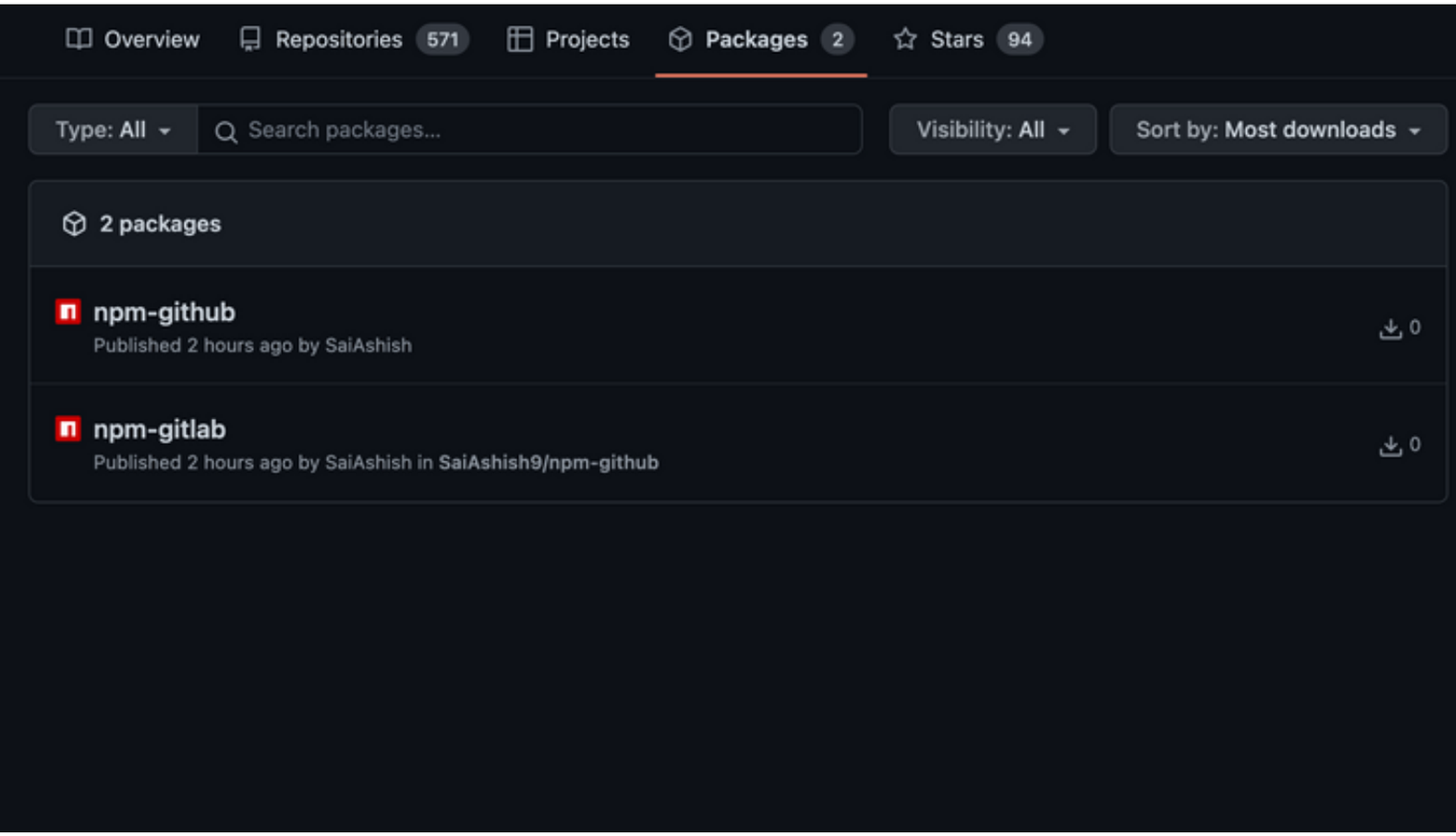
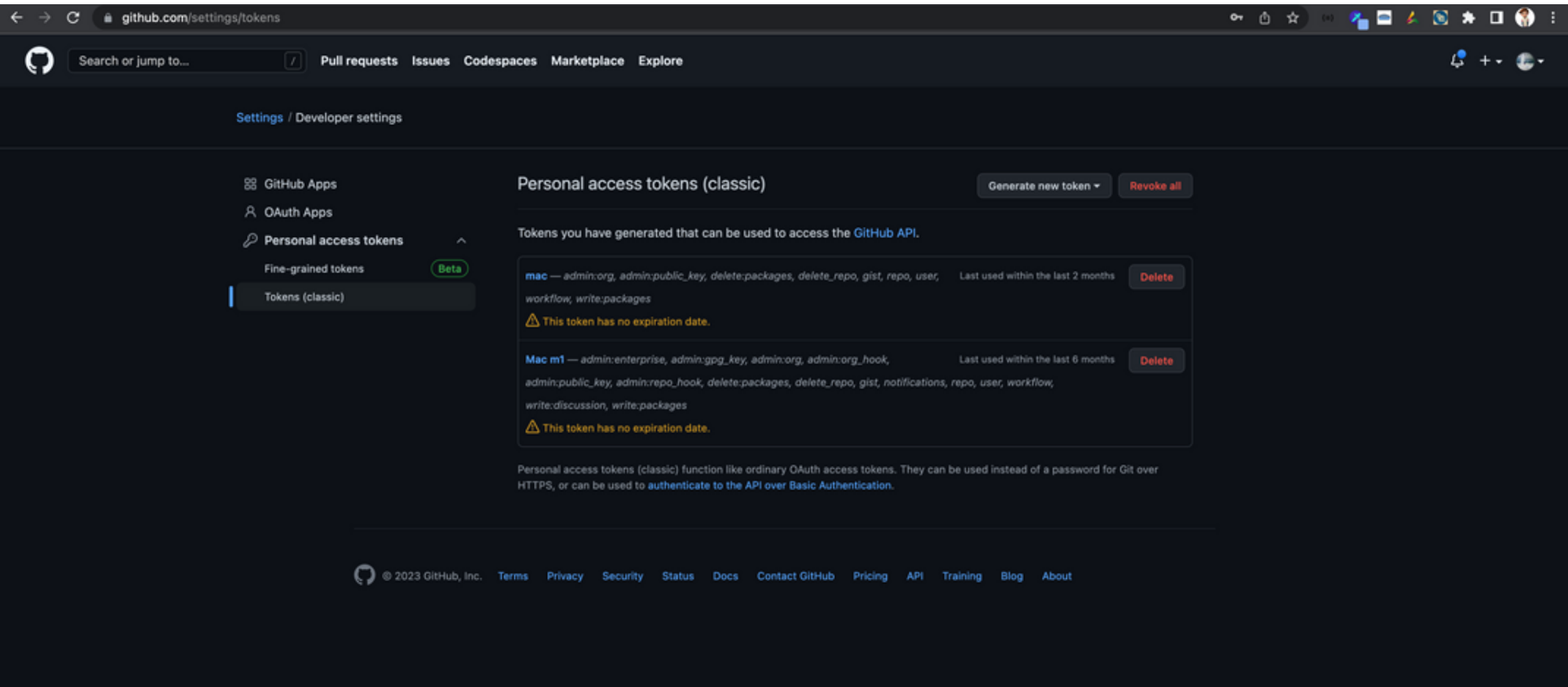
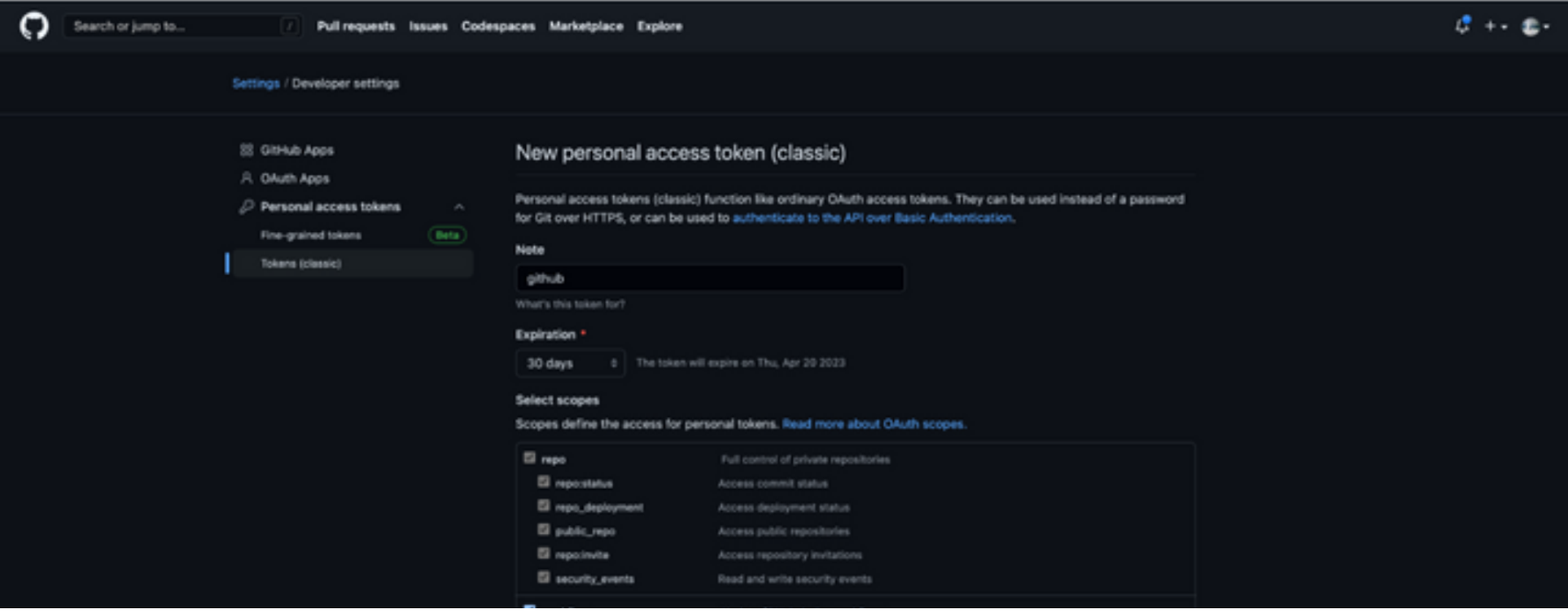
Npm, Github, Gitlab Package Registries ,
RunKit Browser Testing, Monolithic Approach,
Microservices, Event Driven Approach,
Organisational MonoRepos,
Project Level MonoRepos At Swiggy, PolyRepo, &
MicroFrontends

