Hashnode’s journey into its public GraphQL API

From Next.js API routes to a serverless GraphQL API

Author

SANDRO VOLPICELLA

Date

2024-10-03

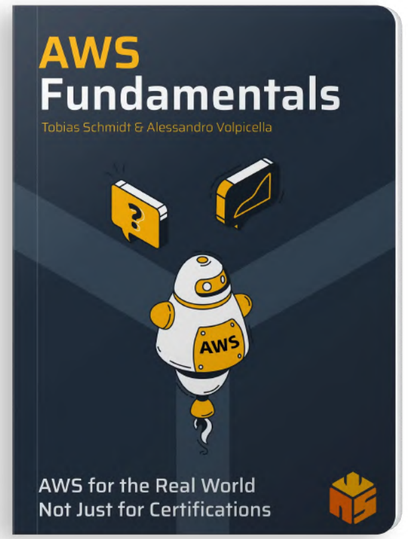
whoami

- Software Engineer (Fullstack Serverless) from Munich, Germany 🇩🇪🥨
- Platform Engineering Lead @hashnode since 2022
- AWS Community Builder 🧡
- 📕 awsfundamentals.com | 📙 cloudwatchbook.com
- Plays tennis 🎾 & lifts wights 🏋️

Author

SANDRO VOLPICELLA

@SANDRO_VOL



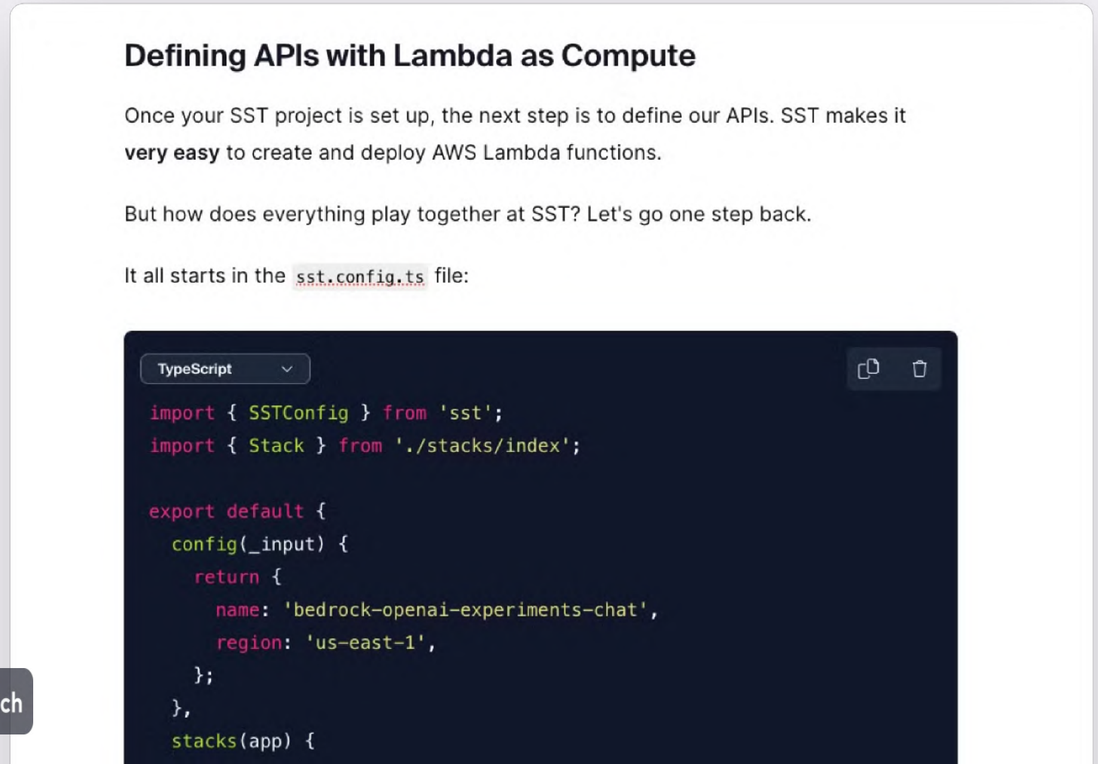
</>

What is Hashnode?

API & DEVELOPER DOCS | DEVELOPER BLOGGING PLATFORM | HEADLESS CMS

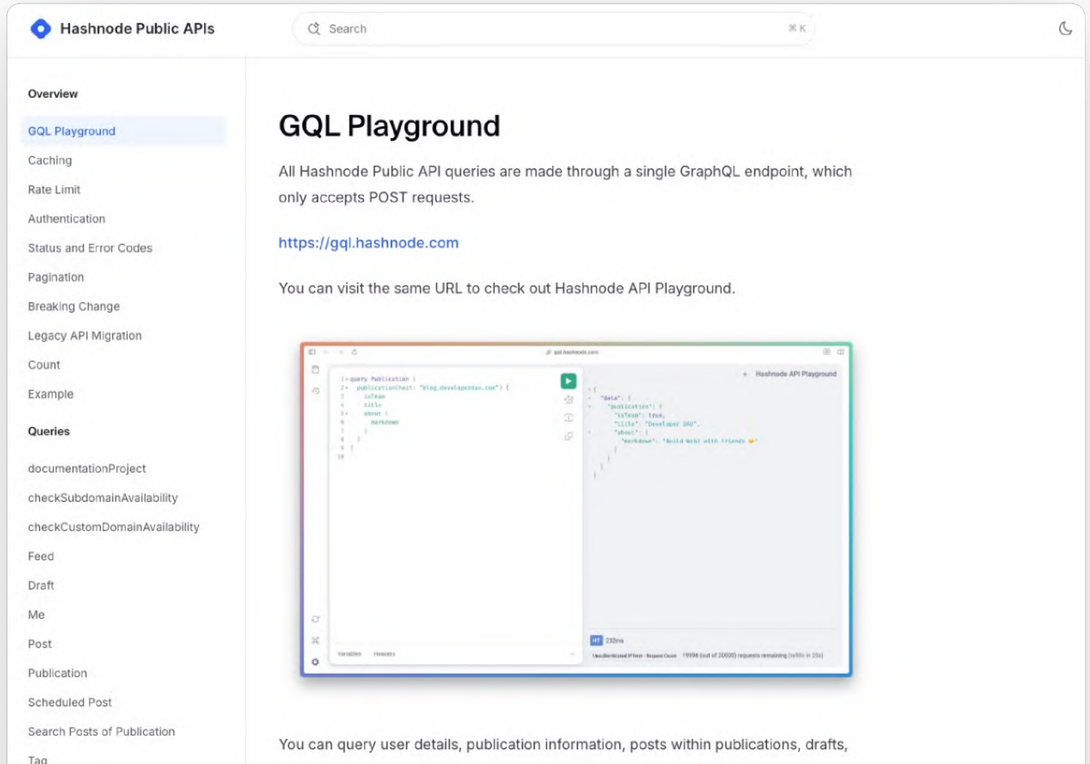
Developer Blogging

- Superb editor + AI
- Blogging on custom domain
- Headless CMS → Bring your own frontend
- Newsletters, Webhooks, GitHub Backup, RSS, and more!



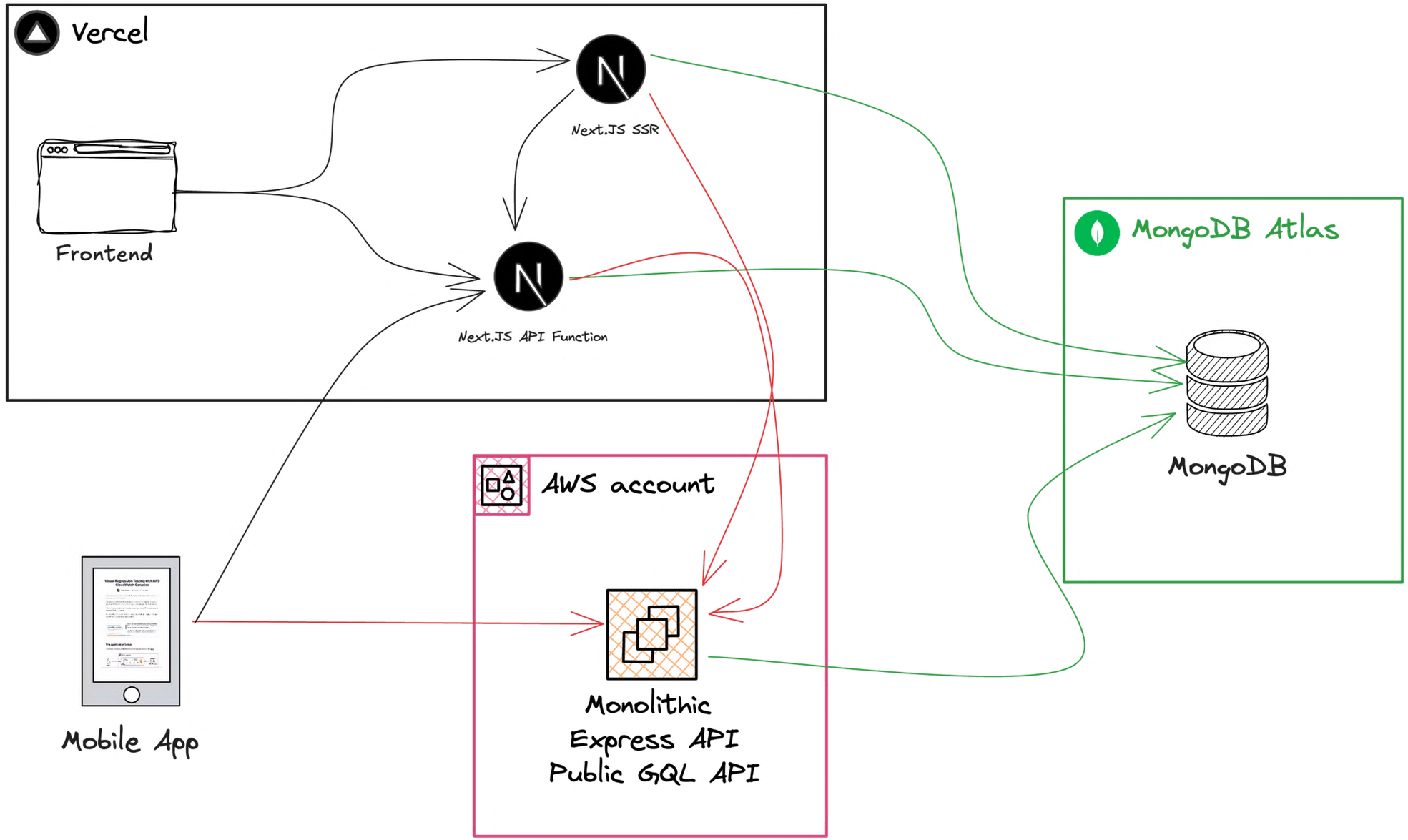
Developer Documentation

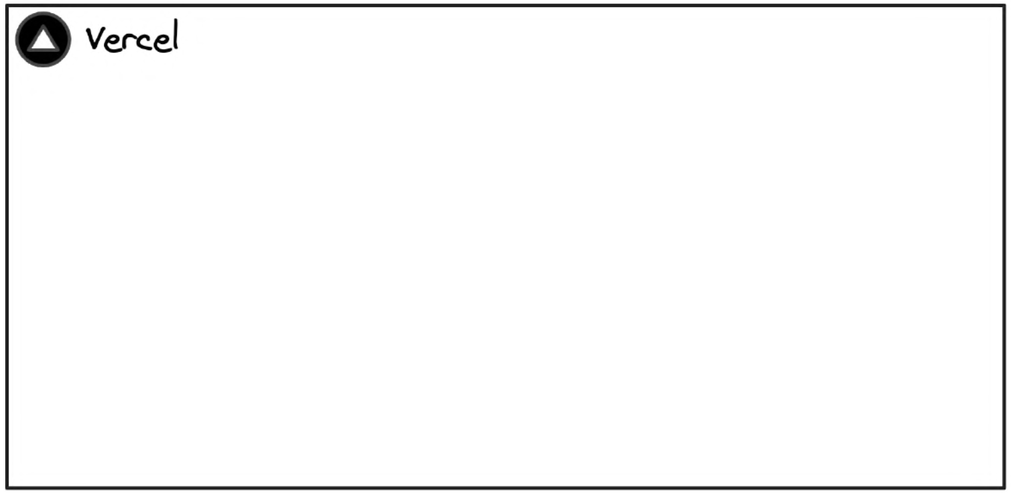
- MDX-powered Editor
- Open API/Swagger automatic API documentation
- Endless flexibility with headless CMS + GQL
- Fulltext + AI Search

