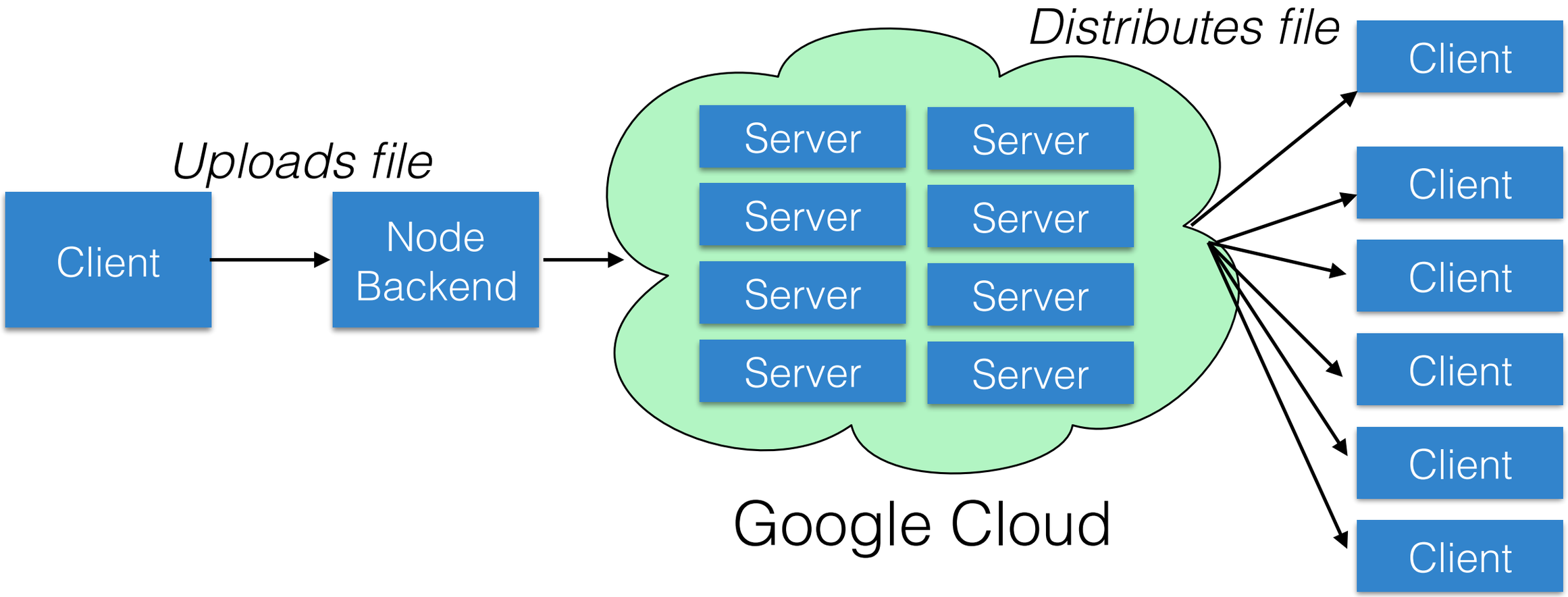
- Long story... can't easily have file uploads and JSON requests at the same time

Where to store blobs

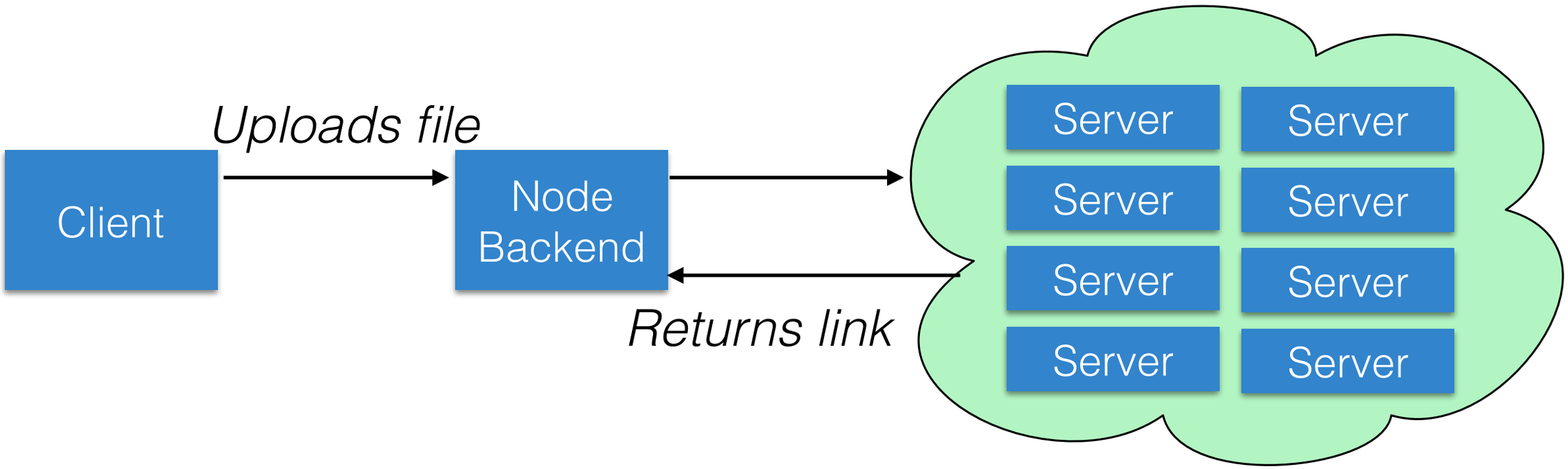
- Saving them on our server is fine, but…
  - What if we don't want to deal with making sure we have enough storage
  - What if we don't want to deal with backing up those files
  - What if our app has too many requests for one server and state needs to be shared between load-balanced servers
  - What if we want someone else to deal with administering a server

Blob stores

- Amazon, Google, and others want to let you use their platform to solve this!



Blob Stores



Typical workflow:

Client uploads file to your backend

Backend persists file to blob store

Backend saves link to file, e.g. in Firebase

Google Cloud Storage

- You get to store 5GB for free (but not used in this class)

- Setup

```
npm install --save @google-cloud/storage
```

```
// Imports the Google Cloud client library
const {Storage} = require('@google-cloud/storage');

// Creates a client
const storage = new Storage();

/**
 * TODO(developer): Uncomment these variables before running the sample.
 */
// const bucketName = 'bucket-name';

async function createBucket() {
  // Creates the new bucket
  await storage.createBucket(bucketName);
  console.log(`Bucket ${bucketName} created.`);
}

createBucket();
```

- https://cloud.google.com/storage/docs/reference/libraries

Google Cloud Storage

```
await storage.bucket(bucketName).upload(filename, {
  gzip: true,
  metadata: {
    cacheControl: 'public, max-age=31536000',
  },
});

console.log(`${filename} uploaded to ${bucketName}.`);

const options = {
  // The path to which the file should be downloaded, e.g. "./file.txt"
  destination: destFilename,
};

// Downloads the file
await storage
  .bucket(bucketName)
  .file(srcFilename)
  .download(options);

console.log(
  `gs://${bucketName}/${srcFilename} downloaded to ${destFilename}.`
);
```

- https://cloud.google.com/storage/docs/reference/libraries

Demo: Let's build a microservice!

- We've now seen most of the key concepts in building a microservice.
- Let's build a microservice!
  - - Firebase for persistence
  - - Handle post requests
  - Microservice for madlibs
  -