Github Public Packages

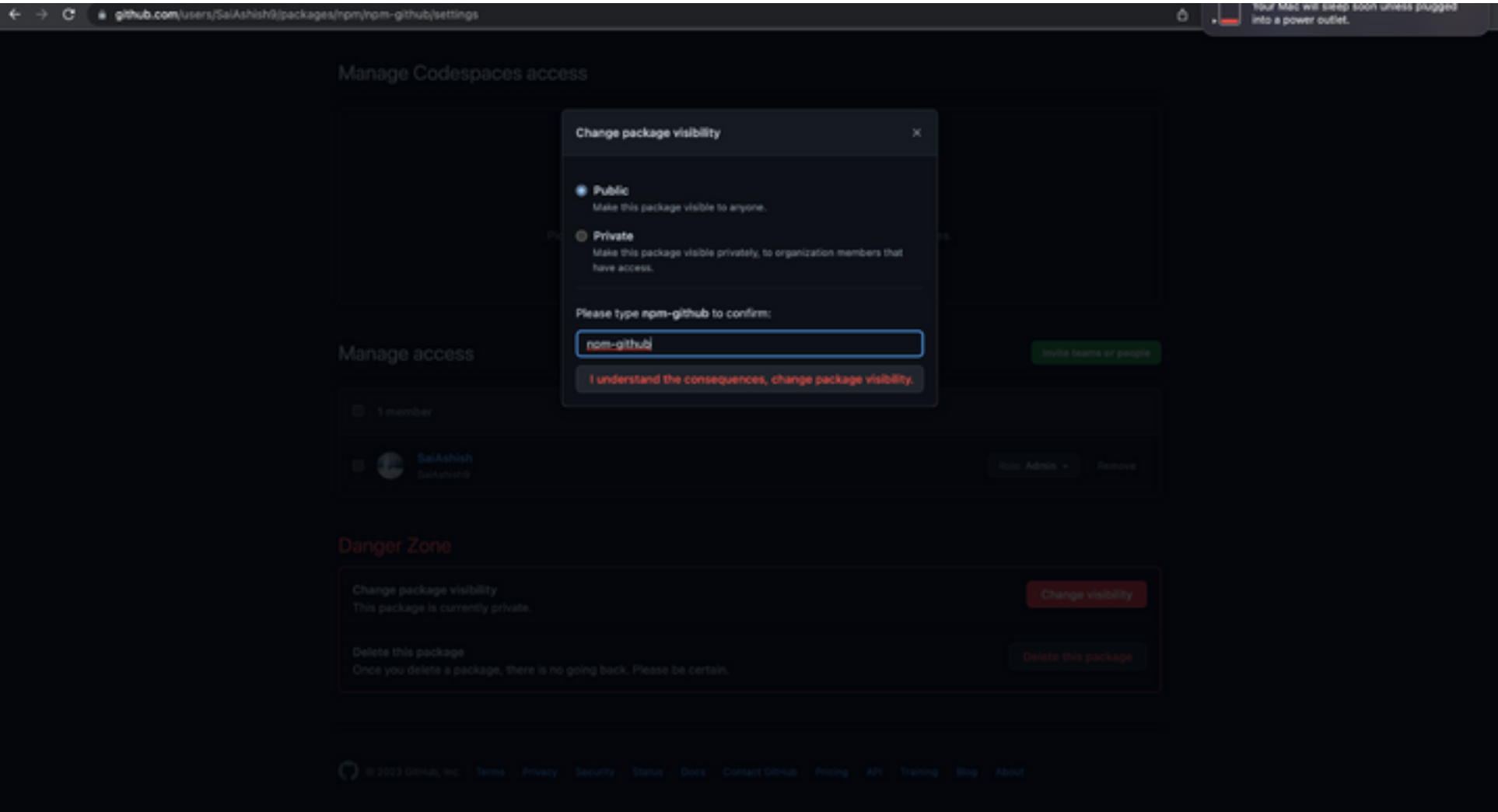


Access token can be generated at https://github.com/settings/tokens

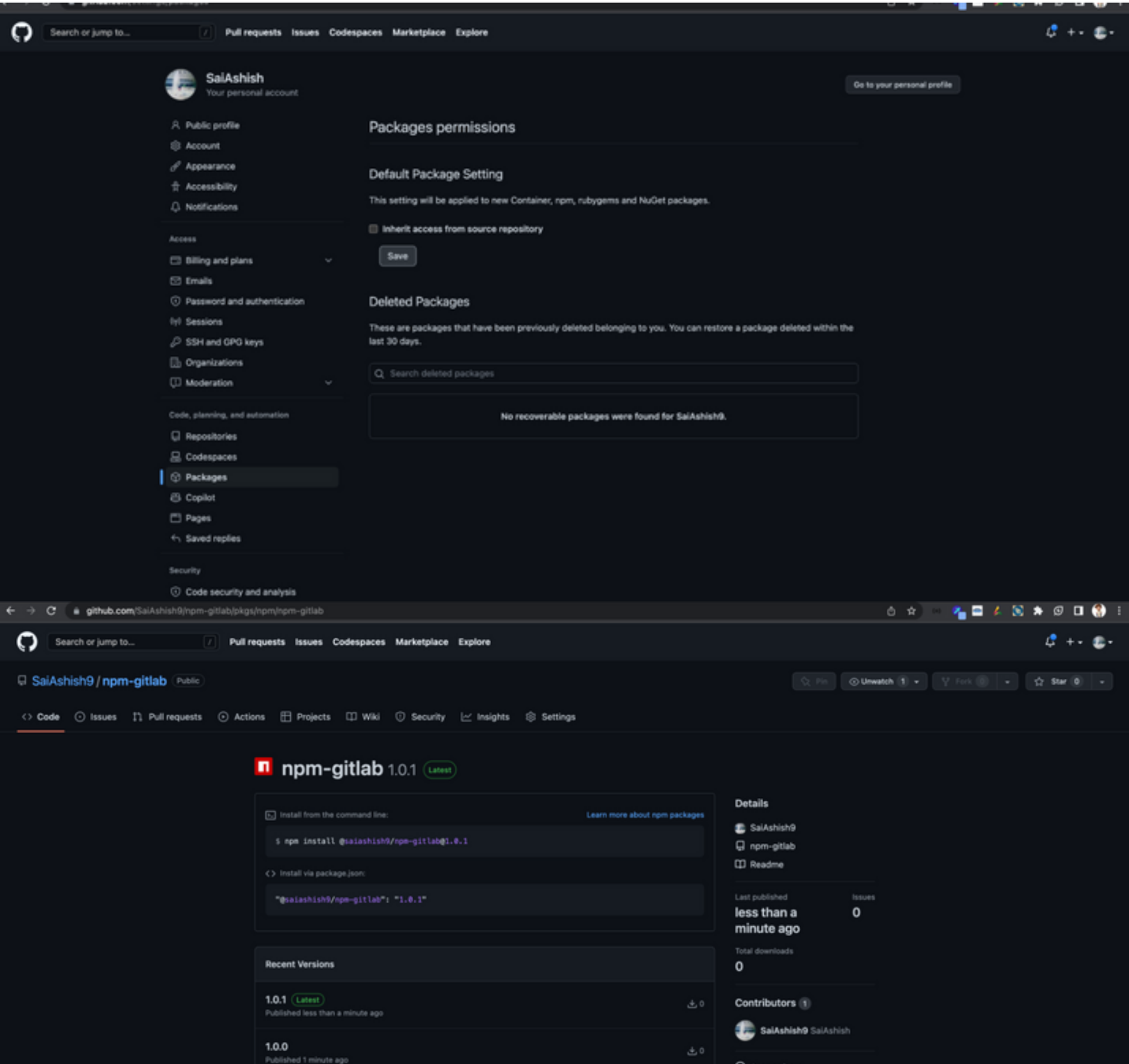




Modify package access from private to public:



Disable Inherit access from source repository at settings screen:



No description, website, or topics provided.

Readme

0 stars

1 watching

0 forks

Releases

No releases published

Create a new release

Packages 1

npm-gitlab

Languages

JavaScript 100.0%

Commands:

```
npm login --scope=@NAMESPACE --auth-type=legacy --registry=https://npm.pkg.github.com

username
access_token
```

```
npm publish --registry=https://npm.pkg.github.com/
```

index.js

```
1   console.log("hi");
2   💡
3   require("@saiashish9/npm-gitlab")
```

npm — -zsh — 80×24

```
npm notice version:       1.0.5
npm notice filename:      @saiashish9/npm-github-1.0.5.tgz
npm notice package size:  405 B
npm notice unpacked size: 552 B
npm notice shasum:        f5b6906a725e0bd5ba225adbf92e90c09005e67c
npm notice integrity:     sha512-NrJU80DZQdPyZ[...]3vkZ8F3krfQAA==
npm notice total files:   2
npm notice
npm notice Publishing to https://npm.pkg.github.com/
+ @saiashish9/npm-github@1.0.5
saiashish@MBP-C02F5ASXMD6M npm % npm install @saiashish9/npm-gitlab@1.0.2

added 1 package, and audited 2 packages in 1s

found 0 vulnerabilities
saiashish@MBP-C02F5ASXMD6M npm % npm install @saiashish9/npm-gitlab@1.0.2

up to date, audited 2 packages in 1s

found 0 vulnerabilities
saiashish@MBP-C02F5ASXMD6M npm % node index.js
hi
hi
saiashish@MBP-C02F5ASXMD6M npm %
```

Code:

.npmrc

```
@saiashish9:registry=https://npm.pkg.github.com
```

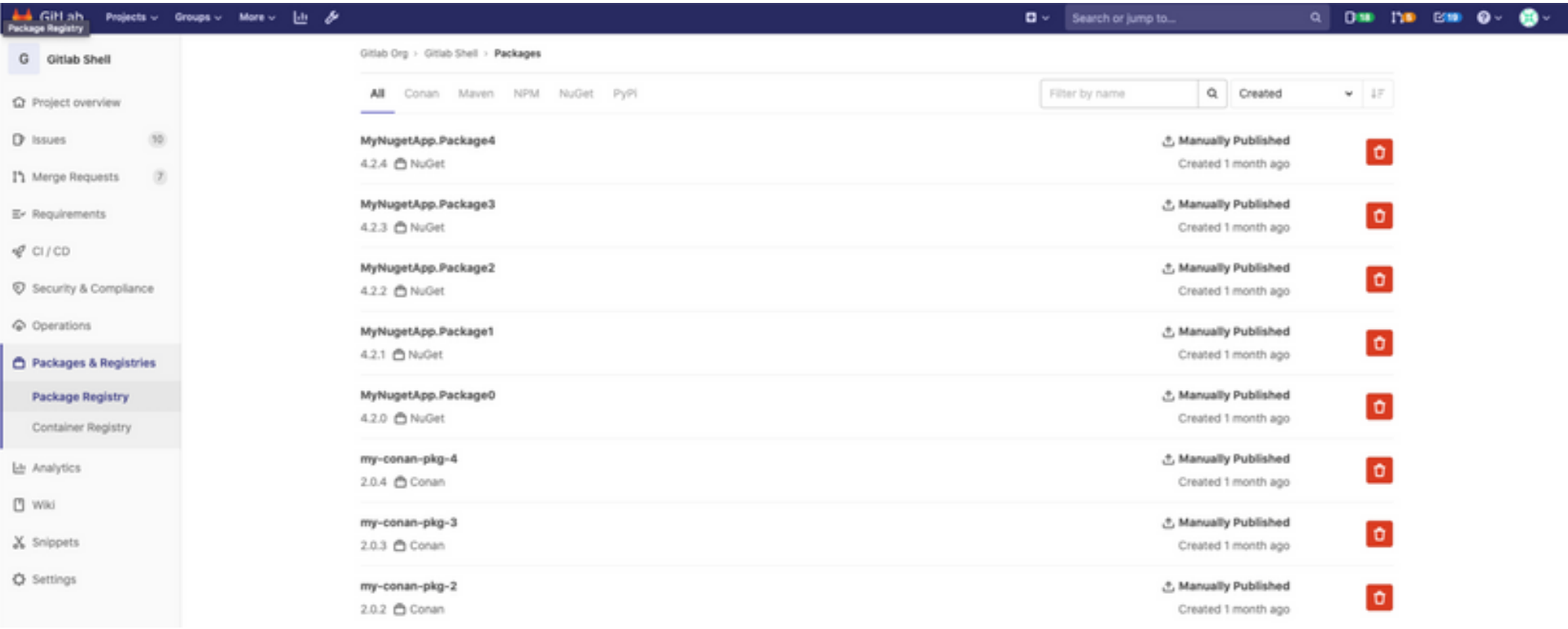
index.js

```
console.log("hi");
```

package.json

```
{
  "name": "@saiashish9/npm-github",
  "version": "1.0.5",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "repository": {
    "type": "git",
    "url": "git+https://github.com/SaiAshish9/npm-gitlab.git"
  },
  "publishConfig": {
    "access": "public"
  },
  "bugs": {
    "url": "https://github.com/SaiAshish9/npm-gitlab/issues"
  },
  "homepage": "https://github.com/SaiAshish9/npm-gitlab#readme",
  "description": ""
}
```