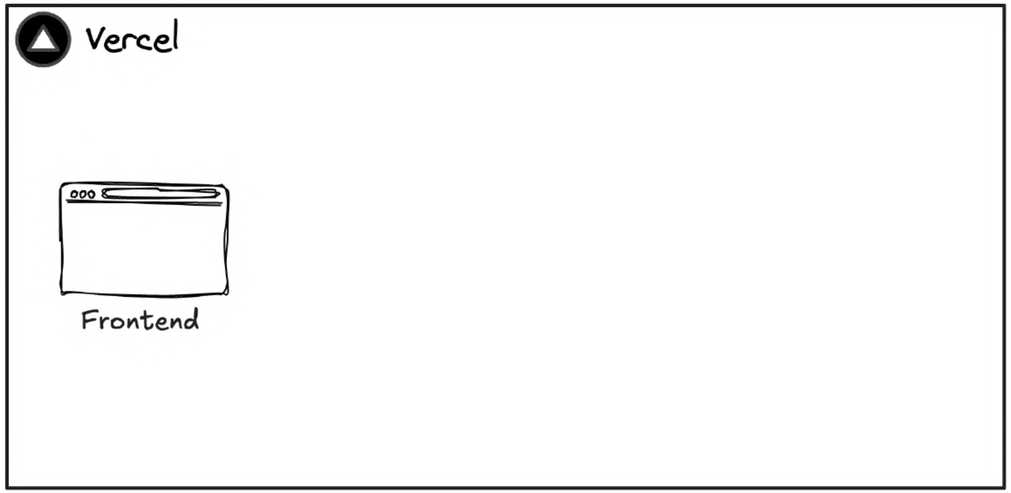


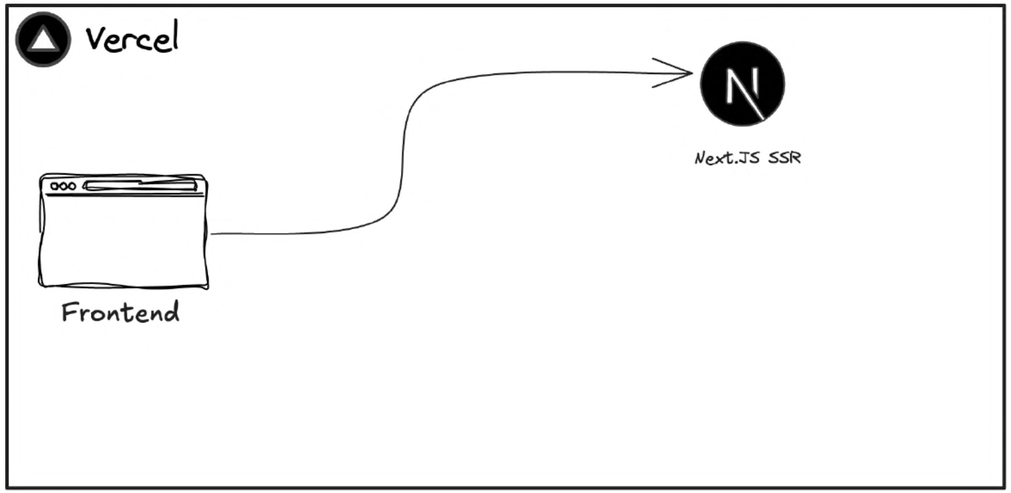
Starting with an API mess

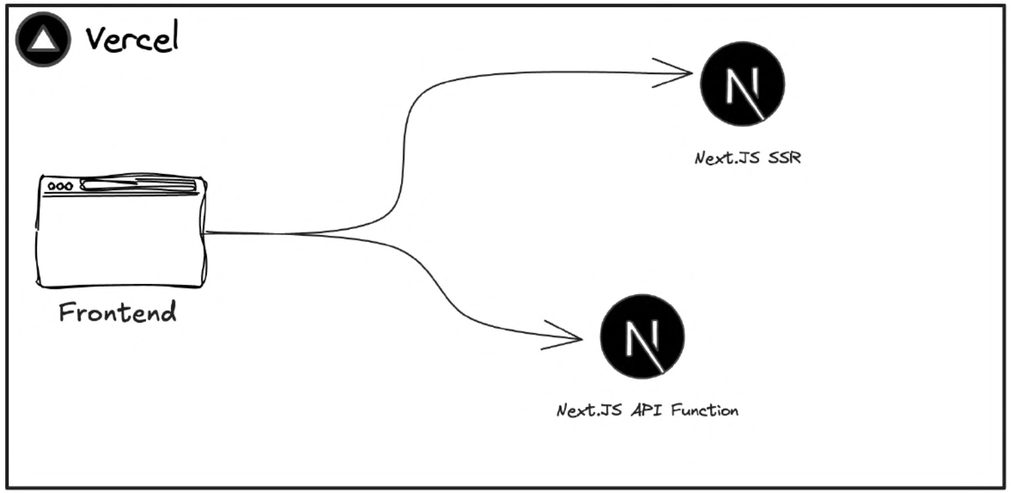
MONOLITHIC EXPRESS APP | NEXT.JS API ROUTES ON VERCEL

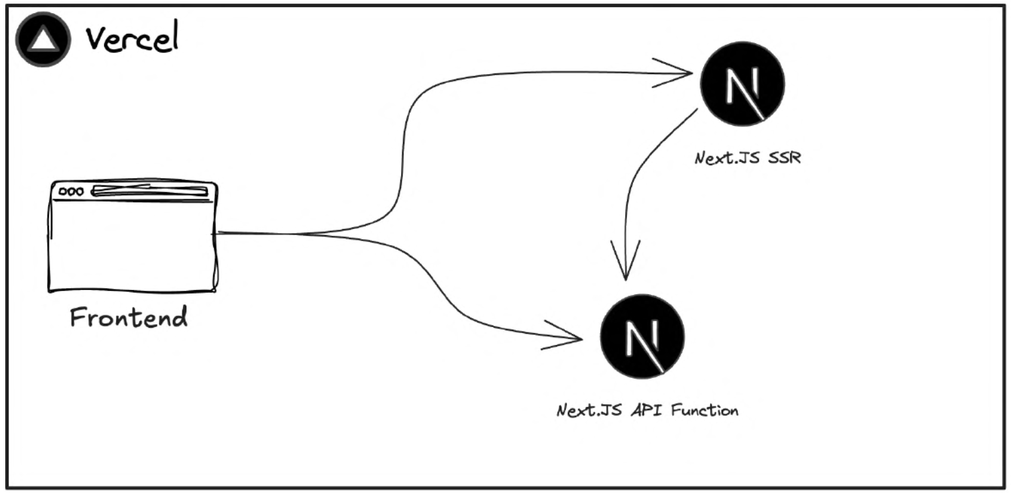

Vercel
Next.JS SSR
Frontend
Next.JS API Function
MongoDB Atlas
MongoDB
AWS account
Mobile App
Monolithic
Express API
Public GQL API