Gitlab Package Registries Screen Sample UI:

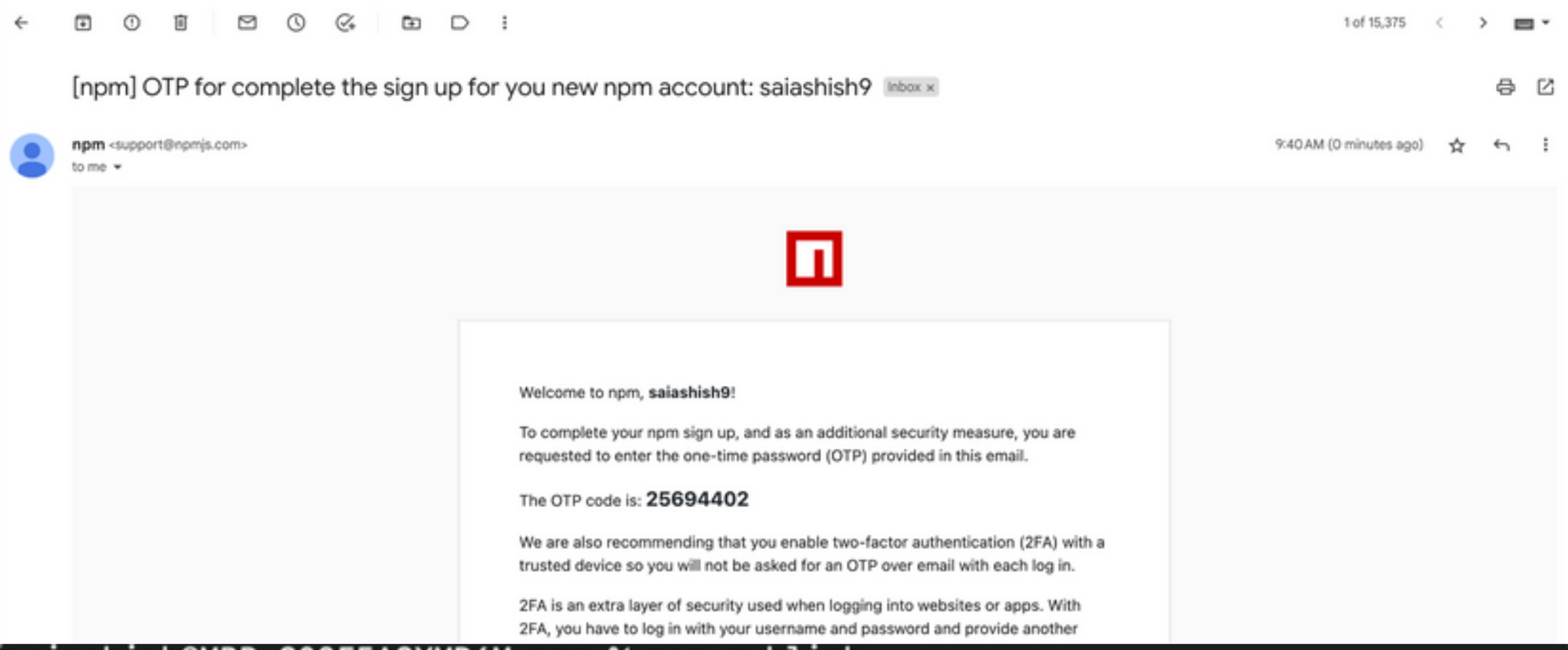


Ref. link : https://docs.gitlab.com/ee/user/packages/npm_registry/

Npm Registry & RunKit

```
npm adduser
npm WARN adduser `adduser` will be split into `login` and `register` in a future version.
`adduser` will become an alias of `register`.
`login` (currently an alias) will become its own command.
npm notice Log in on https://registry.npmjs.org/
Username: saiashish9
Password:
Email: (this IS public) saiashish7777@gmail.com
npm notice Please check your email for a one-time password (OTP)
Enter one-time password: 25694402
Logged in as saiashish9 on https://registry.npmjs.org/.
```

Ref. link : https://docs.github.com/en/packages/working-with-a-github-packages-registry/working-with-the-npm-registry




```
saiashish@MBP-C02F5ASXMD6M npm % npm publish
npm notice
npm notice 📦  npm-test-sai@1.0.0
npm notice === Tarball Contents ===
npm notice 128B index.js
npm notice 226B package.json
npm notice === Tarball Details ===
npm notice name:          npm-test-sai
npm notice version:       1.0.0
npm notice filename:      npm-test-sai-1.0.0.tgz
npm notice package size:  350 B
npm notice unpacked size: 354 B
npm notice shasum:        18e8d0d905082d91bb6ad0615fc786980a968aad
npm notice integrity:     sha512-Oz2gIqgADcf2M[...]Vc/3kws3QjCGQ==
npm notice total files:   2
npm notice
npm notice Publishing to https://registry.npmjs.org/
+ npm-test-sai@1.0.0
saiashish@MBP-C02F5ASXMD6M npm % npm update patch

up to date, audited 1 package in 125ms

found 0 vulnerabilities
saiashish@MBP-C02F5ASXMD6M npm %
```

https://www.npmjs.com/~saiashish9

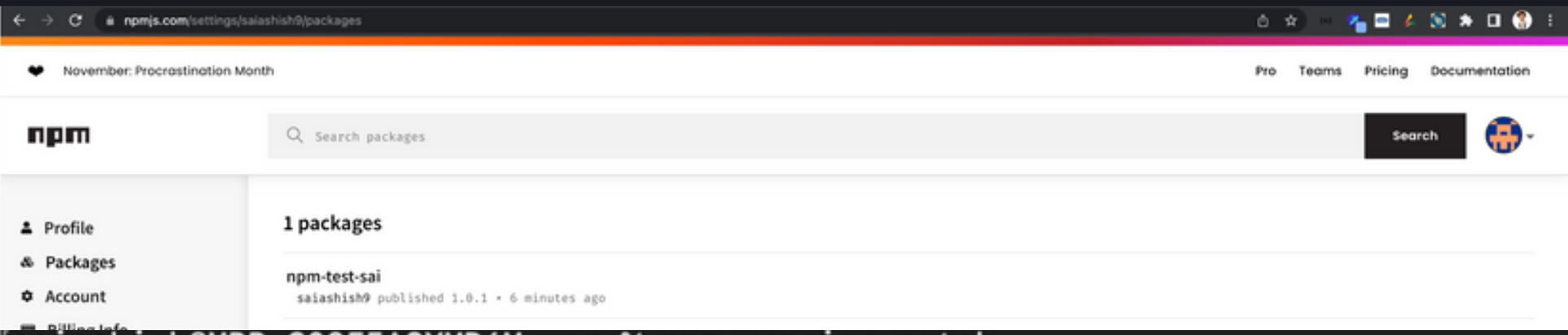
npm publish

npm version patch

It will increment the last version.

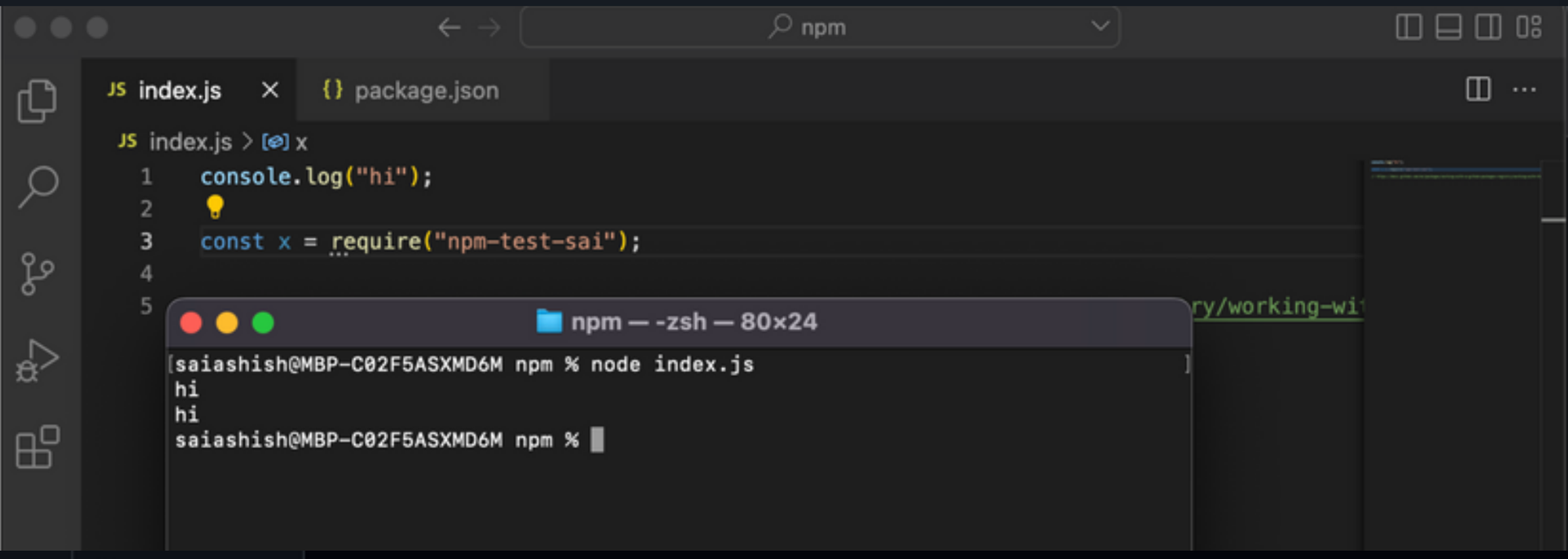
npm install

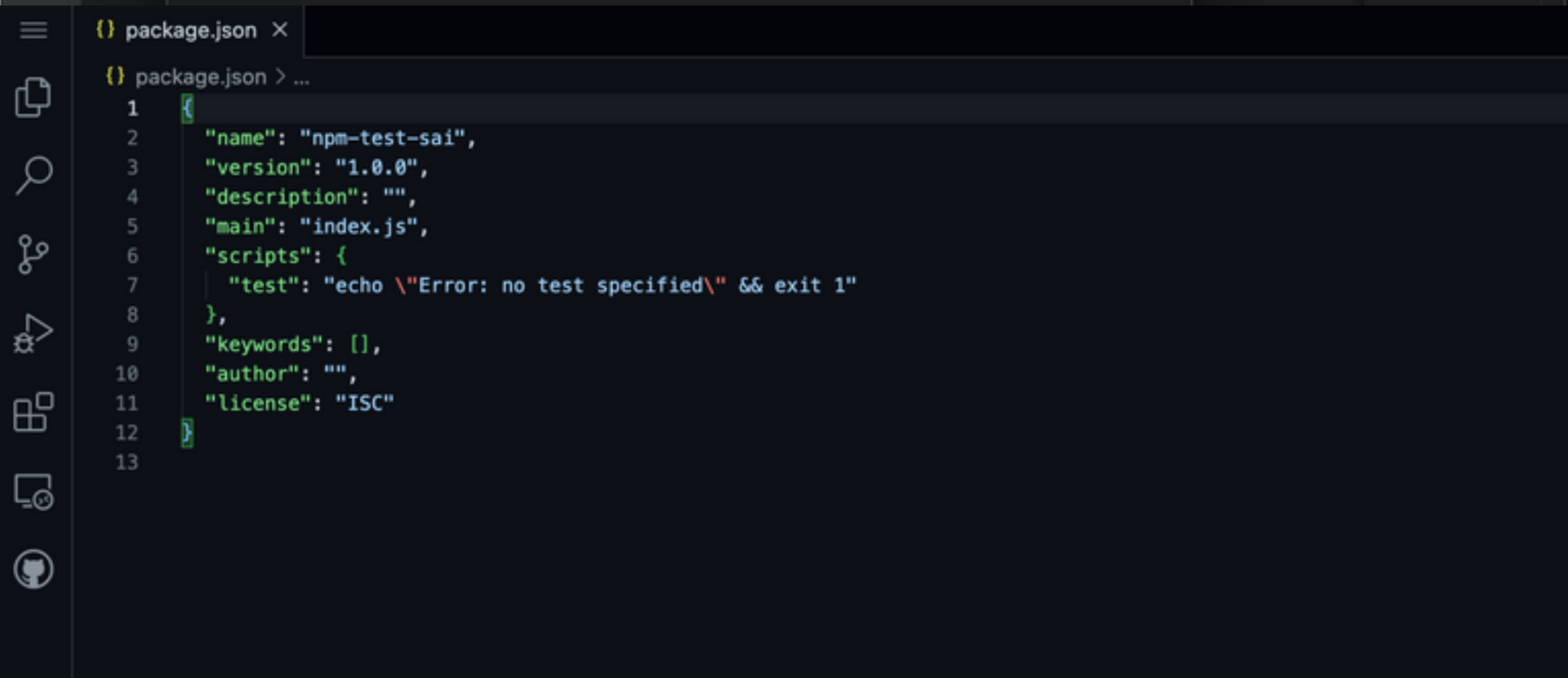
node index.js

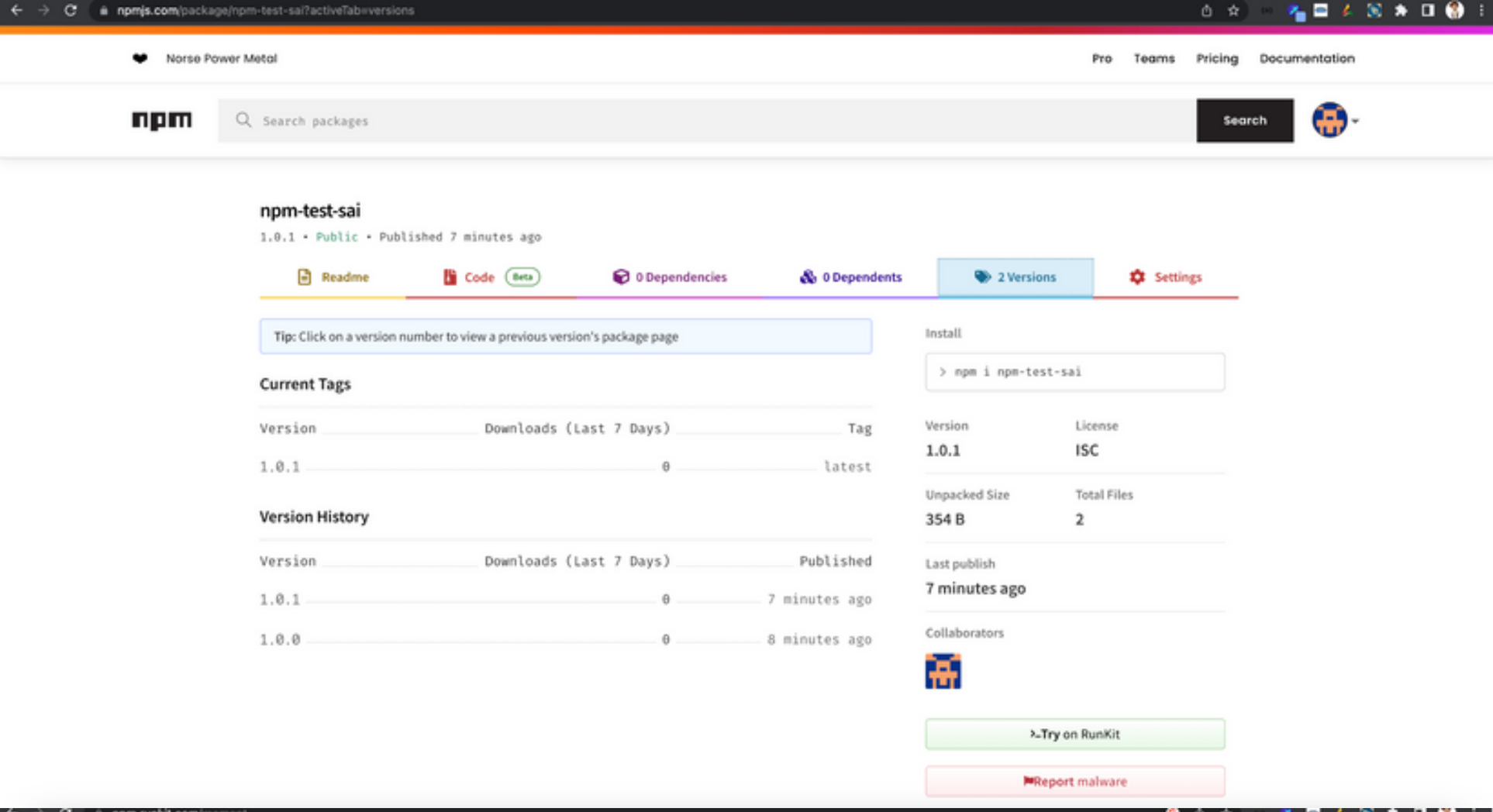


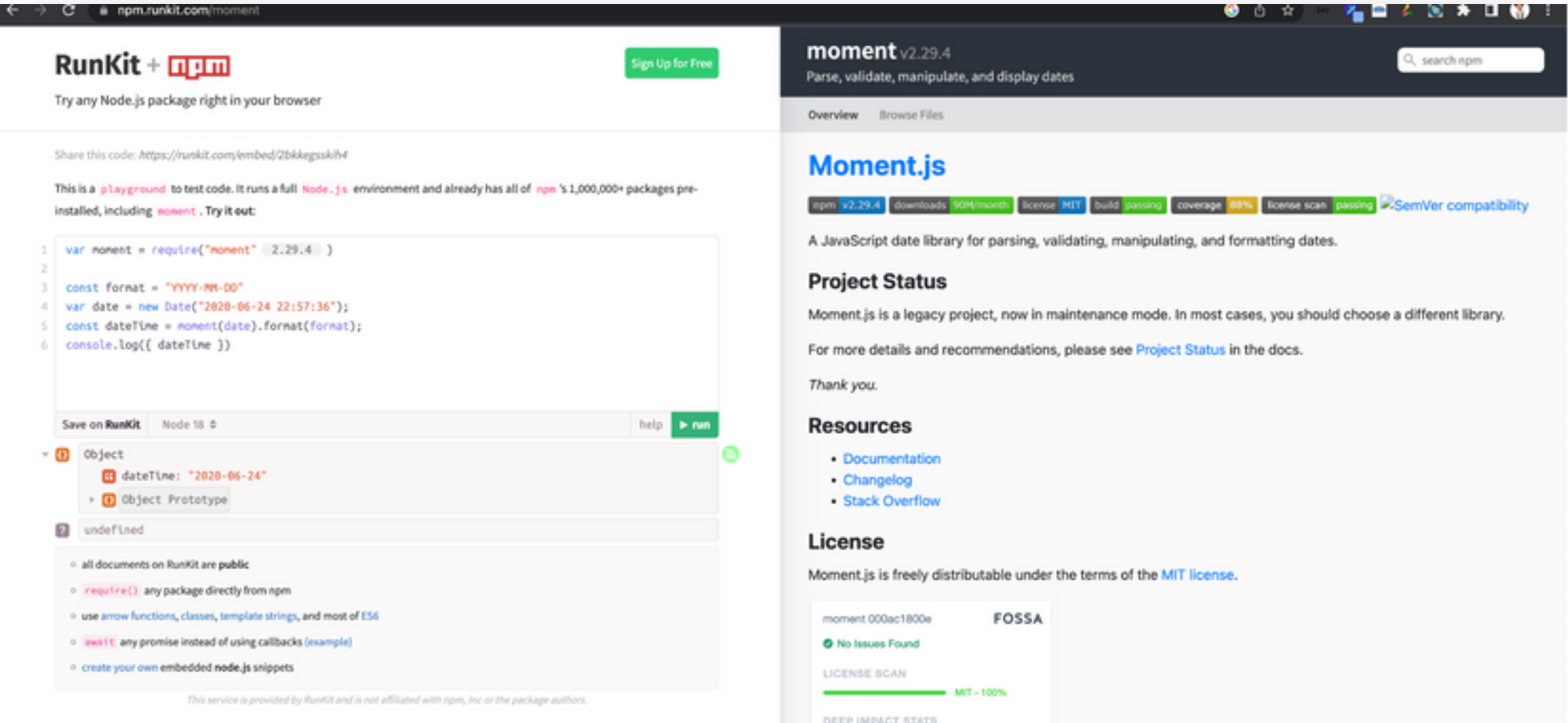

```
saiashish@MBP-C02F5ASXMD6M npm % npm version patch
v1.0.1
saiashish@MBP-C02F5ASXMD6M npm % npm publish
npm notice
npm notice 📦  npm-test-sai@1.0.1
npm notice === Tarball Contents ===
npm notice 128B index.js
npm notice 226B package.json
npm notice === Tarball Details ===
npm notice name:          npm-test-sai
npm notice version:       1.0.1
npm notice filename:      npm-test-sai-1.0.1.tgz
npm notice package size:  350 B
npm notice unpacked size: 354 B
npm notice shasum:        afe2846c8041ddd8138f92fe7bbdfda4bb54abc1
npm notice integrity:     sha512-pNT2rOtlNTxl/[...]M3OjkTL+SBHIQ==
npm notice total files:   2
npm notice
npm notice Publishing to https://registry.npmjs.org/
+ npm-test-sai@1.0.1
```

```
npm i npm-test-sai

added 1 package, and audited 2 packages in 2s

found 0 vulnerabilities

node index.js
hi
```