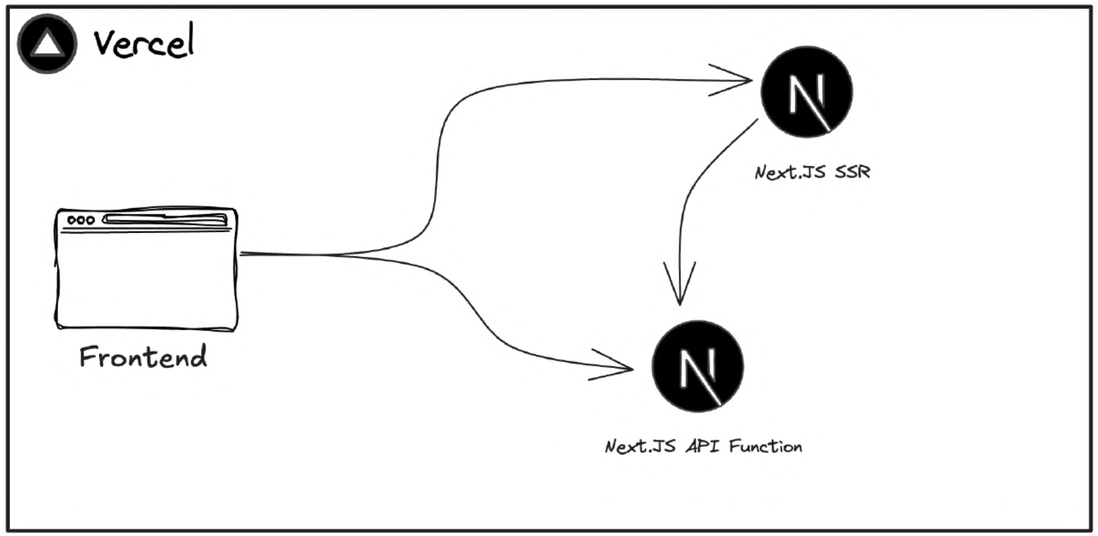


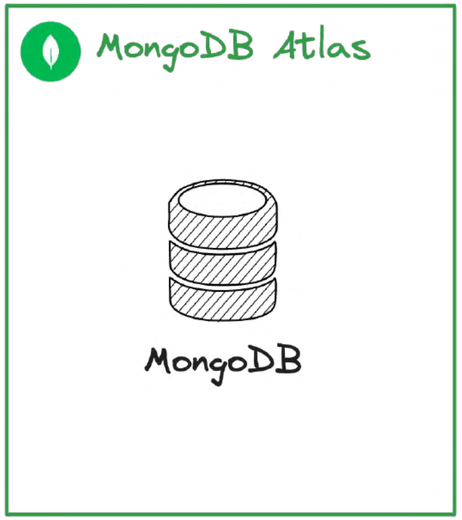


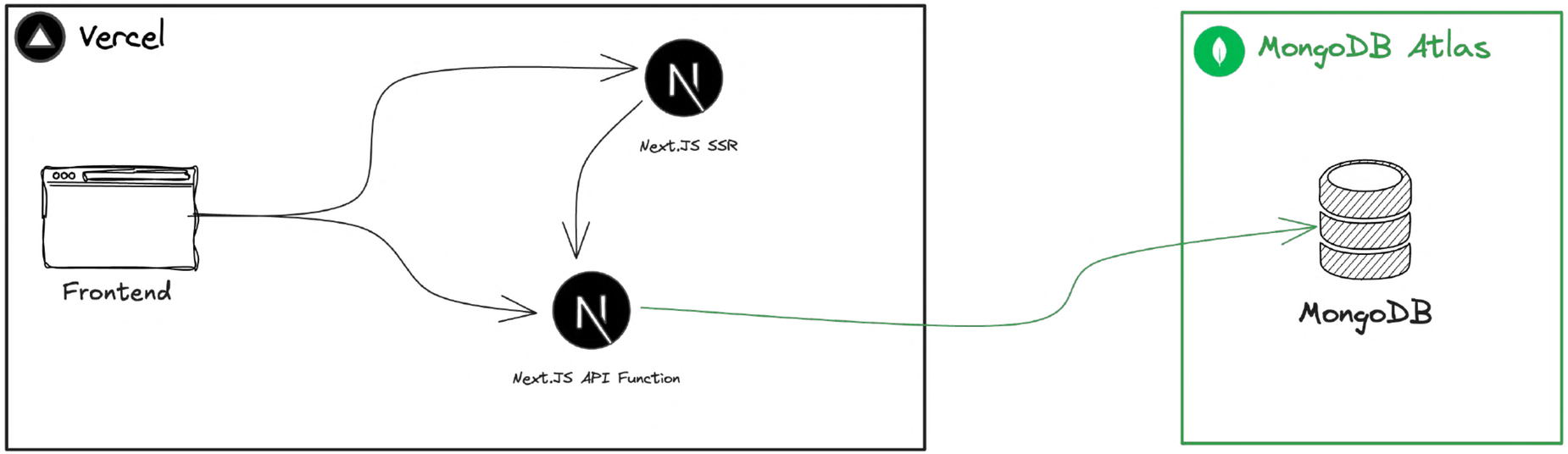


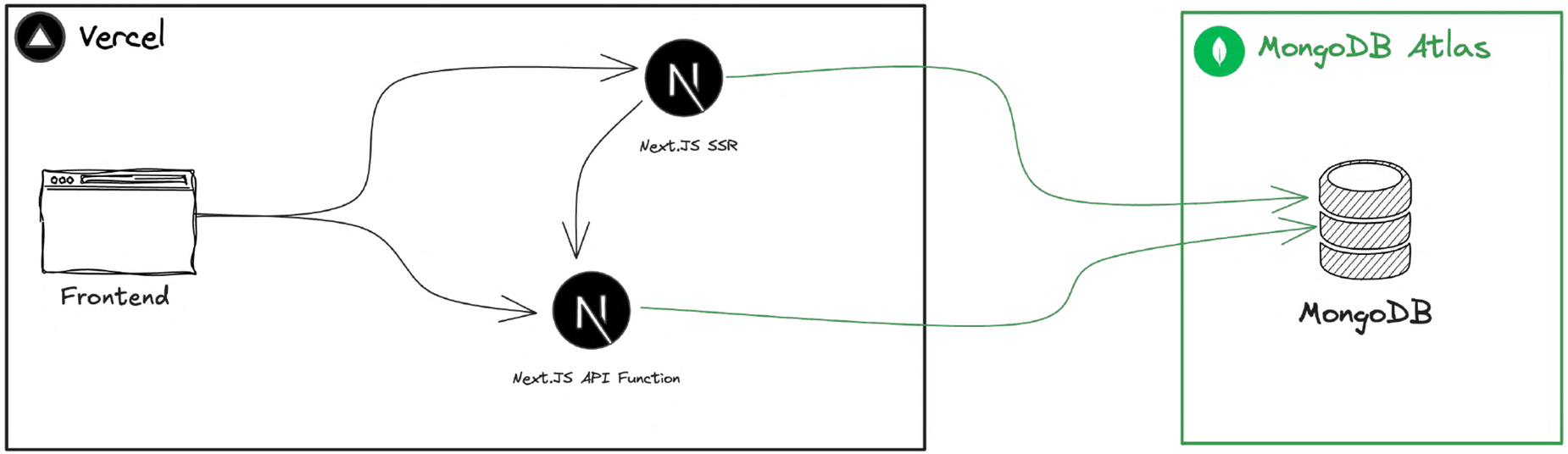


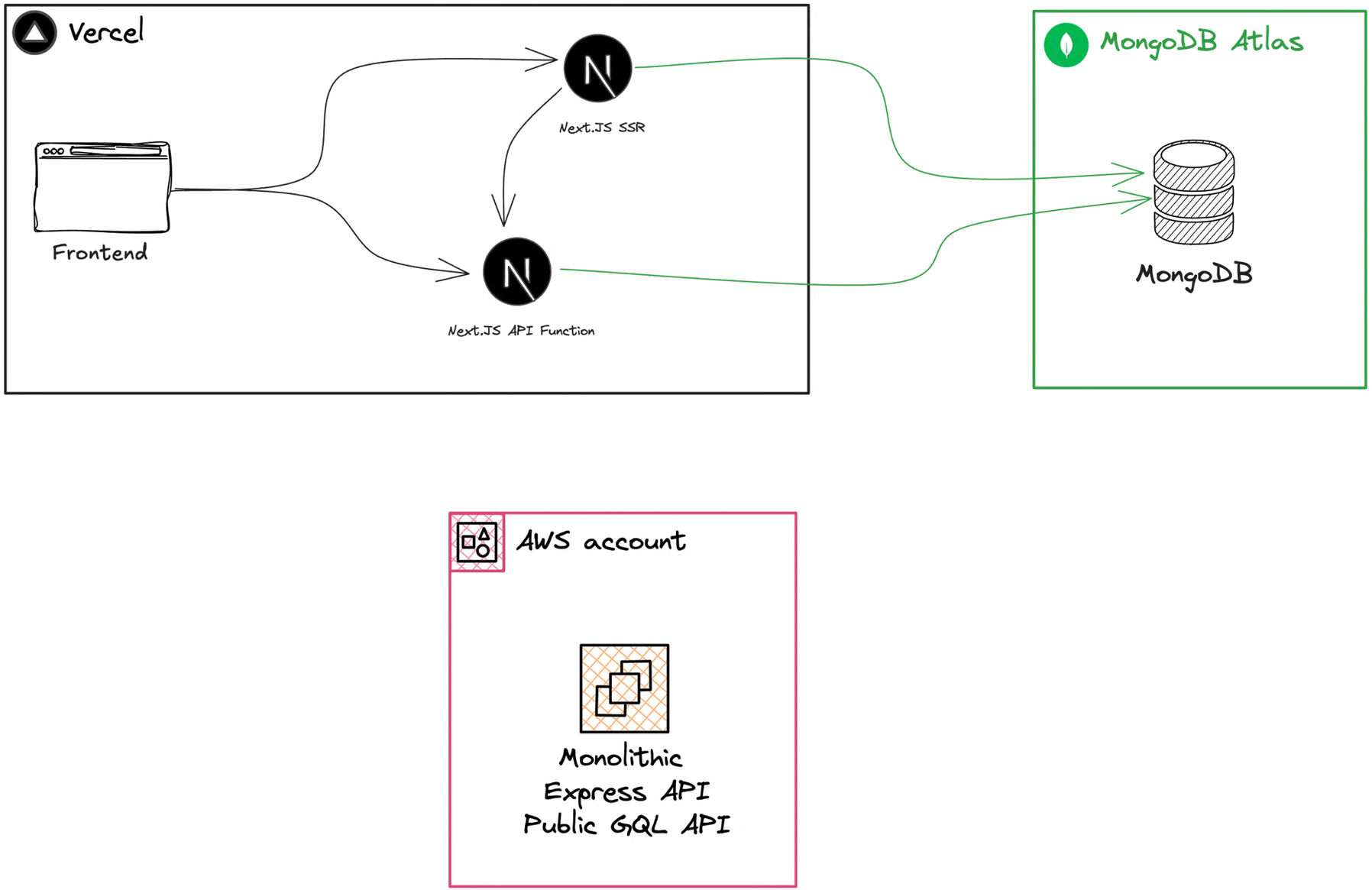


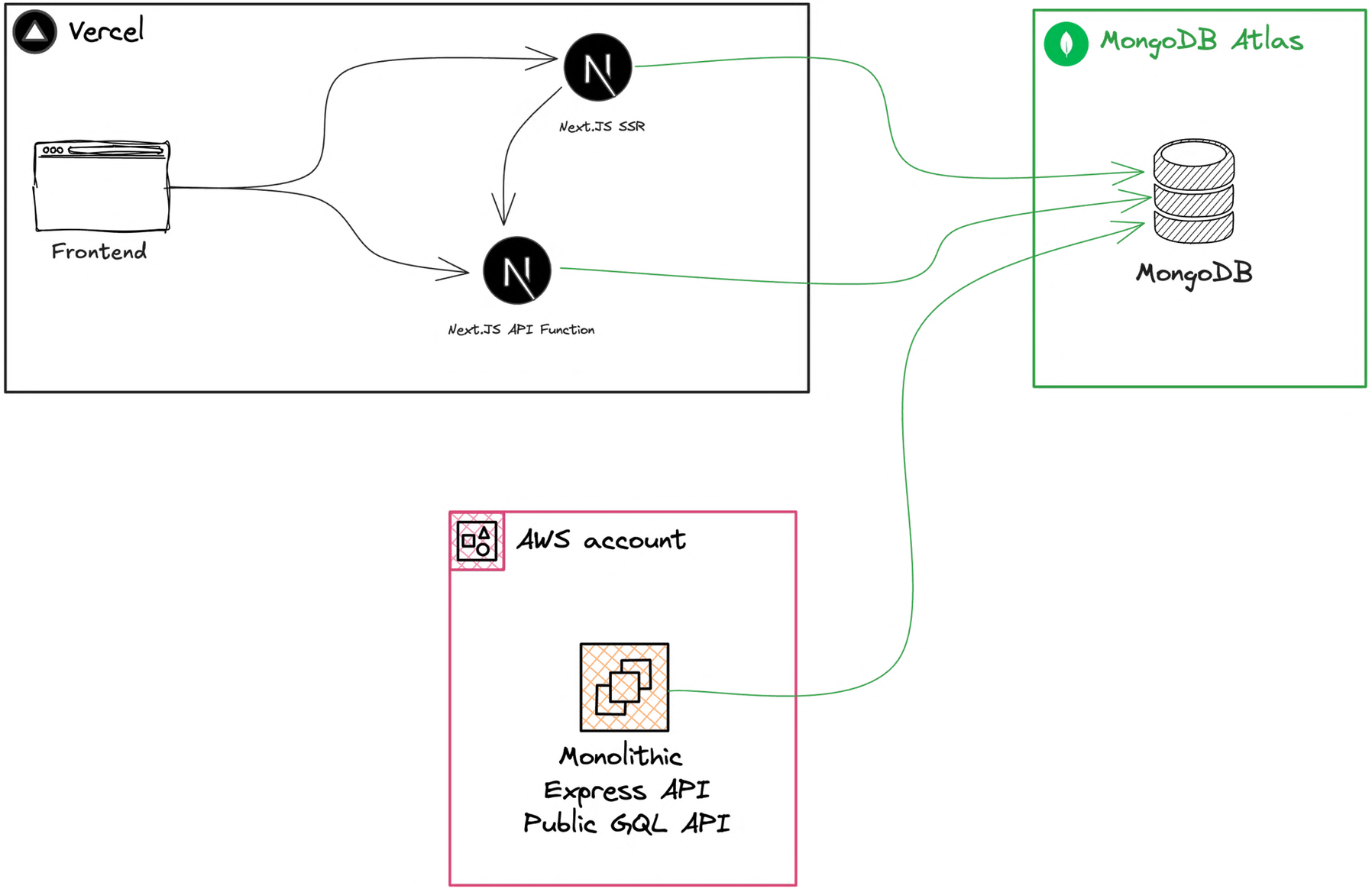

Vercel
Next.JS SSR
Frontend
Next.JS API Function

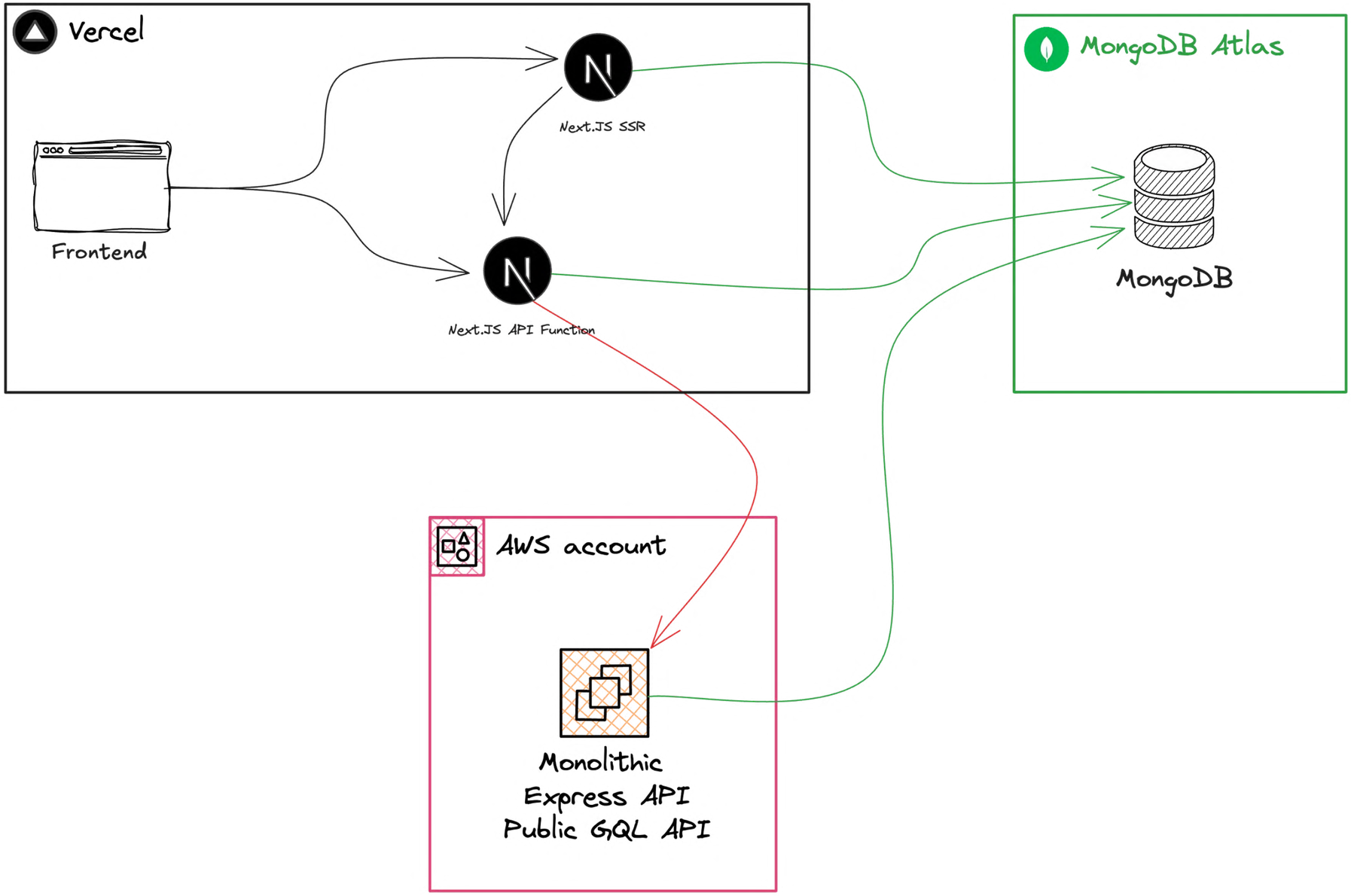

MongoDB Atlas
MongoDB

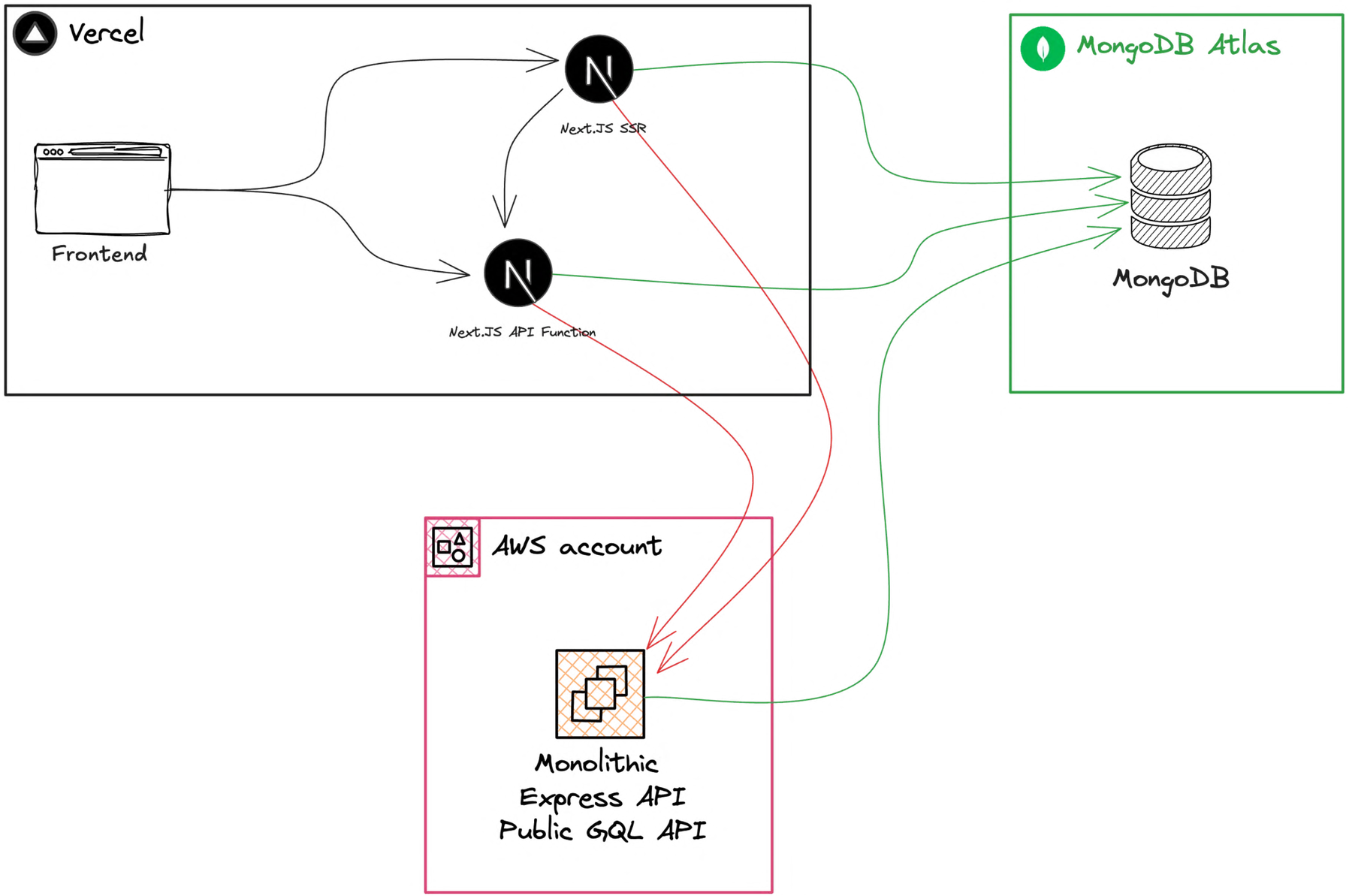

Vercel
Next.JS SSR
Frontend
Next.JS API Function
MongoDB Atlas
MongoDB

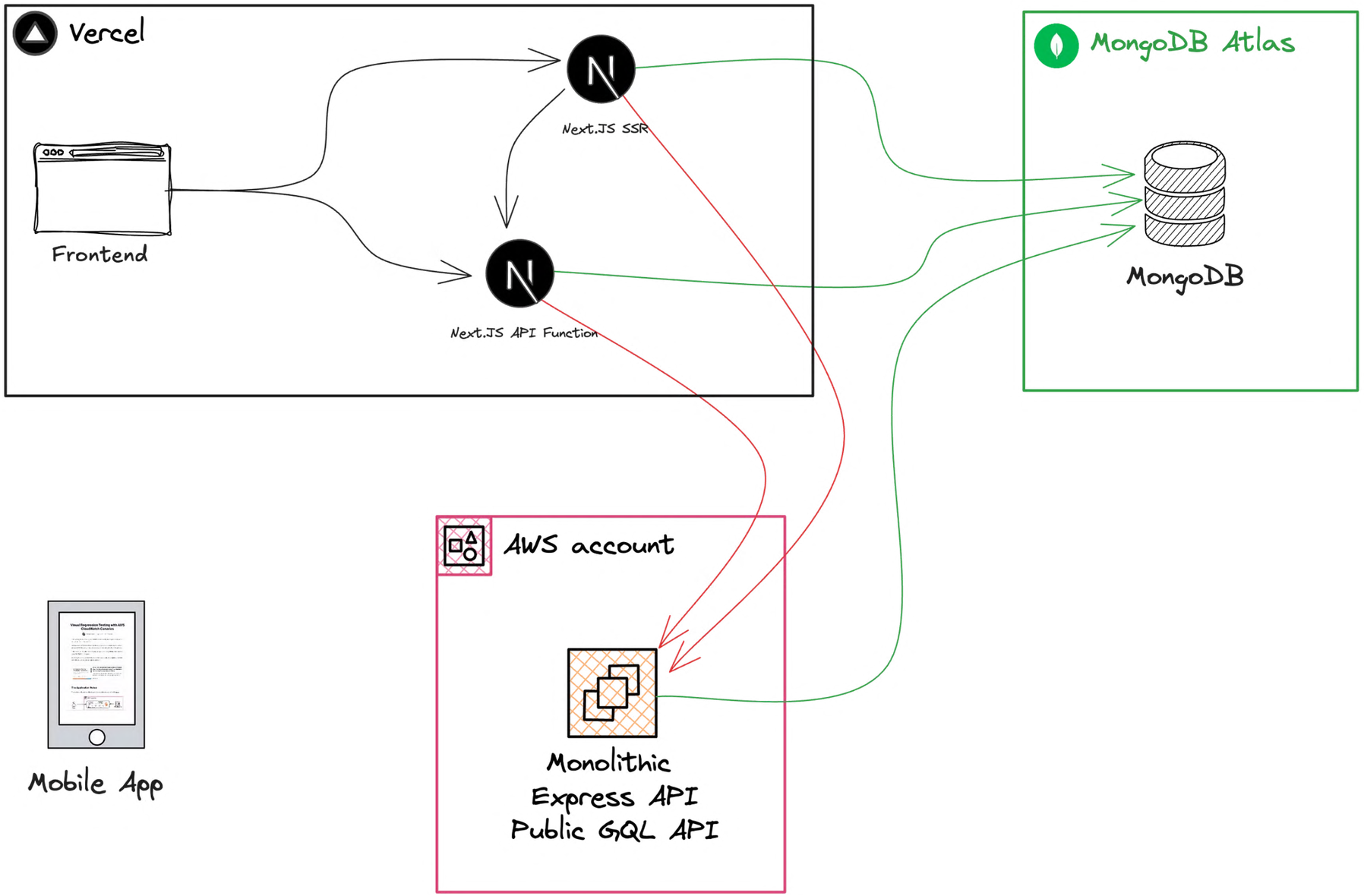

Vercel
Next.JS SSR
Frontend
Next.JS API Function
MongoDB Atlas
MongoDB

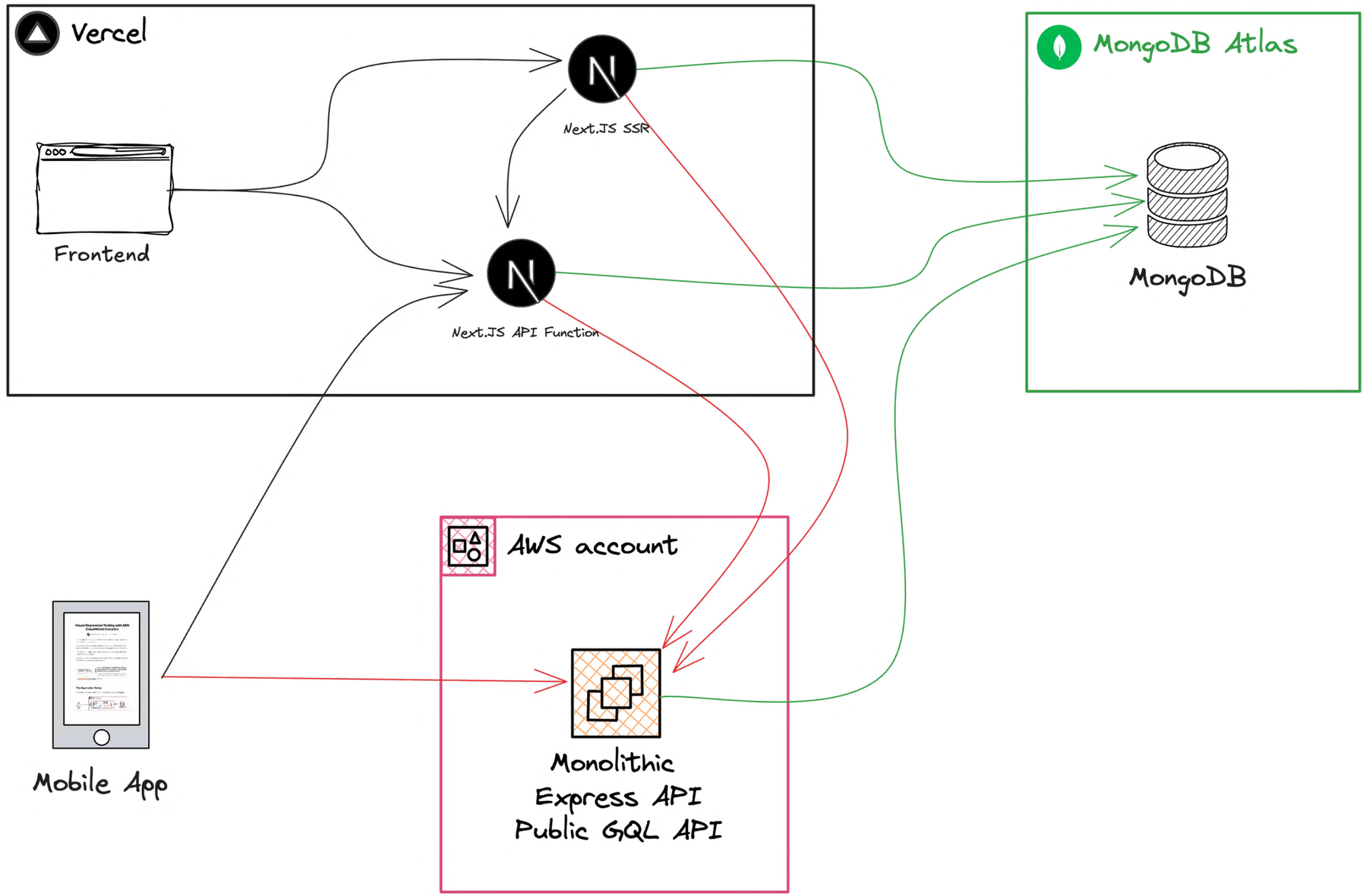

Vercel
Next.JS SSR
Frontend
Next.JS API Function
MongoDB Atlas
MongoDB
AWS account
Monolithic
Express API
Public GQL API


Vercel
Next.JS SSR
Frontend
Next.JS API Function
MongoDB Atlas
MongoDB
AWS account
Monolithic
Express API
Public GQL API


Vercel
Next.JS SSR
Frontend
Next.JS API Function
MongoDB Atlas
MongoDB
AWS account
Monolithic
Express API
Public GQL API


Vercel
Next.JS SSR
Frontend
Next.JS API Function
MongoDB Atlas
MongoDB
AWS account
Monolithic
Express API
Public GQL API


Vercel
Next.JS SSR
Frontend
Next.JS API Function
MongoDB Atlas
MongoDB
AWS account
Monolithic
Express API
Public GQL API
Mobile App


Vercel
Next.JS SSR
Frontend
Next.JS API Function
MongoDB Atlas
MongoDB
AWS account
Mobile App
Monolithic
Express API
Public GQL API

Downsides 👎🏽

Standardization

- Business Logic mixed with Data Access
- Authentication always done differently
- Observability (VM, Sentry, CloudWatch, Axiom, Vercel)

Caching

- Cache only time-based, no purging possible
- Some manual redis caches - no overview

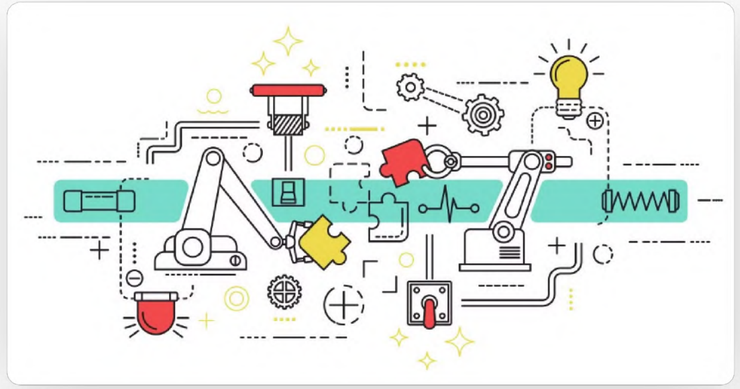
Developer Experience

- Duplication of API/Implementations
- Type-Safety

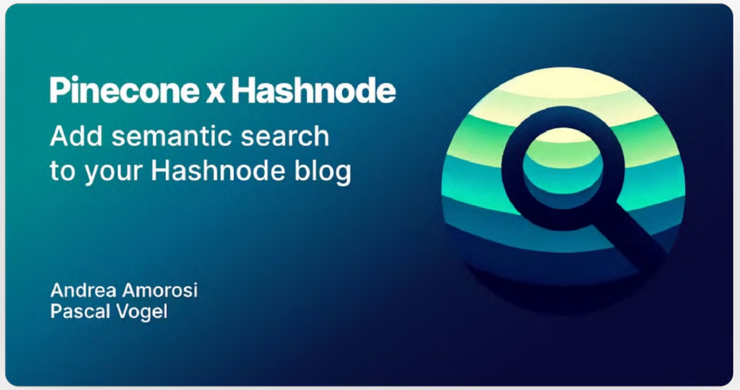
→ One Central API

Why a public API?

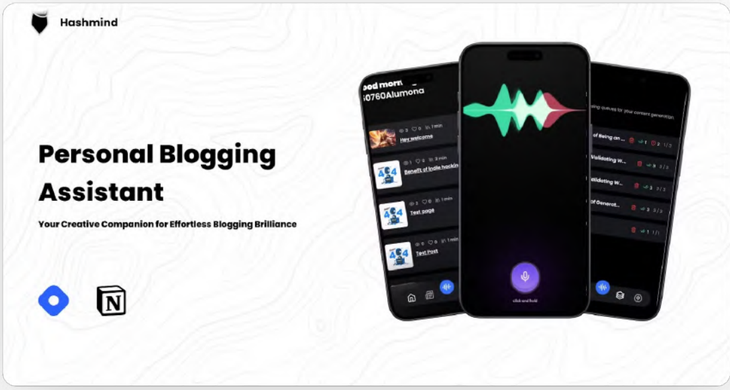
- Our API is in its nature public → Hosting of public blog posts
- Give developers more freedom to build integrations



Allen Helton, Serverless Cross-Post



Andrea Armosi, Pascal Vogel - Semantic Search



Benaiah - Hashmind Blogging Assistant

→ USE HASHNODE AS A HEADLESS CMS

Introduction to GraphQL

OPERATIONS | SCHEMA | ONE ENDPOINT

Introduction to GraphQL

OPERATIONS

```
query User {
  user(username: "SandroVolpicella") {
    id
    username
    bio {
      text
    }
  }
}
```

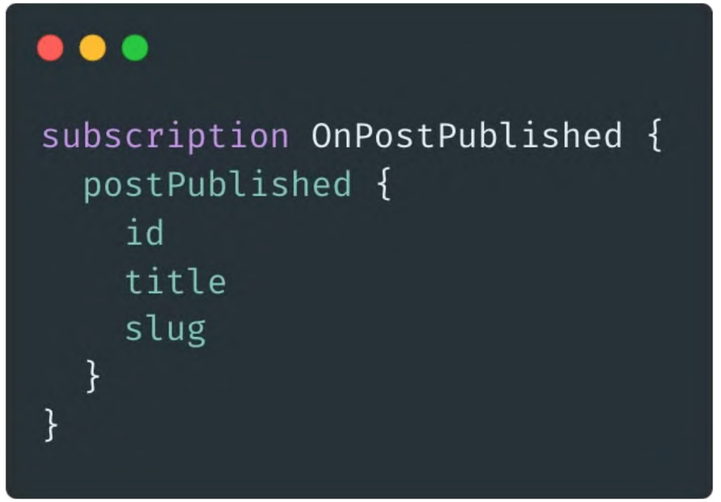