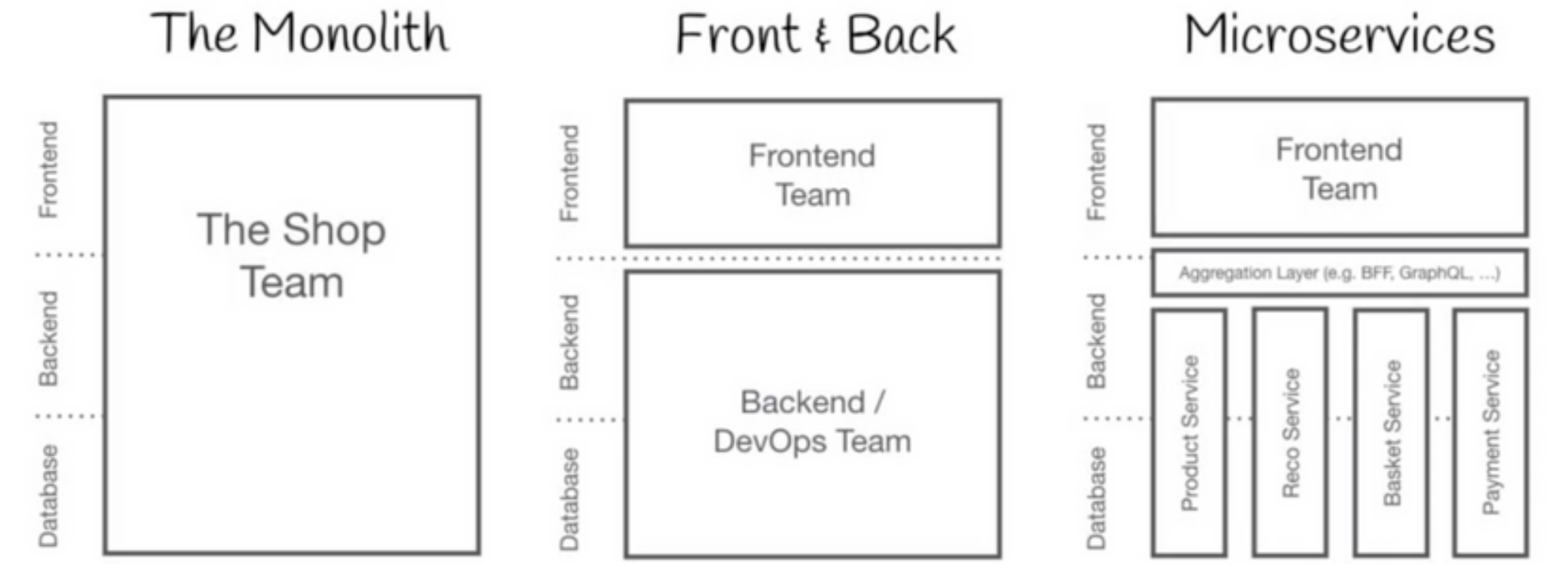




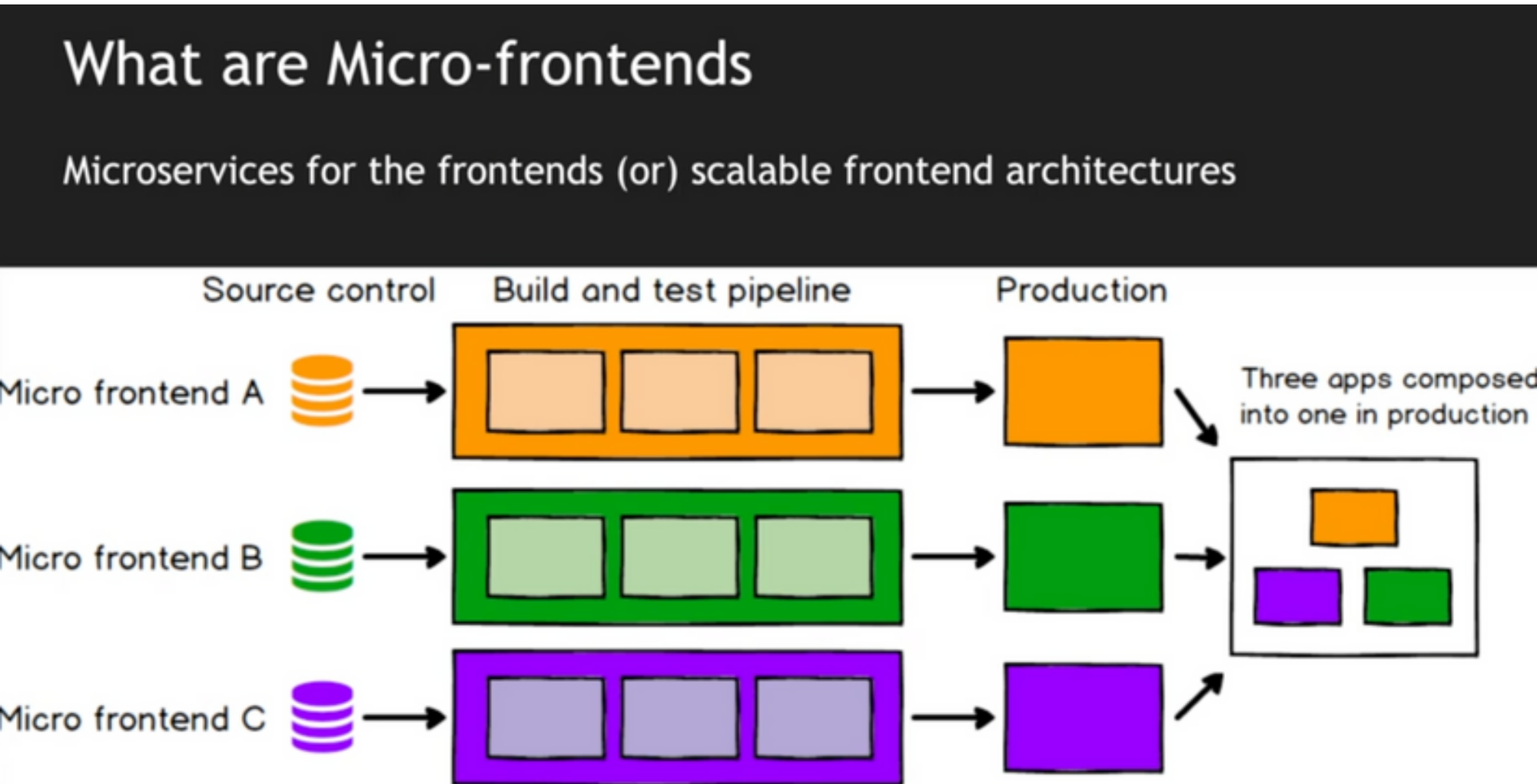

npmjs.com/package/npm-test-sai?activeTab=versions
Norse Power Metal
Pro
Teams
Pricing
Documentation
npm
Search packages
Search
npm-test-sai
1.0.1 • Public • Published 7 minutes ago
Readme
Code
Beta
0 Dependencies
0 Dependents
2 Versions
Settings
Tip: Click on a version number to view a previous version's package page
Current Tags
Version
Downloads (Last 7 Days)
Tag
1.0.1
0
latest
Version History
Version
Downloads (Last 7 Days)
Published
1.0.1
0
7 minutes ago
1.0.0
0
8 minutes ago
Install
> npm i npm-test-sai
Version
1.0.1
License
ISC
Unpacked Size
354 B
Total Files
2
Last publish
7 minutes ago
Collaborators
>_Try on RunKit
Report malware


npm.runkit.com/moment
RunKit + npm
Sign Up for Free
Try any Node.js package right in your browser
Share this code: https://runkit.com/embed/2bkkegsskih4
This is a playground to test code. It runs a full Node.js environment and already has all of npm's 1,000,000+ packages pre-installed, including moment. Try it out:
var moment = require("moment" 2.29.4)
const format = "YYYY-MM-DD"
var date = new Date("2020-06-24 22:57:36");
const dateTime = moment(date).format(format);
console.log({ dateTime })
Save on RunKit
Node 18
help
run
Object
dateTime: "2020-06-24"
Object Prototype
undefined
all documents on RunKit are public
require() any package directly from npm
use arrow functions, classes, template strings, and most of ES6
await any promise instead of using callbacks (example)
create your own embedded node.js snippets
This service is provided by RunKit and is not affiliated with npm, Inc or the package authors.
moment v2.29.4
Parse, validate, manipulate, and display dates
search npm
Overview
Browse Files
Moment.js
npm v2.29.4
downloads 90M/month
license MIT
build passing
coverage 88%
license scan passing
SemVer compatibility
A JavaScript date library for parsing, validating, manipulating, and formatting dates.
Project Status
Moment.js is a legacy project, now in maintenance mode. In most cases, you should choose a different library.
For more details and recommendations, please see Project Status in the docs.
Thank you.
Resources
Documentation
Changelog
Stack Overflow
License
Moment.js is freely distributable under the terms of the MIT license.
moment 000ac1800e
FOSSA
No Issues Found
LICENSE SCAN
MIT - 100%

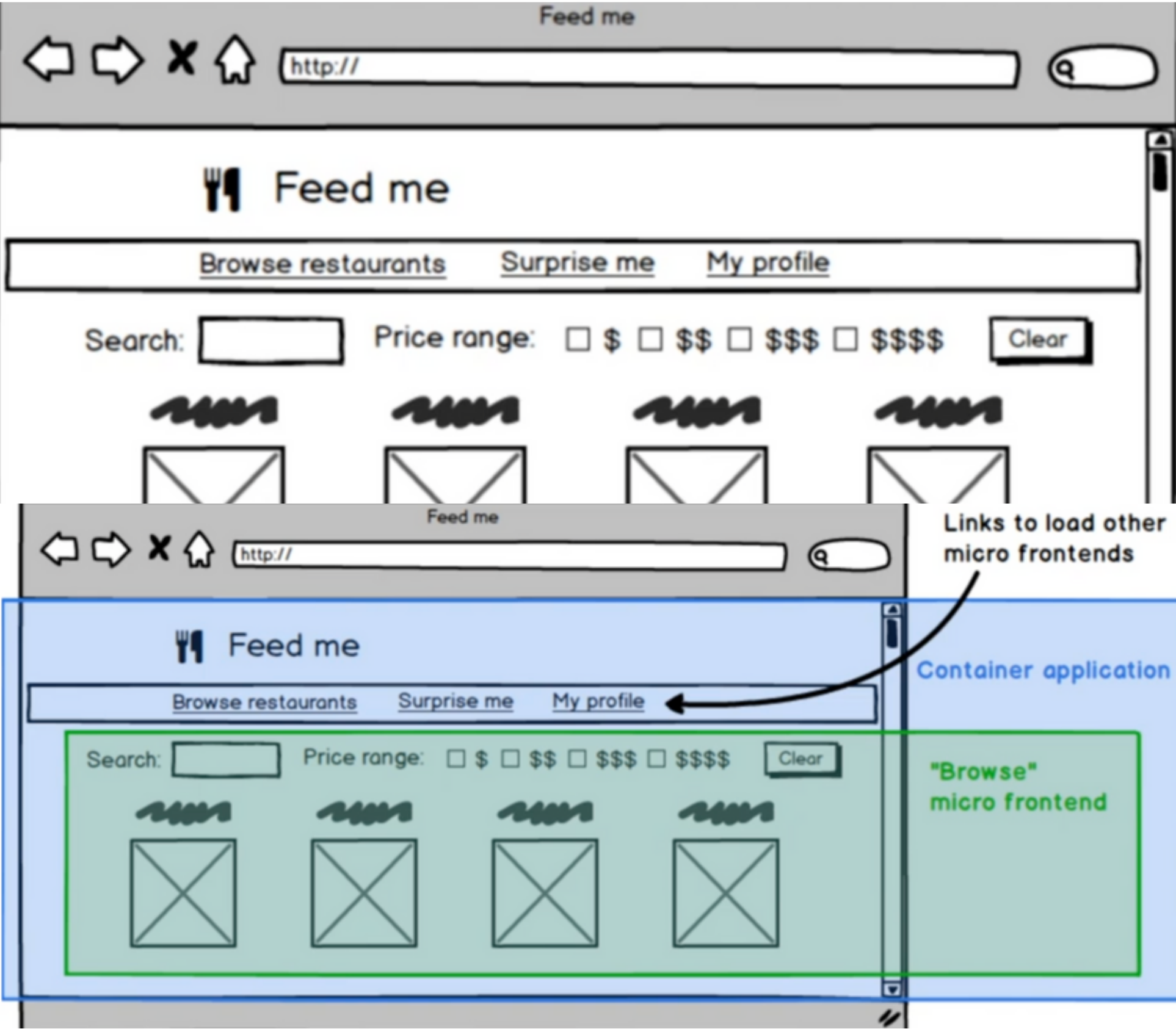
Evolution of Monolithic Frontends:



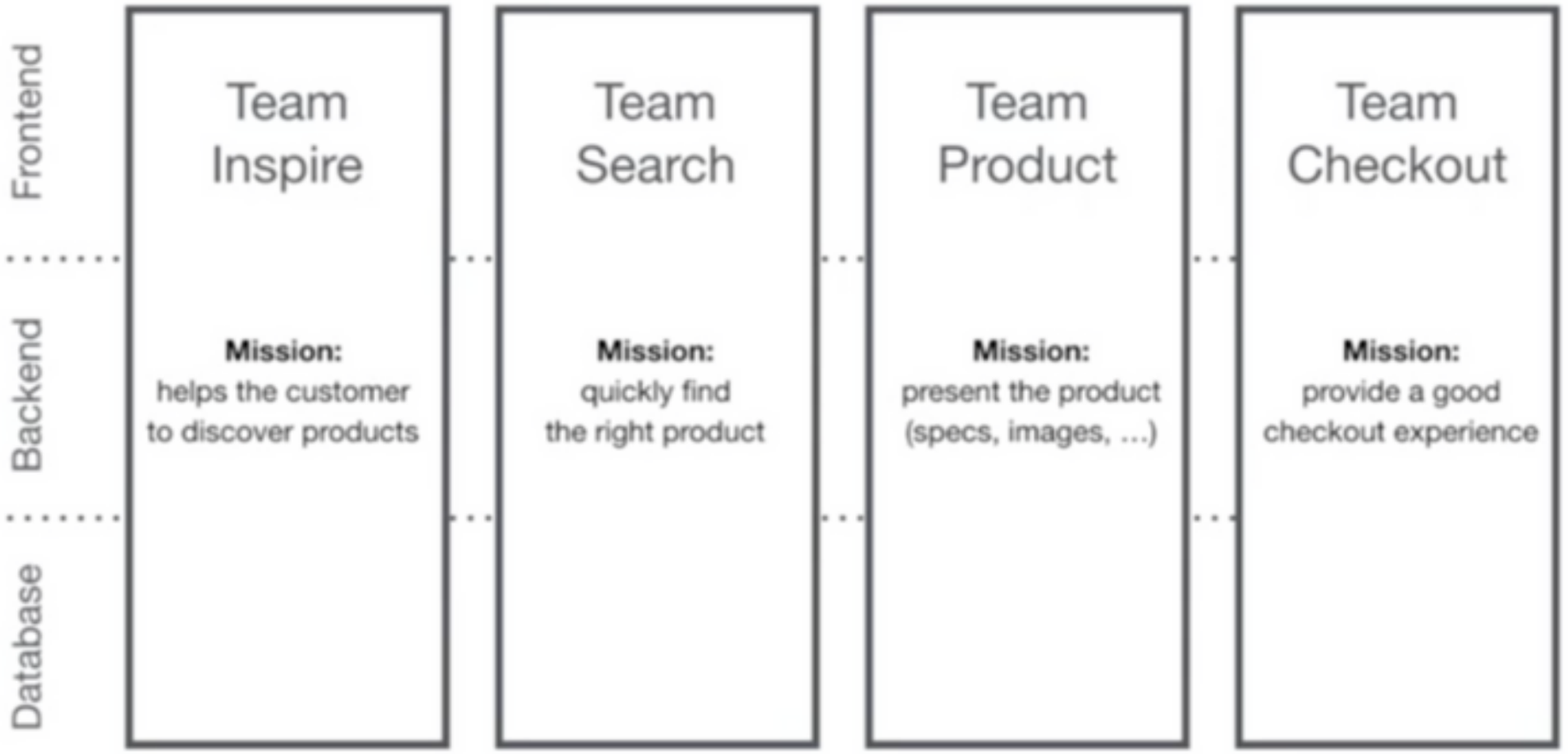
Microservices are more specific to build the scalable backend architecture

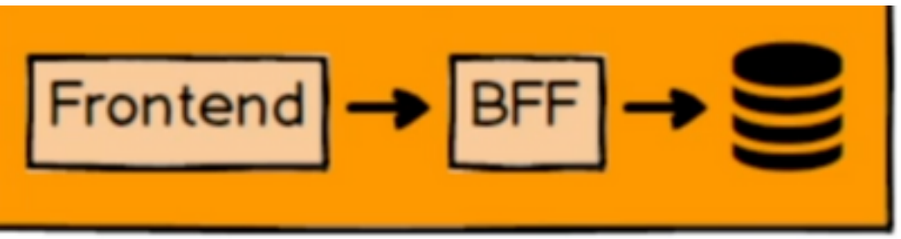


Browse restaurant, surprise me and my profile are basically three micro frontends here. The container application is actually the root or the feed-me page where the links to the micro frontends are present.



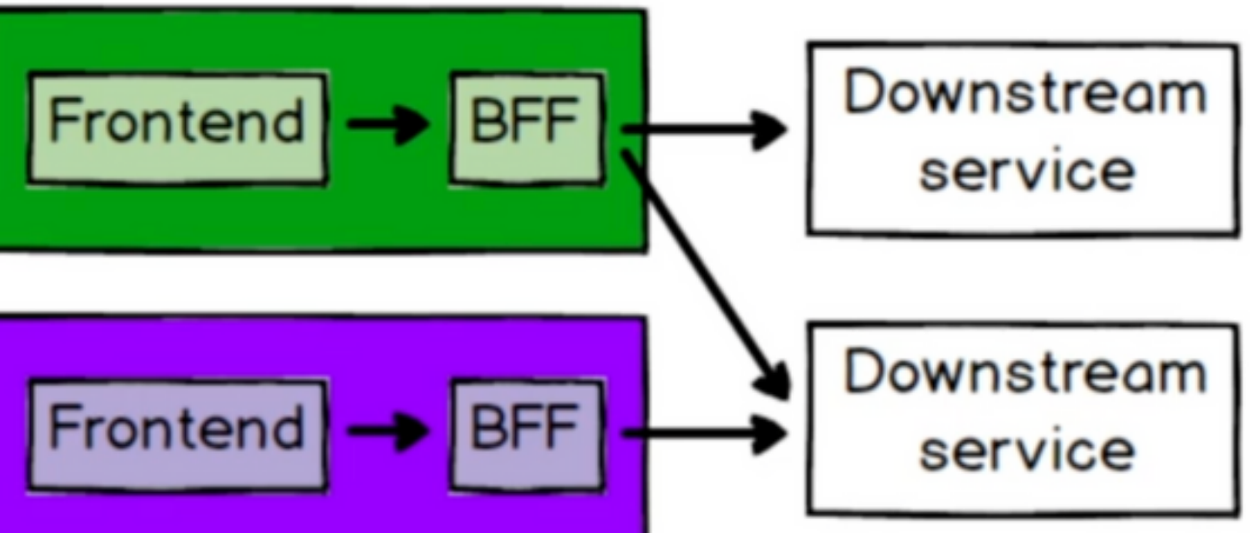
End-to-End Teams with Micro Frontends





BFFs might own their own database...

Backend For Frontend (BFF) Pattern



... or they might call through to shared downstream services

An upstream system is any system that sends data to the Product Master Server system. A downstream system is a system that receives data from the Product Master Server system. You can load data into the Product Master Server system at regular intervals (weekly, daily, or hourly) from an upstream system.

Mono repo

In revision control systems, a mono-repo is a software development strategy where code for many projects are stored in the same repository.

Individual projects have separate life-cycles as opposed to one in monolithic-apps.

In case of monolithic app whole project is deployed in one go but in case of monorepos it doen't happen the same way, we can have individual projects or products within the same repo with separate life cycles compared to what we had in the monolithic app, these could be individual pipelines or deployables depending on the type of individual product which we've in the same repository.

In short, we've a one repository but with multiple deployables with different ways in which we can deploy the repo because of the presence of multiple contexts in the repo and based on them deployment lifecycle changes.

Google, facebook, twitter and uber uses mono repo in same form.

Mono repo became very popular when Google declared that they are using a single monorepo for both gmail and youtube. The whole company uses a single repository for storing all their code.

Google, Meta, Microsoft, Uber, Airbnb, and Twitter all employ very large monorepos with varying strategies to scale build systems and version control software with a large volume of code and daily changes.

Most of these use organisational monorepo as they are at the organisational level where everybody uses a same repo for adding their code.

Spring boot, airbnb and swiggy use project level monorepo.

Project level monorepos are seggregated as an individual product or a specific area for which we've a monorepo where we've multiple subprojects which can be hosted in a single repository.

Spring boot is a single repository which is a project. Under vmware, we've spring boot but vmware doesn't have a single monorepo or organisational monorepo instead spring boot as a project has a separate monorepo.

Similarly, airbnb has a monorepo for their ui code and a monorepo for their backend code. These are two different repositories they've separated.

Tools/framework which can be leveraged for monorepos:

1. Nrwl (Next.js comes with a bundle with nrwl and nrwl can be used to create monorepos with next)
2. Lerna
3. Rush(Microsoft)

Microsoft created rushjs which is now called rush because they moved it into typescript.

4. Yarn Workpaces
5. Npm Workpaces
6. Google's Bazel

Case study: Mono repos at swiggy (food delivery app):

They earlier had the dweb which is the desktop application of swiggy and mweb which is the mobile application of swiggy. So they had two different repositories , so when we open the mobile we get a different version and at mobile, we'll get another version.

They'll served from different deployment and codebase.

They were all coverging into a nodejs based api proxy which is also known as a service-proxy and it acts as a component between different middleware components which are hosted behind the systems.

At a high level, we've three repositories one each for dweb, mweb and service api proxy.

They were using react from frontend applications, webpack for packaging and babel for compiling js code into final executables.

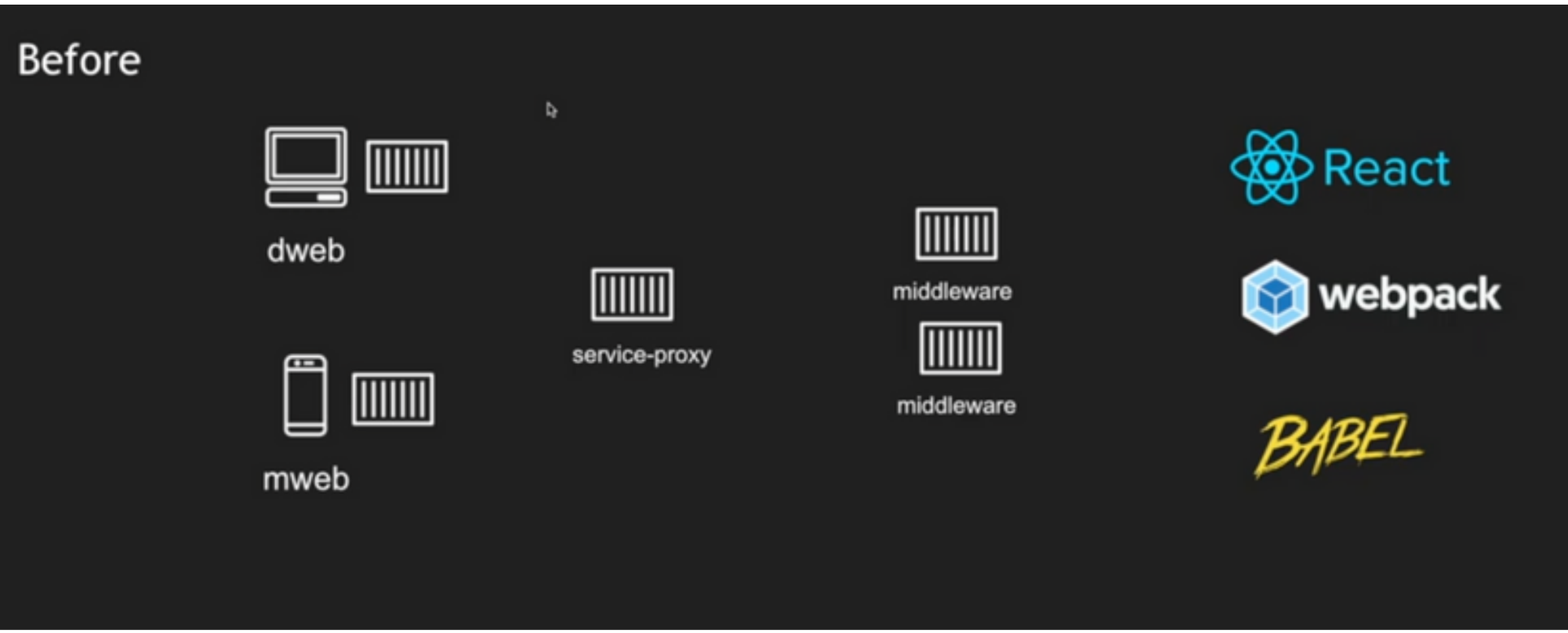
As the codebase grew swiggy recognised that they were some patterns where the code is being reused between mobile web and desktop web application.