Queries

Choose which data to get

```
mutation PublishPost {
  publishPost(
    input: {
      title: "Amazing Post",
      publicationId: "1234",
      contentMarkdown: "# Heading 1"
    }
  ) {
    post {
      id
      slug
    }
  }
}
```

Mutations

Mutate data. Create posts, remove users, buy pro plan.



```
subscription OnPostPublished {
  postPublished {
    id
    title
    slug
  }
}
```

Subscriptions

Listen to changes and get real-time updates.

Introduction to GraphQL

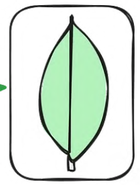
SCHEMA

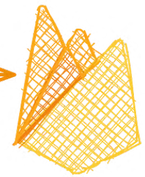
The Schema SDL

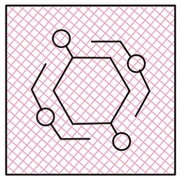
```
type Publication implements Node {
  """The about section of the publication."""
  about: Content

  """
  Returns the list of drafts of the authenticated user in the publication.
  """
  drafts(
    """A cursor to the last item in the previous page."""
    after: String
    """The filters to be applied to the draft list."""
    filter: PublicationDraftConnectionFilter
    """The number of drafts to return."""
    first: Int! @constraint(min: 1, max: 50)
  ): DraftConnection! @requireAuth

  isTeam: Boolean!

  pro: PublicationProInfo @hidden

  scheduledDrafts(
    """A cursor to the last item in the previous page."""
    after: String
    """The filters to be applied to the draft list."""
    filter: PublicationDraftConnectionFilter
    """The number of scheduled drafts to return."""
    first: Int! @constraint(min: 1, max: 50)
  ): DraftConnection! @requireAuth
```

The Schema SDL

```
type Publication implements Node {
  """The about section of the publication."""
  about: Content

  """
  Returns the list of drafts of the authenticated user in the publication.
  """
  drafts(
    """A cursor to the last item in the previous page."""
    after: String
    """The filters to be applied to the draft list."""
    filter: PublicationDraftConnectionFilter
    """The number of drafts to return."""
    first: Int! @constraint(min: 1, max: 50)
  ): DraftConnection! @requireAuth

  isTeam: Boolean!

  pro: PublicationProInfo @hidden

  scheduledDrafts(
    """A cursor to the last item in the previous page."""
    after: String
    """The filters to be applied to the draft list."""
    filter: PublicationDraftConnectionFilter
    """The number of scheduled drafts to return."""
    first: Int! @constraint(min: 1, max: 50)
  ): DraftConnection! @requireAuth
```