There was reusable code which was copy pasted and if there was a bug fix , they had to do that in both dweb and the mweb.

In addition to that adding dependencies for react, webpack and babel becomes complicated as we keep on upgrading to different versions, we need to updated both the repositories but they had to do two diff. deployments separately, review them and test them separately.

It was tidious for the team to add all the common code and setup in a library as it was huge and there are multiple teams which were working on these repositories.

Swiggy ui architecture before moving to monorepo :



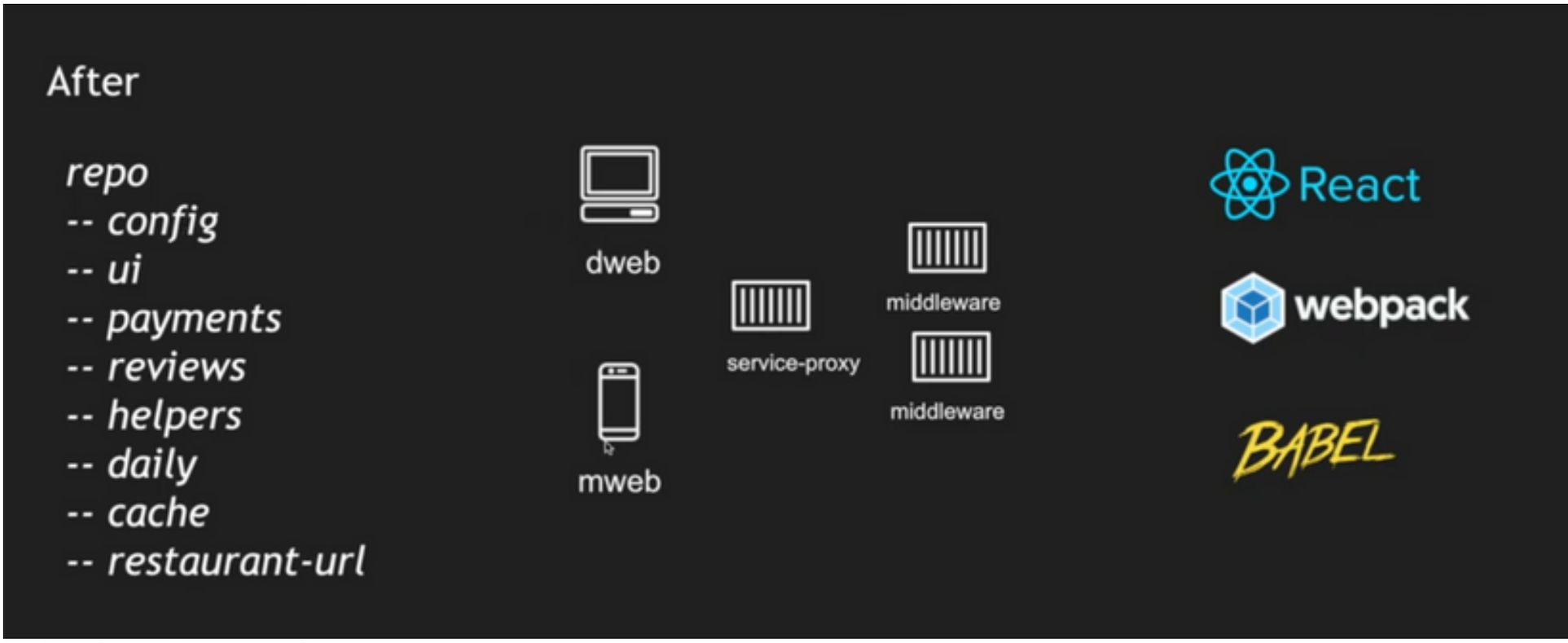
After they moved into monorepos, they changed the whole structure of the repository. So there was only one repository where they had the dweb, mweb and the service proxy. Config inside repo package had the webpack configuration which is reused by both the mobile and the web application.

Hence, if we update one particular webpack under config package it will be updated to both the deployments.

They'll are in a same codebase, but they are deployed as a separate artifact.

Ref. Link : https://bytes.swiggy.com/monorepos-lets-talk-about-it-4b1b7d4f1038

Swiggy ui architecture after moving to monorepo :



Limitations of mono repo:

1. Codebase :
There are going to be challenges in terms of size since mono repo can have a single codebase. If there are too many images stored in that codebase, we need to look at the version control system which can

handle this humongous codebase.

Google uses their own version control system as with repository size of more than 10 GB, we can face issues with git.

2. Operational Complexity:

There's going to be a lot of PR's to a single repo and merge conflicts are going to be coming in as there are more teams contributing to the same repository.

3. Payload Size during runtime:

What if one particular feature has a version for a dependency and other feature has different version for other dependency. This creates longer run time because page is going to download js files we've different micro front ends serving and adding different libraries during time. This adds to the payload size and websites can be heavier in individual parts.

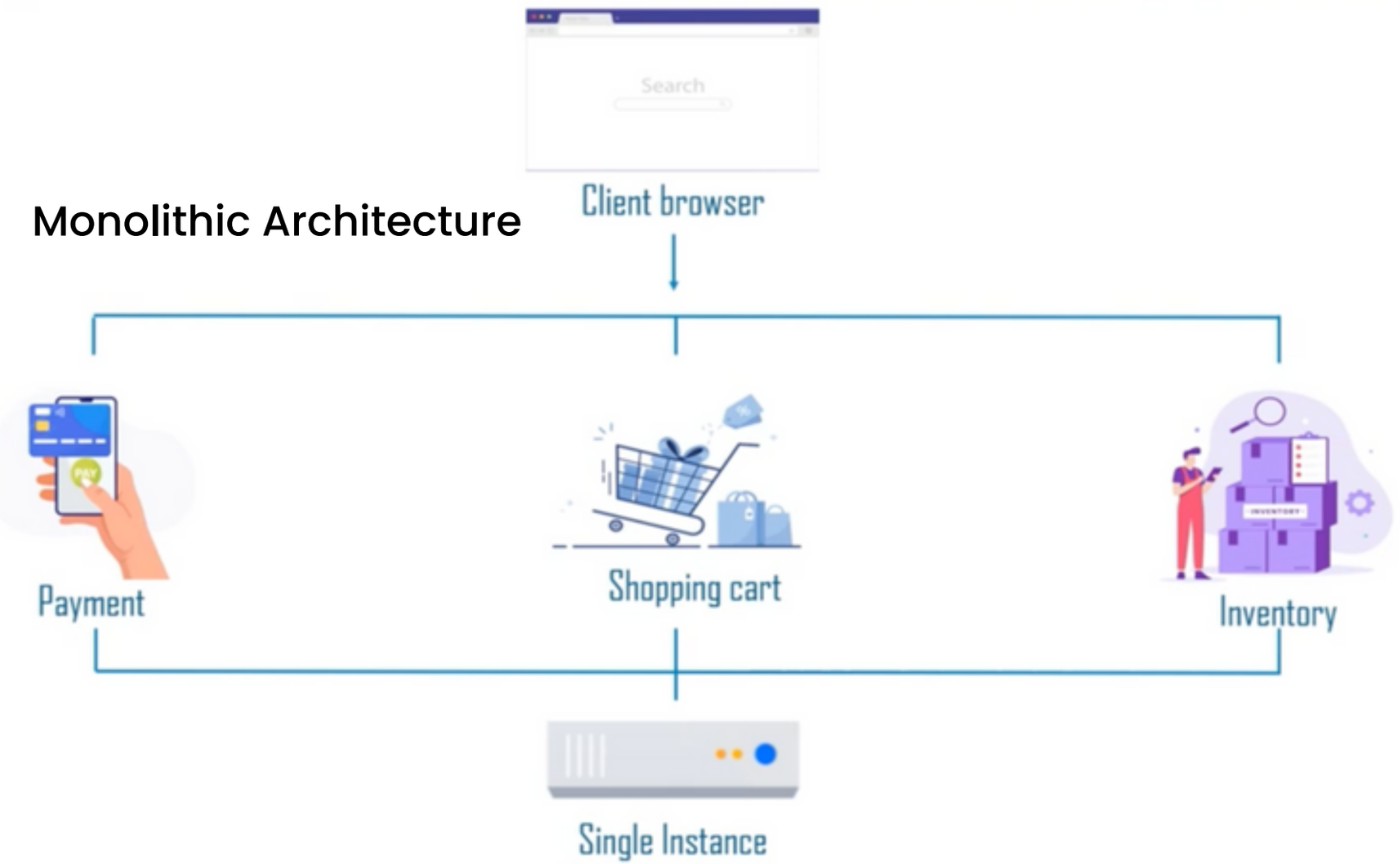
4. Environment Differences:

When we test applications locally, we might test it in a diff fashion and with micro front ends we cannot deploy the whole app in the way we run it locally. So it could be more tedious to set up micro front ends with mono repo in local compare to how we deploy to production. This may lead to production bugs which cannot be tested at local but only at production.

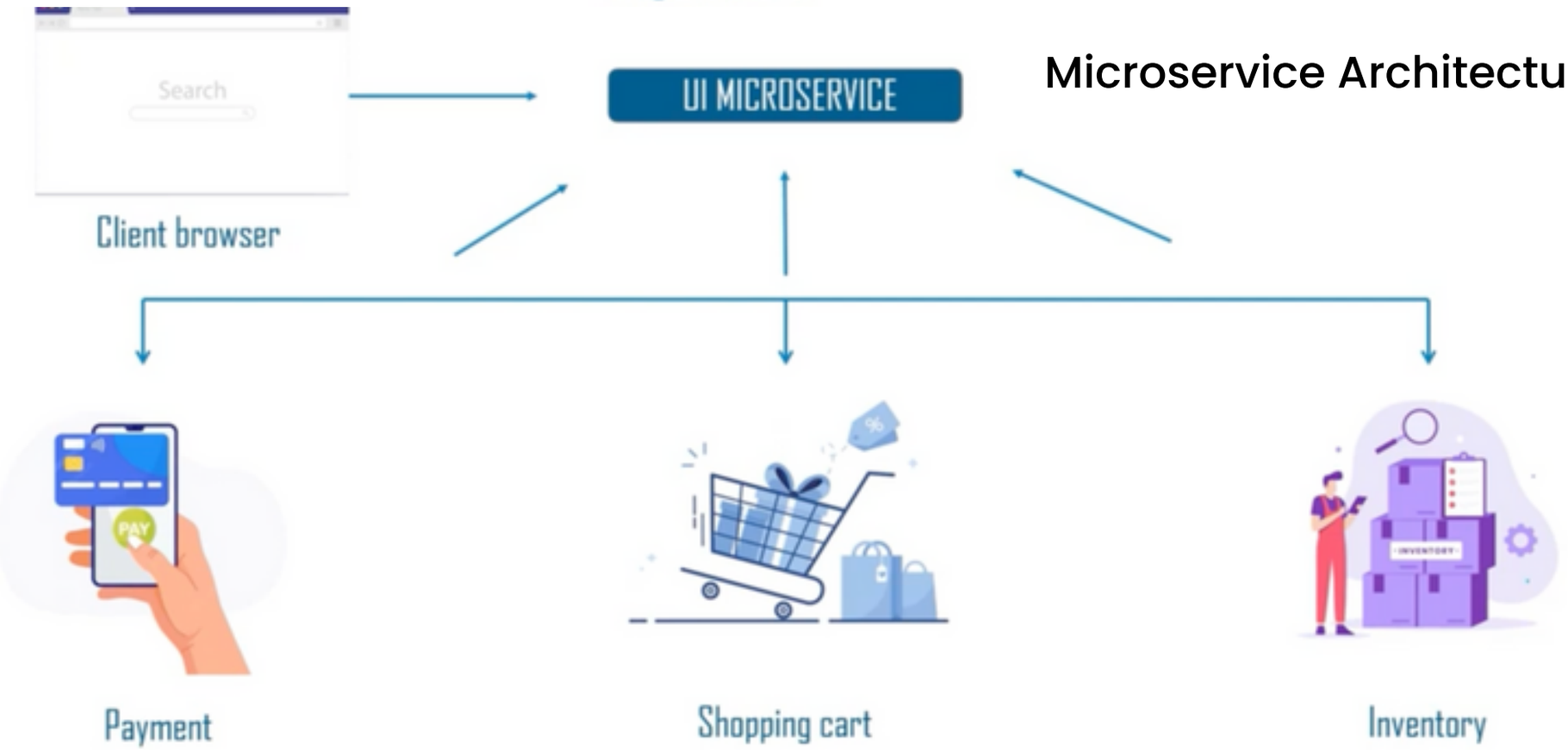
Conclusions :

1. Monorepo != Monolith : Monorepos are a way to manage multiple projects in a single repository. Microfrontends are a useful way to read benefits of monorepo as a whole
2. Monorepos is not a silver bullet. It's not going to solve all the problems which we've.
3. Identify why the current codebase doesn't scale and decide whether we need monorepos or not.