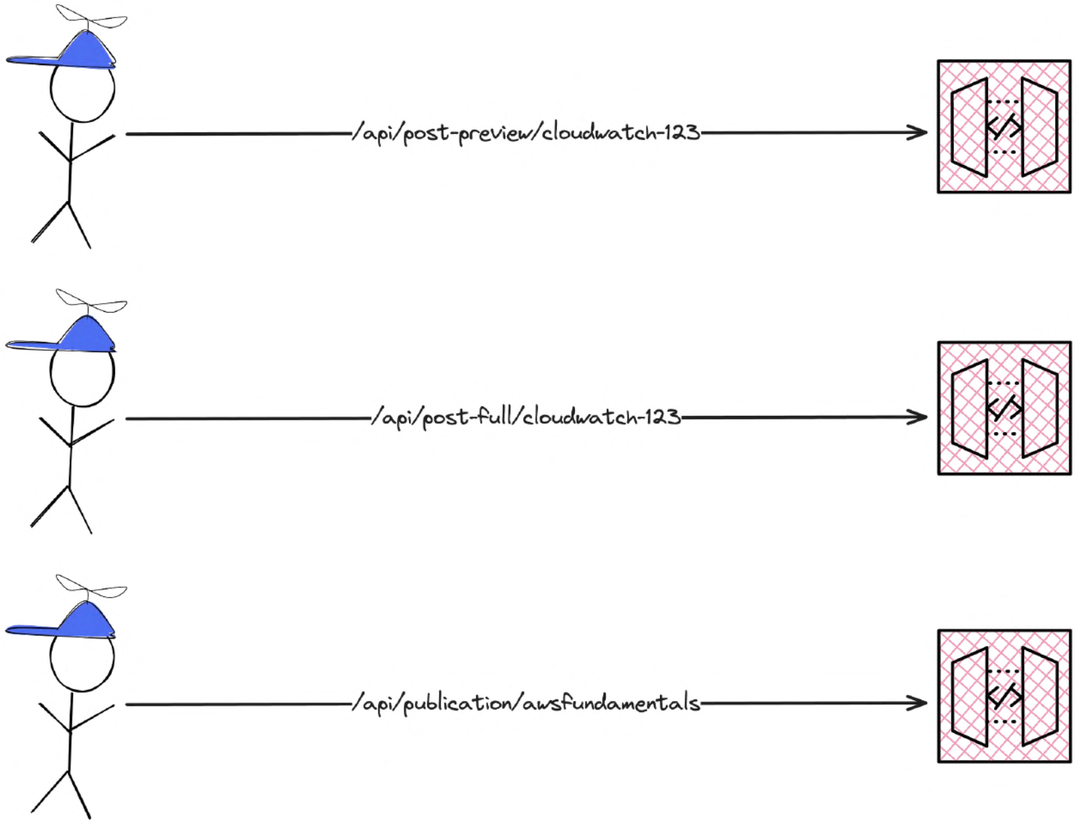






Introduction to GraphQL

ONE ENDPOINT



REST: Multiple Endpoints

In REST you need to define several endpoints for similar data

GQL: One Endpoint

GraphQL has one POST Endpoint

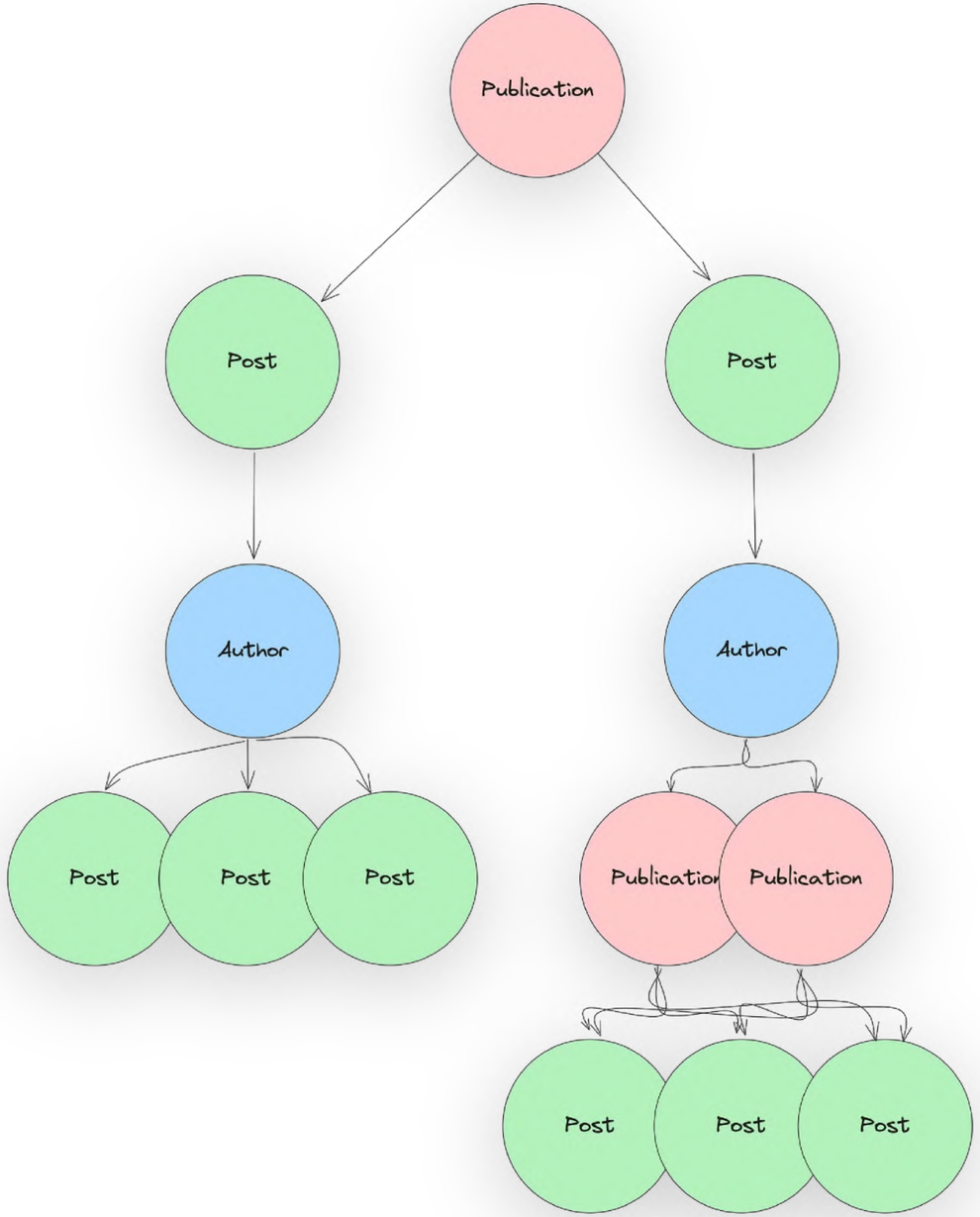
fewer network requests, simpler observability

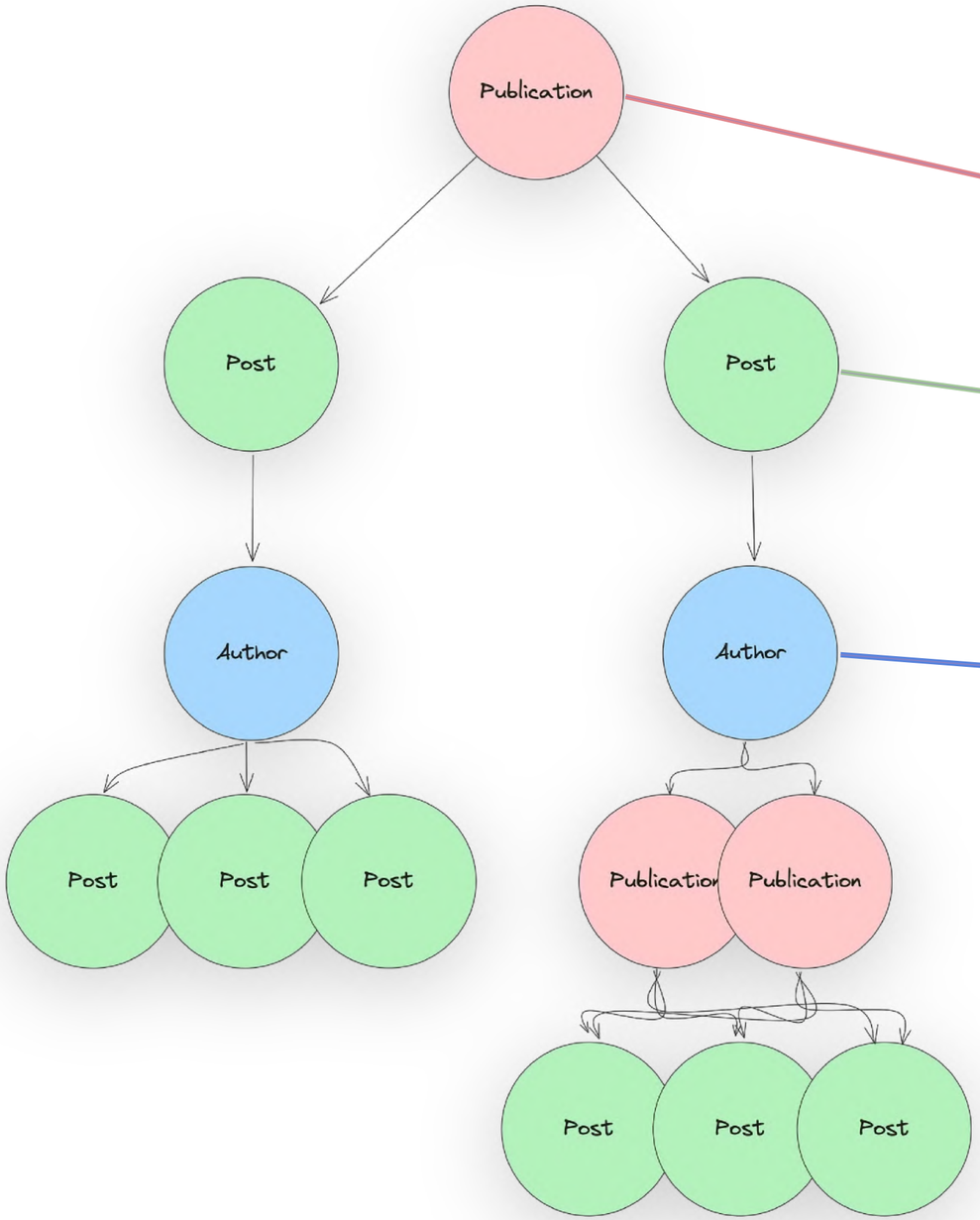
Why GraphQL?

GRAPH | CLIENT CHOOSES DATA | OBSERVABILITY

Why GraphQL?

GRAPH

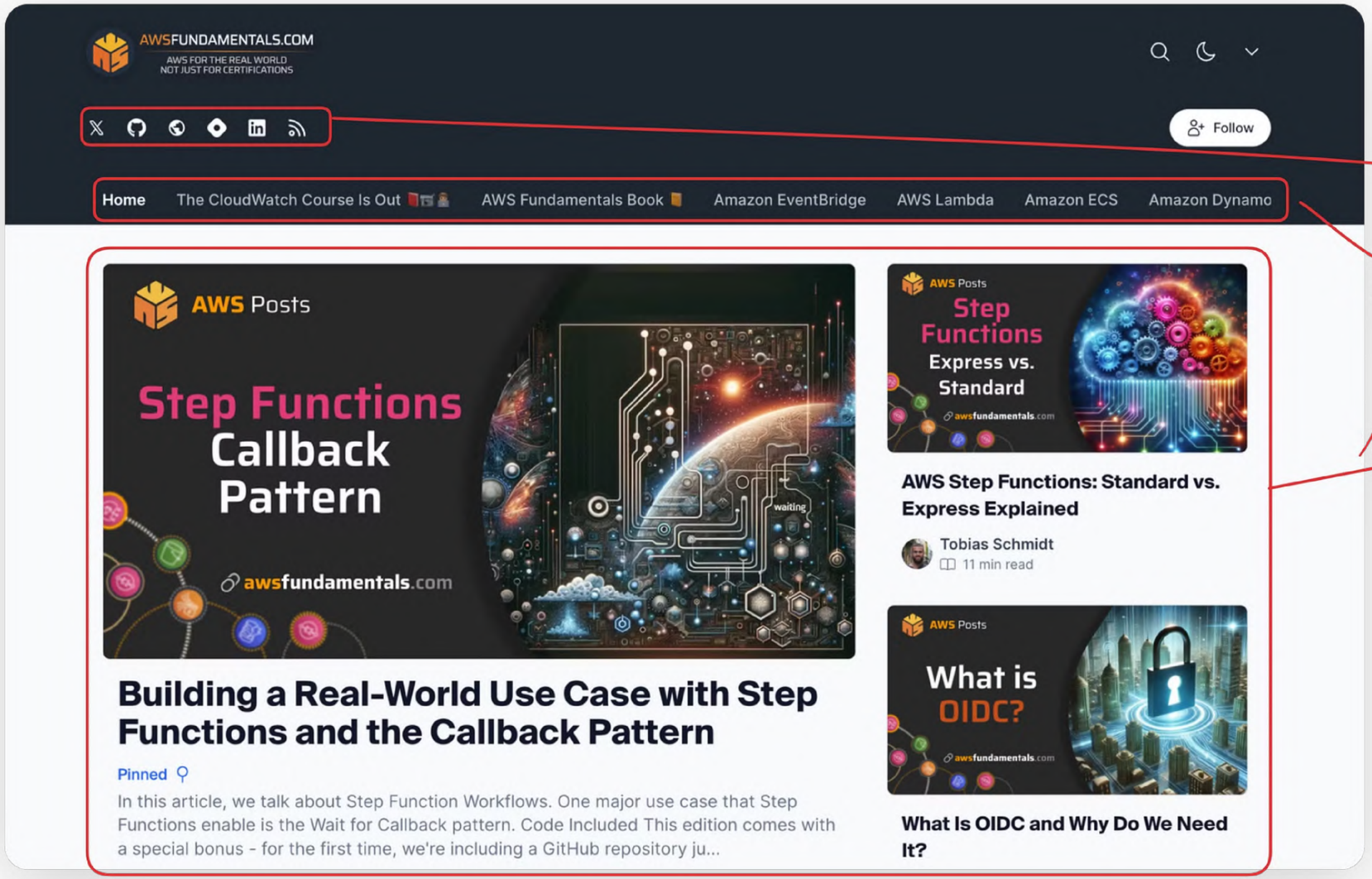

Publication
Post
Post
Author
Author
Post
Post
Post
Publication
Publication
Post
Post
Post

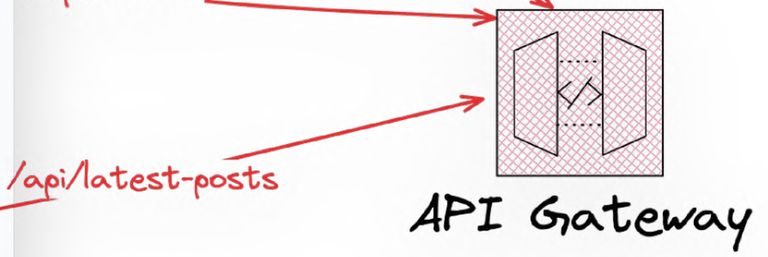


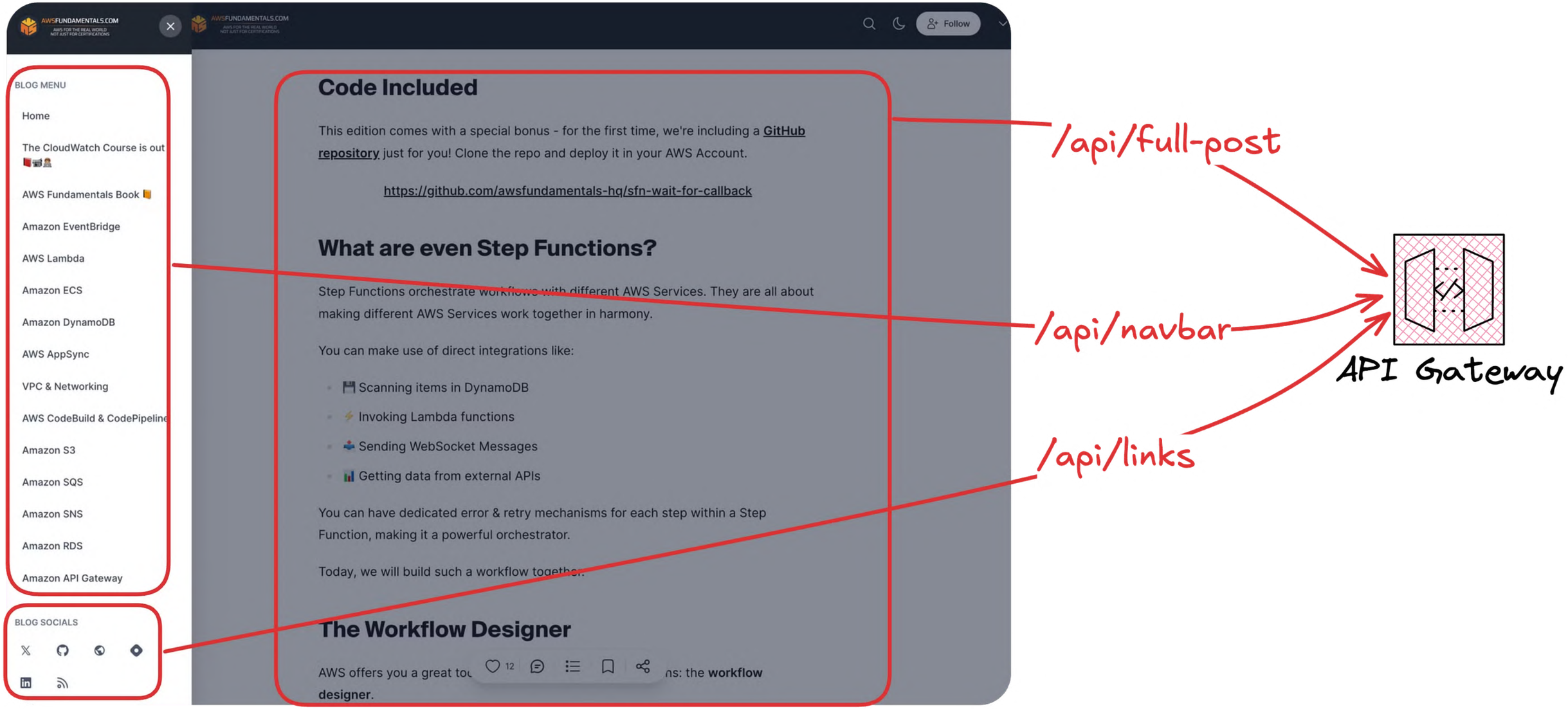

```
query GetPosts {
  publication(host: "blog.awsfundamentals.com") {
    id
    title
    posts(first: 2) {
      edges {
        node {
          id
          title
          author {
            id
            username
          }
        }
      }
    }
  }
}
```

Why GraphQL?

CLIENT CHOOSES DATA


AWSFUNDAMENTALS.COM
AWS FOR THE REAL WORLD
NOT JUST FOR CERTIFICATIONS
Follow
Home
The CloudWatch Course Is Out
AWS Fundamentals Book
Amazon EventBridge
AWS Lambda
Amazon ECS
Amazon Dynamo
AWS Posts
Step Functions
Callback
Pattern
awsfundamentals.com
Building a Real-World Use Case with Step Functions and the Callback Pattern
Pinned
In this article, we talk about Step Function Workflows. One major use case that Step Functions enable is the Wait for Callback pattern. Code Included This edition comes with a special bonus - for the first time, we're including a GitHub repository ju...
AWS Posts
Step Functions
Express vs. Standard
awsfundamentals.com
AWS Step Functions: Standard vs. Express Explained
Tobias Schmidt
11 min read
AWS Posts
What is OIDC?
awsfundamentals.com
What Is OIDC and Why Do We Need It?


/api/links
/api/navbar
/api/latest-posts
API Gateway


AWSFUNDAMENTALS.COM
AWS FOR THE REAL WORLD NOT JUST FOR CERTIFICATIONS
Follow
BLOG MENU
Home
The CloudWatch Course is out
AWS Fundamentals Book
Amazon EventBridge
AWS Lambda
Amazon ECS
Amazon DynamoDB
AWS AppSync
VPC & Networking
AWS CodeBuild & CodePipeline
Amazon S3
Amazon SQS
Amazon SNS
Amazon RDS
Amazon API Gateway
BLOG SOCIALS
Code Included
This edition comes with a special bonus - for the first time, we're including a GitHub repository just for you! Clone the repo and deploy it in your AWS Account.
https://github.com/awsfundamentals-hq/sfn-wait-for-callback
What are even Step Functions?
Step Functions orchestrate workflows with different AWS Services. They are all about making different AWS Services work together in harmony.
You can make use of direct integrations like:
Scanning items in DynamoDB
Invoking Lambda functions
Sending WebSocket Messages
Getting data from external APIs
You can have dedicated error & retry mechanisms for each step within a Step Function, making it a powerful orchestrator.
Today, we will build such a workflow together.
The Workflow Designer
AWS offers you a great to
ns: the workflow designer.
12
/api/full-post
/api/navbar
/api/links
API Gateway

Why GQL over REST → Client chooses data

PREVIEW

```
query PubWithPreviews {
  publication(host: "blog.awsfundamentals.com") {
    id
    title
    links {
      twitter
    }
    preferences {
      navBarItems {
        id
        label
      }
    }
    posts(first: 3) {
      edges {
        node {
          id
          title
          author {
            name
          }
        }
      }
    }
  }
}
```

FULL POST

```
query PostWithContent {
  publication(host: "blog.awsfundamentals.com") {
    id
    title
    links {
      twitter
    }
    preferences {
      navBarItems {
        id
        label
      }
    }
    posts(first: 3) {
      edges {
        node {
          id
          title
          author {
            name
          }
          content {
            markdown
          }
        }
      }
    }
  }
}
```

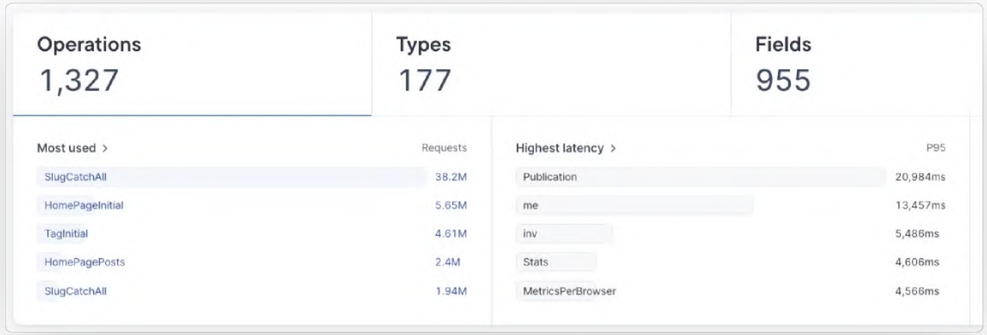
Why GraphQL?

PLAYGROUND, OBSERVABILITY, TYPE-SAFETY

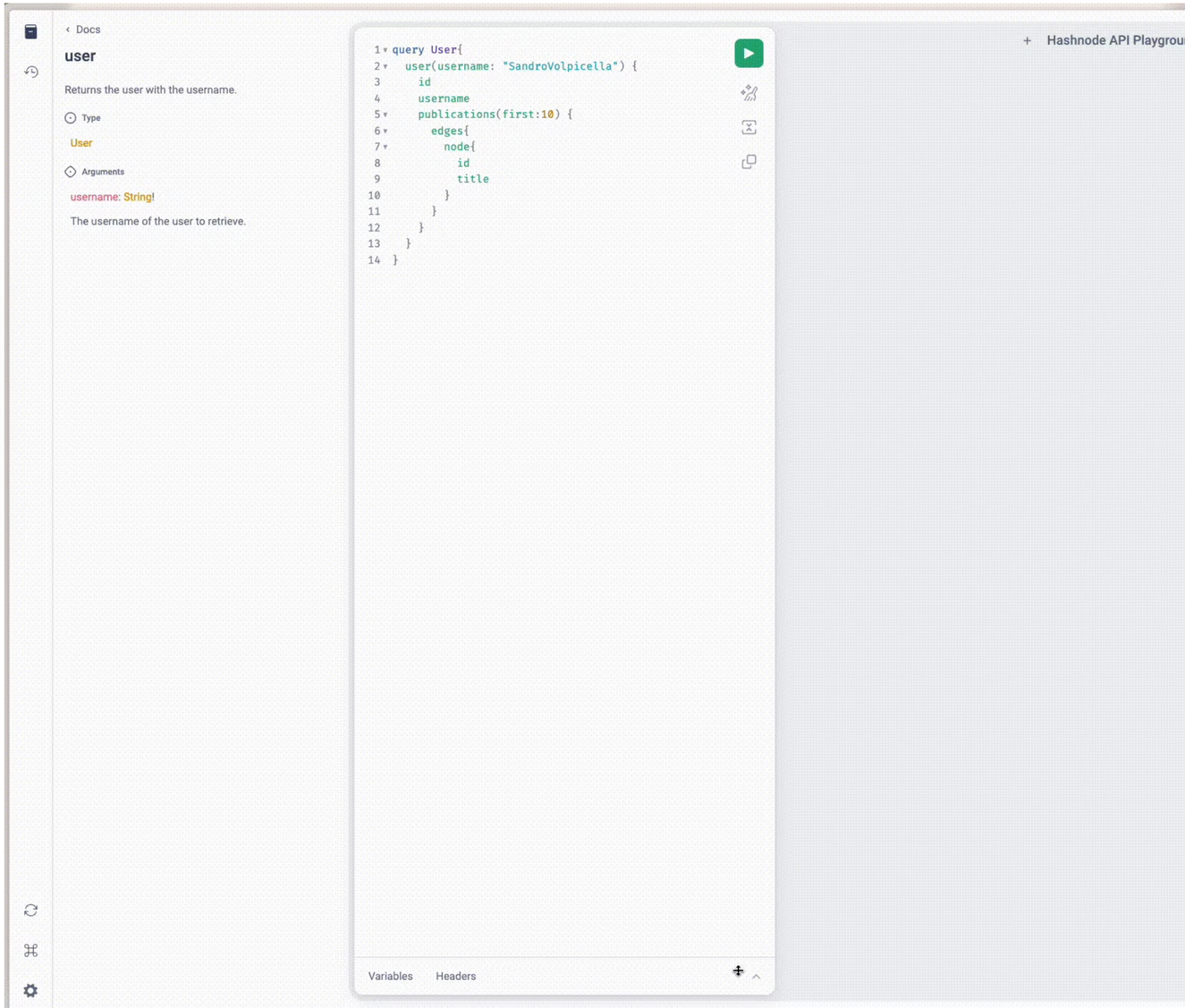
Playground, Observability, Type-Safety

Playground: **Live documentation** and execution environment: gql.hashnode.com

Observability: Errors by types/users



Type-Safety: Code Generation depending on the used queries



Serverless implementation of GraphQL API

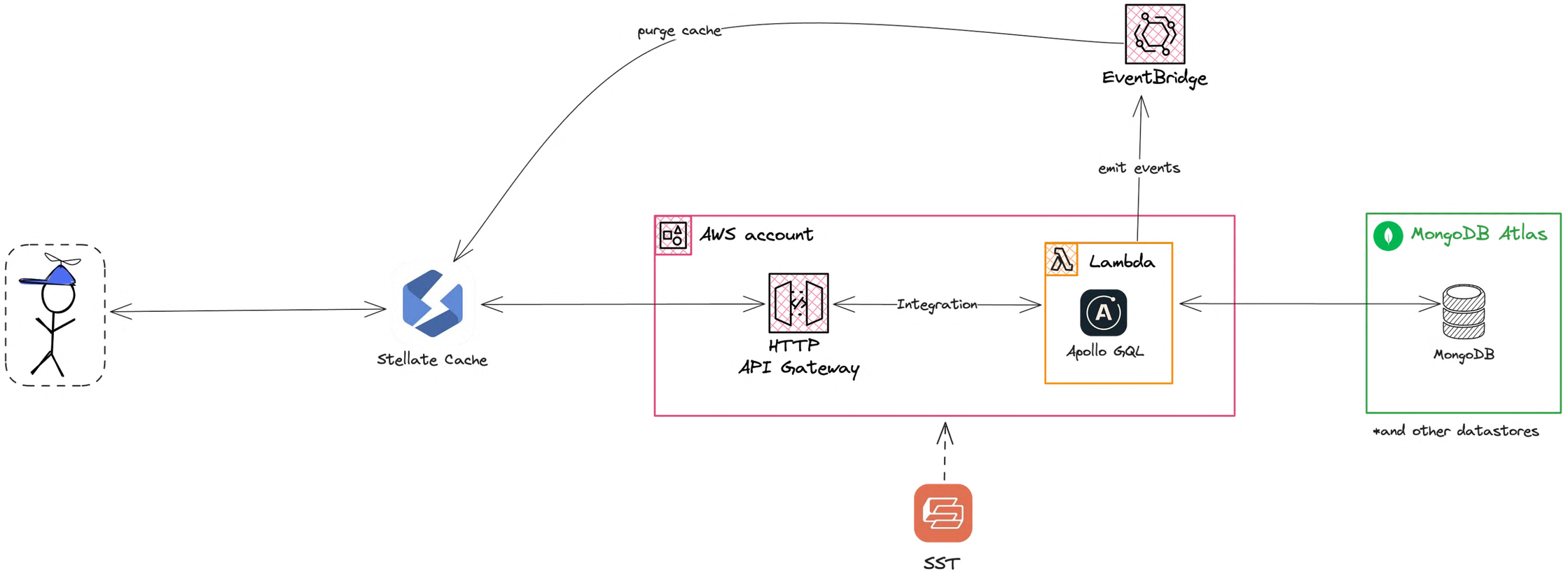
API GATEWAY | LAMBDA | APOLLO | SST

</>

Serverless implementation of GraphQL API

ARCHITECTURE

Architecture



Why no AppSync?