Monolithic Architecture



Microservice Architecture



Why to migrate from monolith to microservices:

1. Netflix:

In 2009, netflix faced issues with the rapidly growing video streaming services.

The company decided to update its architecture from a private data center to a public cloud. It has won multiple global awards for this migration and today it has more than 10k microservices which support different parts of the platform.

2. Amazon, ebay & Twitter

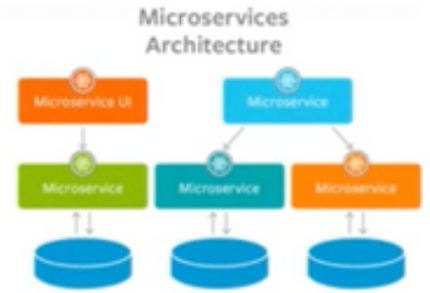
3. Atlassian:

Atlassian is an Australian company founded in 2002, and its head quater is in Sydney, Australia.
Atlassian migrated to microservices after they faced challenges with growing scale at jira and confluence.

They found that their single tenant monolithic architecture cannot be able to scale as the feature needs. They decided to rearchitecture jira and confluence and move their stateful single tenant monolithic system to multi-tenant stateless cloud application hosted by aws. They'll decompose them over time to microservices.

This project was named vertical and it was their largest infrastructure project till date and it took 2 years to move into AWS migrating over 100k customers in just 10 months with no service adoption, they also considered these services to micro services.

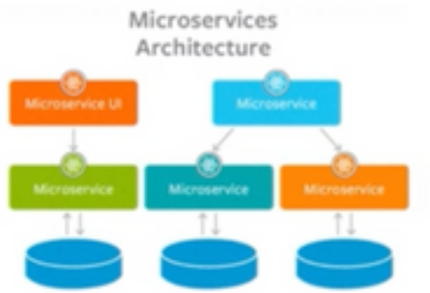
Definition:



A monolithic architecture is a traditional model of a software program, which is built as a unified unit that is self-contained and independent from other applications

A microservices architecture, also simply known as microservices, is an architectural method that relies on a series of independently deployable services

Deployment:

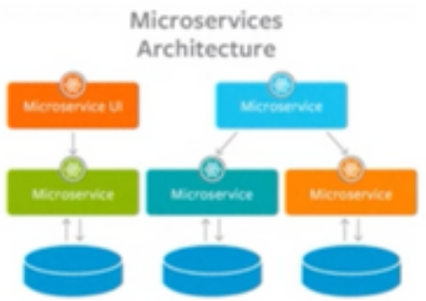


One executable file or directory makes deployment easier

Independent deployable, since microservices are individual units, they allow for fast and easy independent deployment of individual features

Scalability:

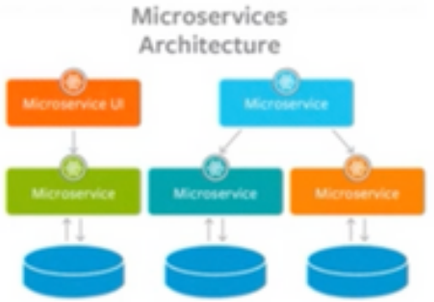
You can't scale individual components



If a microservice reaches its load capacity, new instances of that service can rapidly be deployed to the accompanying cluster to help relieve pressure

Reliability:

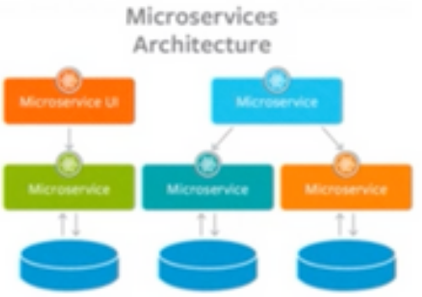
If there's an error in any module, it could affect the entire application's availability



You can deploy changes for a specific service, without the threat of bringing down the entire application

Development:

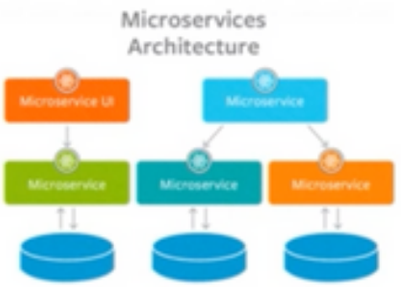
When an application is built with one code base, it is easier to develop



Microservices add more complexity compared to a monolith architecture, since there are more services in more places created by multiple teams

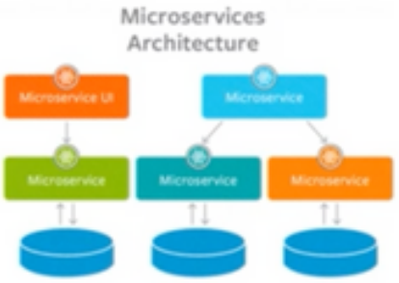
Debugging:

Easy debugging – With all code located in one place, it's easier to follow a request and find an issue



Each microservice has its own set of logs, which makes debugging more complicated

Flexibility:

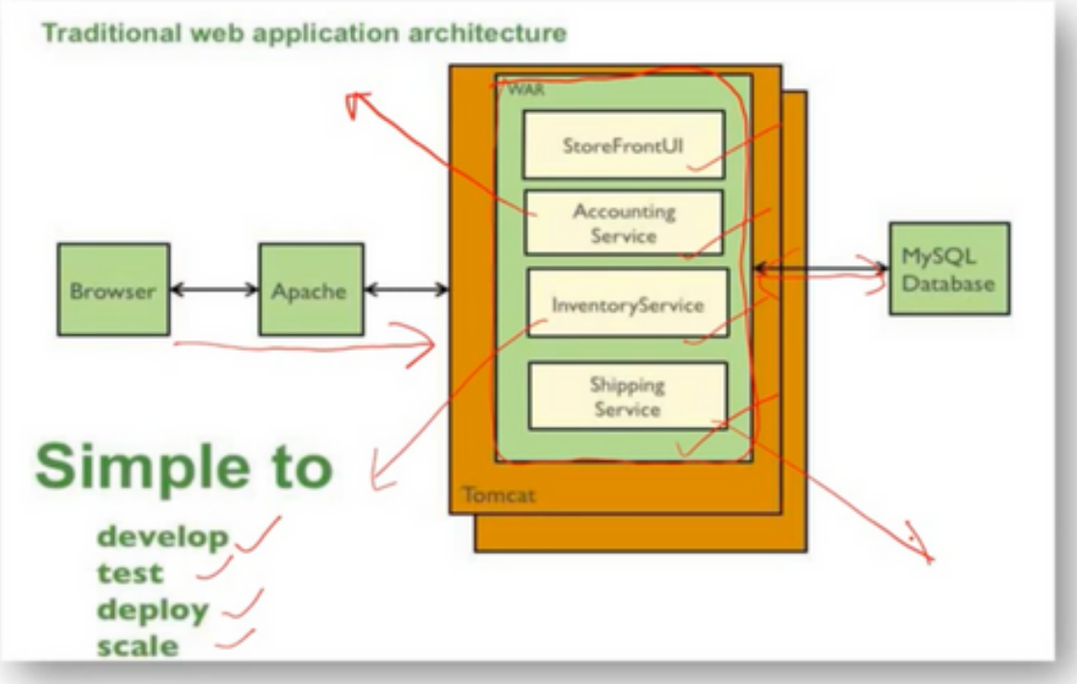


Lack of flexibility, a Monolith is constrained by the technologies already used in the monolith

Flexibility , Microservice architectures allow teams the freedom to select the tools they desire

MONOLITHIC ARCHITECTURE

- Single jar/war file for whole application
- Issues
  - Less flexible for large team and code base
  - Overload IDE
  - Continuous development is difficult
  - Scaling the app is difficult
  - Scaling development is difficult
  - Technology stack change is difficult



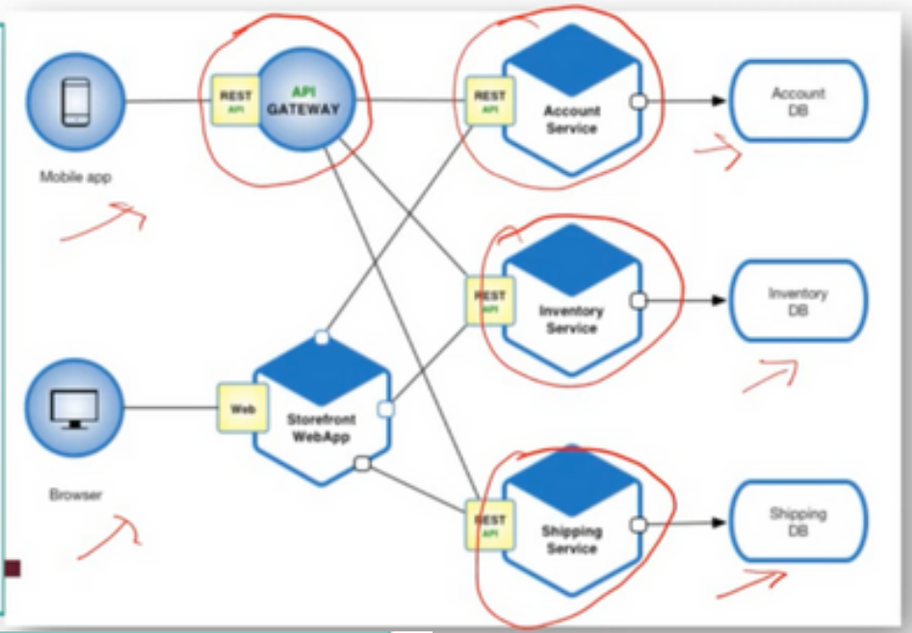
An event-driven microservices architecture is an approach to software development where decoupled microservices are designed to communicate with one another when events occur.

These architectures are comprised of three critical features:

1. Event producers, which detect and send events to routers when they occur,
2. Event routers, which analyze events and determine where to send them and
3. Event consumers, which process events, handle them, and complete the action (e.g., updating a data warehouse to reflect current inventory levels).

MICROSERVICE ARCHITECTURE

- A set of loosely coupled, collaborating services. Each service is :
- Highly maintainable and testable
- Loosely coupled with other services
- Independently deployable
- Capable of being developed by a small team
- Services can be developed independent of each other
- Communication among service via HTTP/REST/AMQP



MICROSERVICES CHARACTERISTICS

- Service granularity
- Linguistic approach
- Technologic agnostic
- Some points
  - Microservices is a specialization of an implementation approach for **service-oriented architectures** (SOA) used to build flexible, independently deployable software systems.
  - Followed the introduction of DevOps
  - Strategy - "Do one thing and do it well"