Technical Misses in AppSync (back then):

- No executable schema
- API should be public → API Key needed (workaround needed)
- Authentication: No cookies in request without workarounds (Lambda at Edge) ⚠️ **now changed**
- VTL hard to build and debug ⚠️ **now JS resolvers**
- We don't need direct integrations

Main Part:

- Team was familiar with Apollo (old API on Apollo)
- Apollo does have nice features + big community

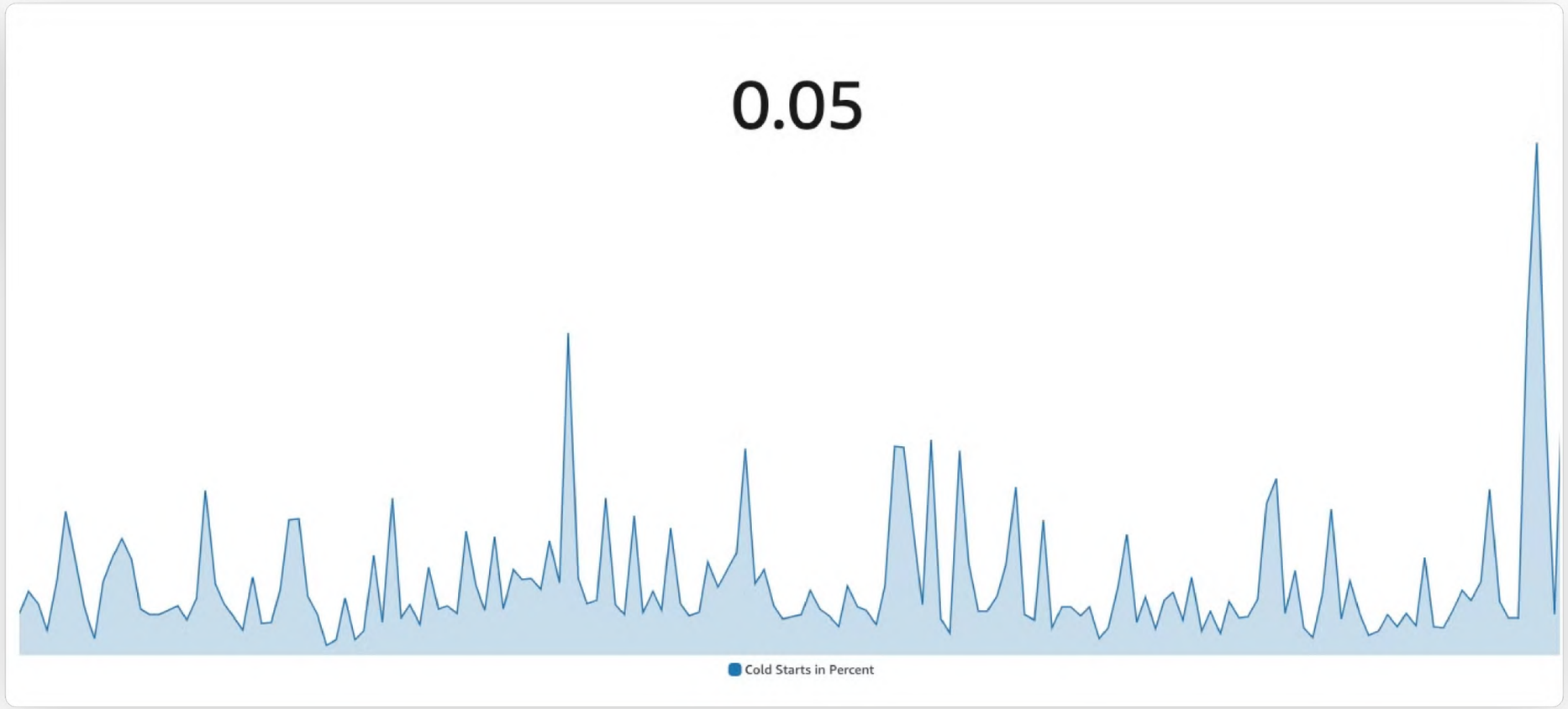
Let's talk about cold starts

Bundle Size

Code properties Info

Package size
4 MB

Number of Cold Starts in % (already *100 👆🏽)



Let's talk about cold starts

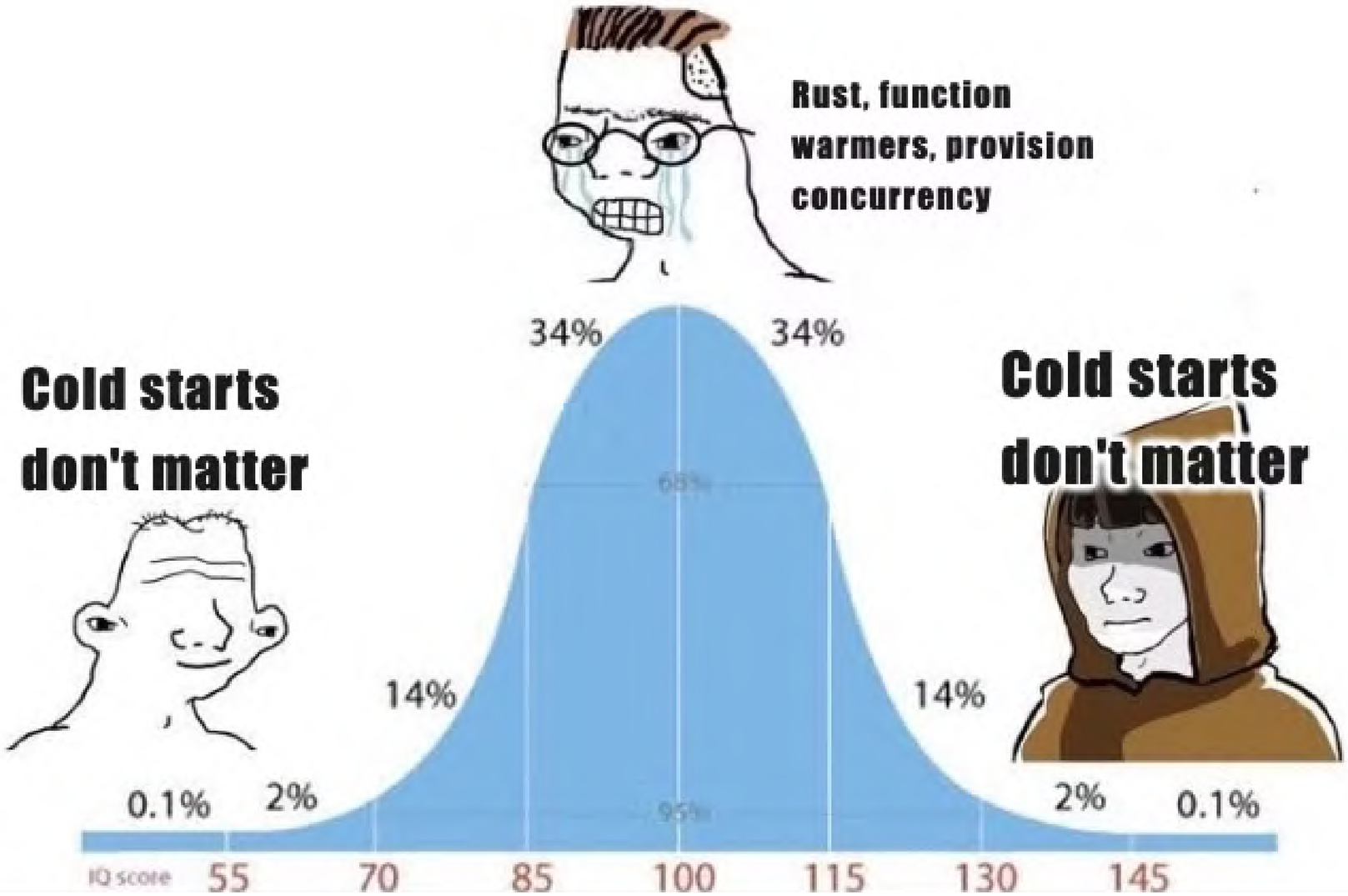
Daily	totalRequests	totalColdStarts	coldStartPct	AverageColdStartMs	P95Ms
2024-06-21…	1455326	1641	0.1128	144.4553	479.9773
2024-06-20…	1296888	931	0.07179	201.8736	657.6846
2024-06-19…	1130920	1008	0.08913	216.9162	722.8642
2024-06-18…	1170211	1043	0.08913	176.4889	617.5303
2024-06-17…	1079429	1157	0.1072	182.4302	657.6846
2024-06-16…	1729176	980	0.05667	108.3641	328.9003
2024-06-15…	1068530	837	0.07833	173.2719	527.5453

- Cold starts **are bad**
- But they **don't occur very often**
- We use **caching a lot**

Let's t

Daily	P95Ms
2024-06-21…	479.9773
2024-06-20…	657.6846
2024-06-19…	722.8642
2024-06-18…	617.5303
2024-06-17…	657.6846
2024-06-16…	328.9003
2024-06-15…	527.5453

- Cold start
- But they
- We use ca



Try Pitch

Caching

WHY? | EDGE VS. NON-EDGE | VERCEL VS. API RESPONSE

Caching

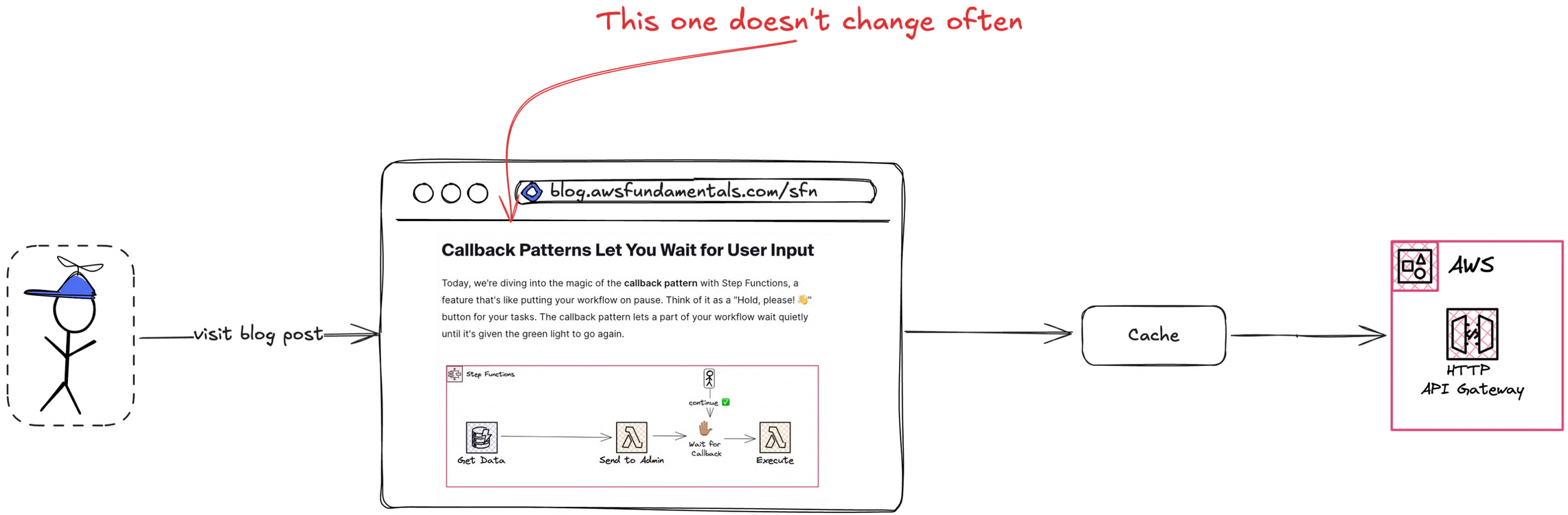
WHY?

</>

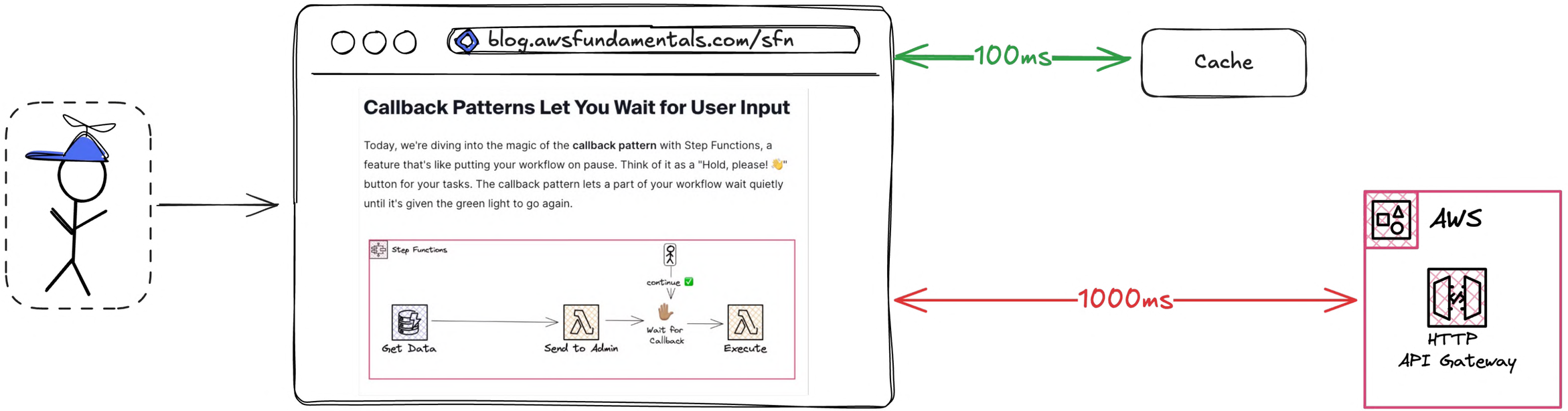
"Caching is a cheat code in distributed systems"

YAN CUI

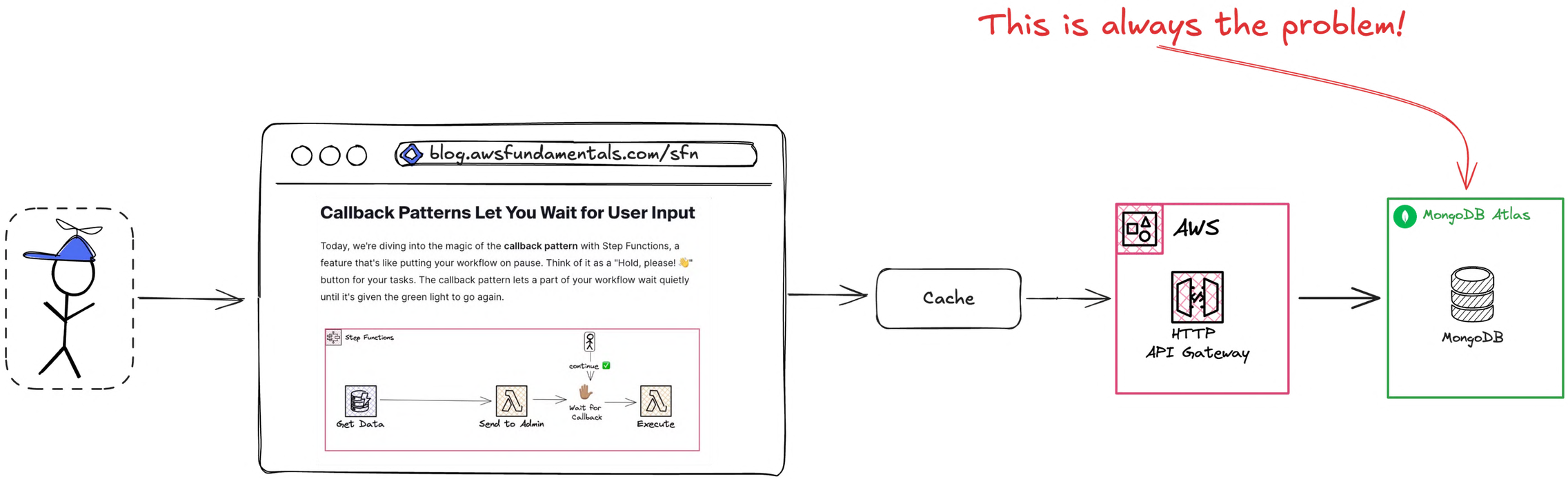
Static Content



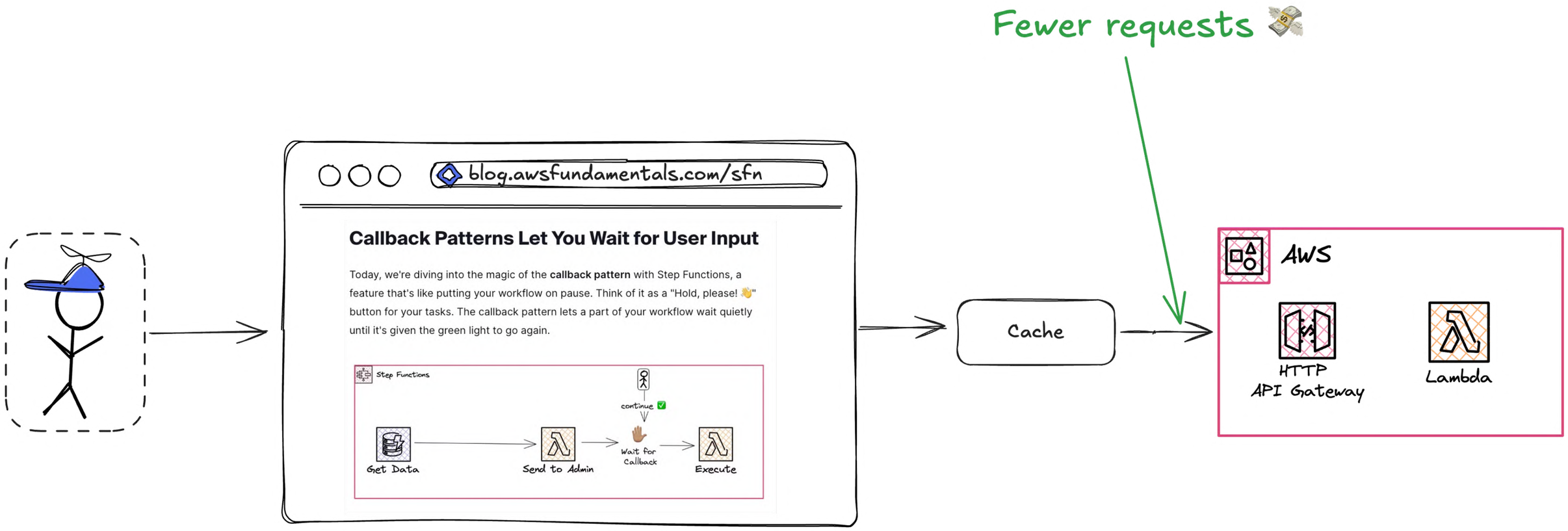
Performance



Protect Downstream Systems



Less Computation = More Money



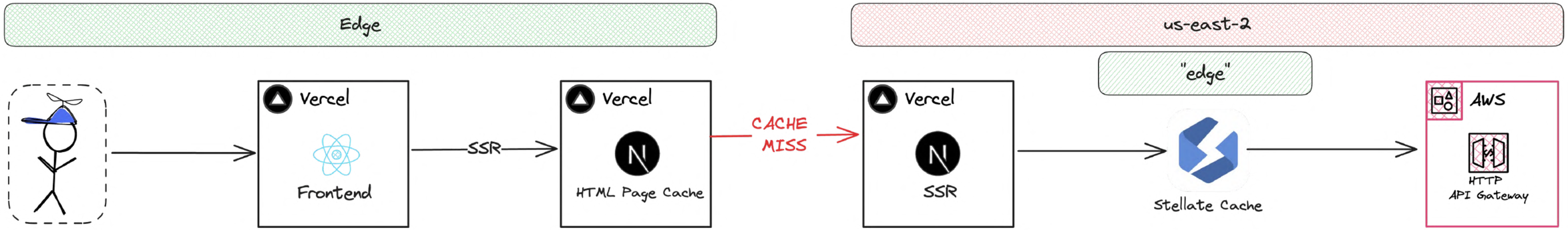
Why caching TLDR;

- static content doesn't change often
- performance
- protect downstream system
- less computation

Caching

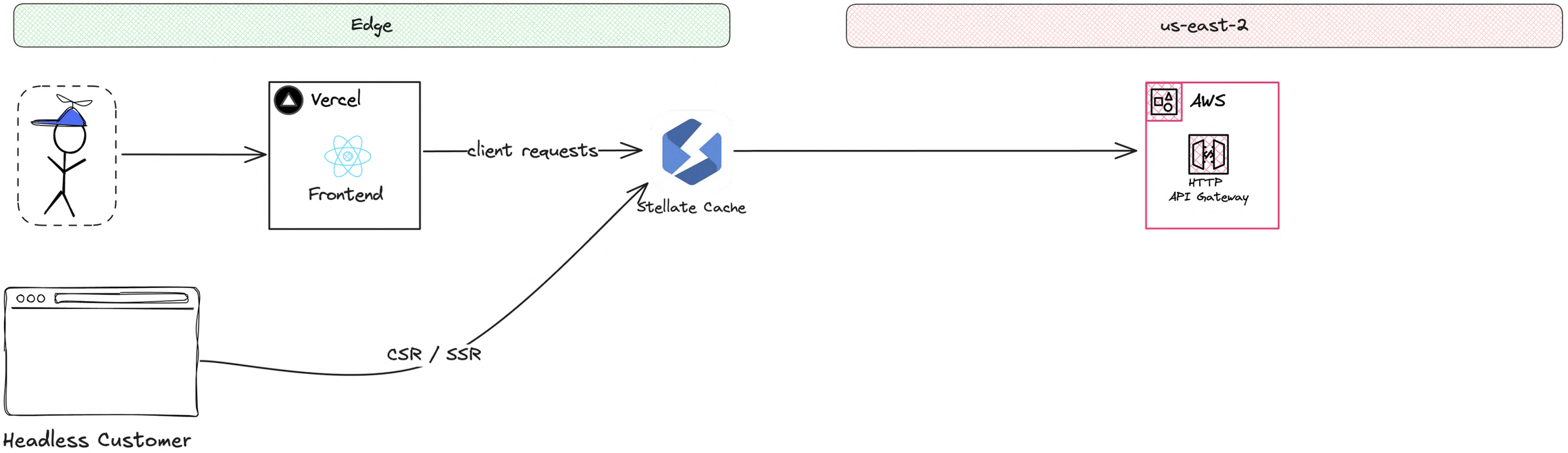
EDGE VS. NON-EDGE

Edge vs. non-edge - Server-Side Rendering



- still faster
- but not edge
- not 100% optimal
- more security than performance

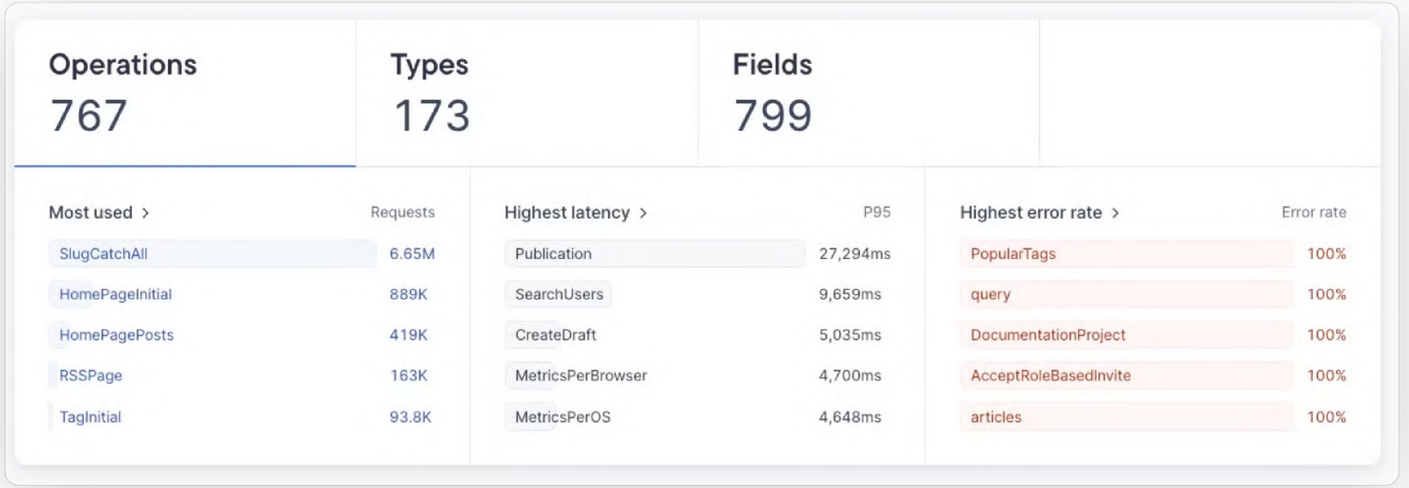
Edge vs. non-edge - Client-Side Rendering | Headless

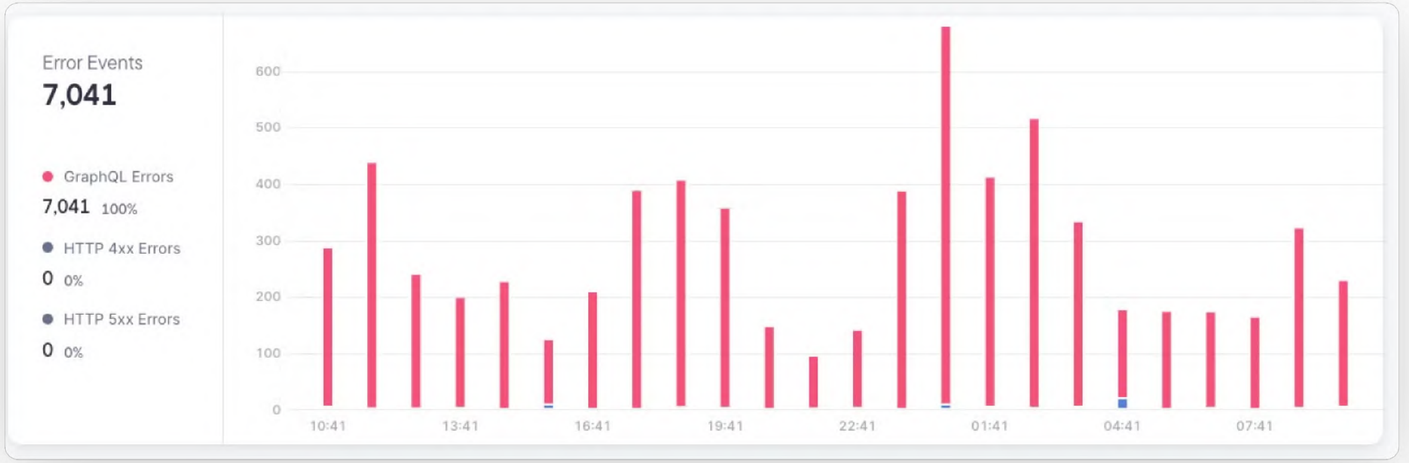


Stellate

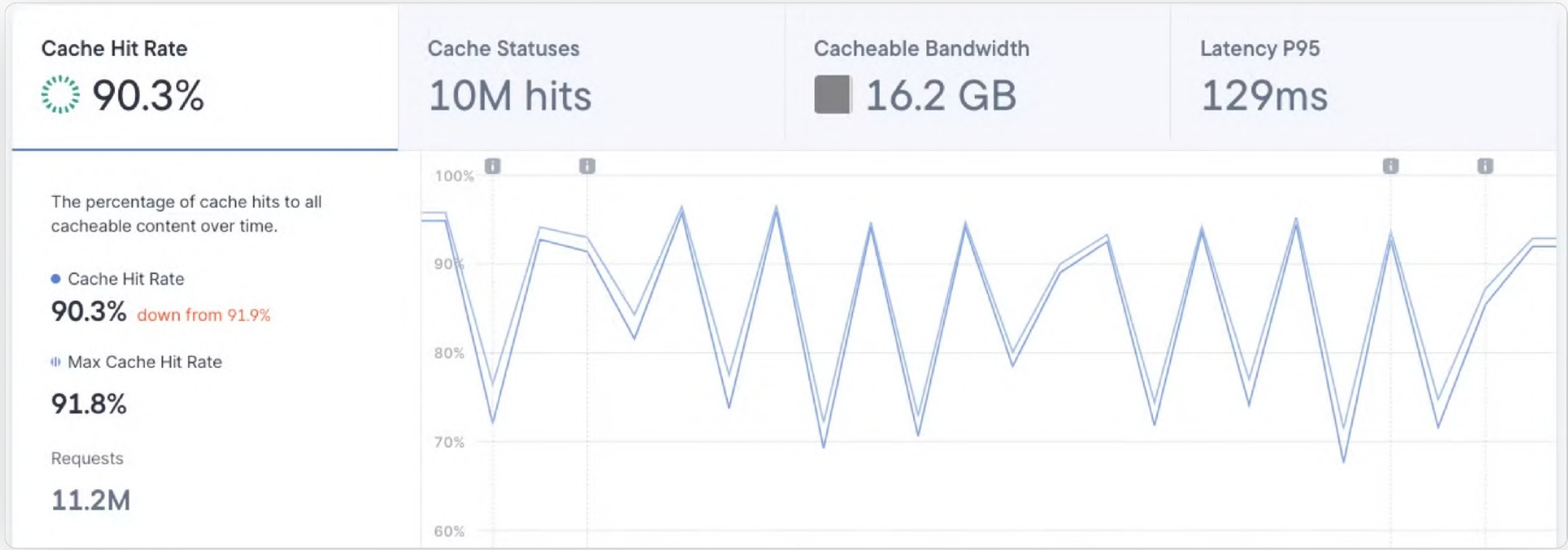
Stellate is a third-party provider that provides GraphQL Tooling:

- Caching
- Metrics
- Rate Limits





Cache Hit Rate



Lessons Learned

ERROR HANDLING | ABUSING | CACHING IS HARD | NAMING IS HARD

GQL Lessons Learned from a non-GQL pro

GQL Learnings:

- Error handling is hard
- Stand on the shoulder of giants (sorry Luc) → Have Design Guidelines
- Breaking changes → Mobile Apps
- Versioning
- n+1 Problem: Use Dataloaders
- You offer data → Frontend will use **all data** → Education

General Lessons Learned from a non-GQL pro

Public API Learnings

- Your API will be bombarded
- The issue is **always** the downstream system
- Polling is easier than realtime subscriptions
- Code will always be a mess → Constant improvements with every PR
- Caching is hard
- Naming is hard

Resources

You probably don't need GQL - Max Stoiber (Stellate)

200 OK! Error Handling in GraphQL

The Hidden Cost of GraphQL & NodeJS

I am done with GraphQL after 6 years

Introduction to Amazon API Gateway and AWS AppSync

Pitch

Want to make a presentation like this one?

Start with a fully customizable template, create a beautiful deck in minutes, then easily share it with anyone.

Create a presentation (It's free)