# FizzBuzz.md

# Docs

## Table of Contents



## Challenge Server API Reference

The FizzBuzz cli creates a challenge server that provides endpoints:

- for applicants to request to start a challenge
- to render your markdown file
- for applicants to submit questions about a challenge
- for applicants to submit their solutions

### `POST api/challenges/<id>/applications`

Request to start a challenge.

Request body:

```
{
  "email": "string"
}
```

Response body:

```
{
  "cookie": "fbmd_token=<token>",
  "challengeUrl": "https://makefizz.buzz/challenges/<id>",
  "docs": "https://makefizz.buzz/docs#get-challenge"
}
```

### `GET /challenges/<id>`

Set the token received from the **`POST /<id>/applications`** endpoint as a cookie value via browser dev tools to render your challenge.

The clock starts against **`timeLimitHours`** as soon as this endpoint is called.

### `POST api/challenges/<id>/queries`

Pass the token received from the **`POST /<id>/applications`** endpoint in an Authorization header to submit questions about the challenge.

```
Authorization: Bearer <token>
```

Request body:

```
{
  "query": "string"
}
```

The query will be sent to the support email address set in the front matter. Replies will be sent to the email sent in the POST **`/<id>/applications`**

request.

### `POST api/challenges/<id>/submissions`

Pass the token received from the `POST /<id>/applications` endpoint in an Authorization header to submit your solution.

Request body:

```
{
  "solutionLink": "string"
}
```

## Front Matter Reference

```
# Email address that will receive messages when applicants submit quest
# about the challenge via a POST to /api/<id>/queries
supportEmail: "support@fizzbuzz.md"

# Email address that will receive submissions via a POST to api/<id>/su
submissionEmail: "submission@fizzbuzz.md"

# Number of hours applicants have to submit their solution. Late submis
timeLimitHours: 72

# Whether the challenge is viewable w/o a session token generated from
public: true

# Email address that will receive rejection emails
rejectionEmail: >
  Thanks for applying. Unfortunately, we've decided to move forward wit

# Email address that will receive acceptance emails
acceptanceEmail: >
  Congratulations! We're moving forward with your application. Please b
```

```
# Company name that will be used in the submission email response
companyName: "FizzBuzz.md"
```

## Usage in Slack

An easy way to set up a challenge so that multiple team members can respond to queries and solutions is to use **Slack's email integration feature**. If you create separate channels for queries and solutions (e.g., `#challenge-queries` and `challenge-submissions`), you can have separate emails for the `supportEmail` and `submissionEmail` front matter fields.

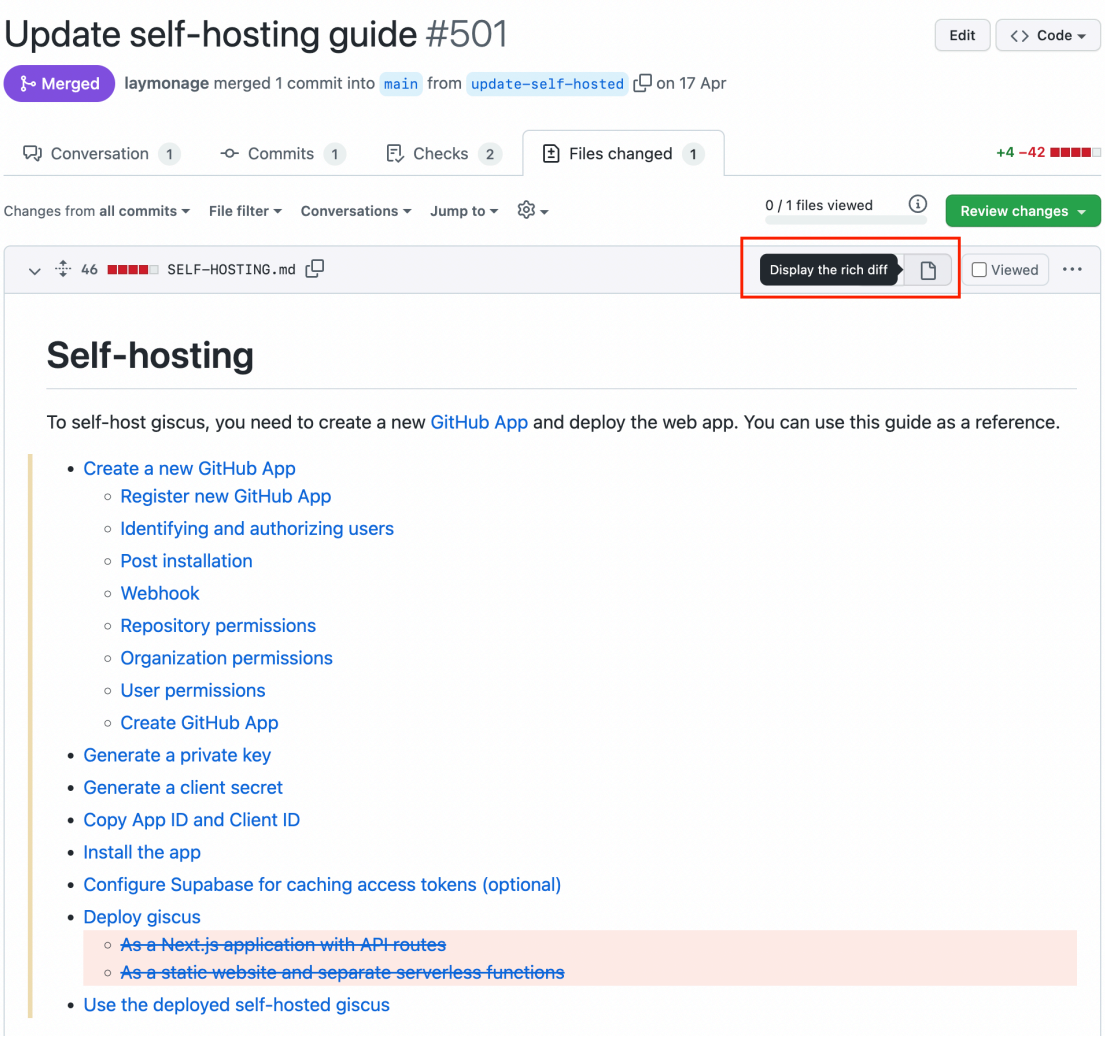
## Usage in Github Actions

If you'd like to update your challenge file in a Github Actions workflow, you can use download the `fizzbuzz.md` cli and use it easily in a github action:

```
steps:
  - name: Update challenge file
    env:
      FBMD_TOKEN: ${{ secrets.FBMD_TOKEN }}
      FBMD_EMAIL: ${{ secrets.FBMD_EMAIL }}
    # npx is intaled by default on runners
    # https://github.blog/changelog/2023-09-22-github-actions-transitio
    run: npx zx https://makefizz.buzz/cli <Challenge-File>
```

## Usage in Github PRs

Reviewing challenges is recommended. Apparently, Github lets you review rich renderings of markdown files in PRs:


Update self-hosting guide #501
Edit
Code
Merged
laymonage merged 1 commit into main from update-self-hosted on 17 Apr
Conversation 1
Commits 1
Checks 2
Files changed 1
+4 −42
Changes from all commits
File filter
Conversations
Jump to
0 / 1 files viewed
Review changes
46
SELF-HOSTING.md
Display the rich diff
Viewed
Self-hosting
To self-host giscus, you need to create a new GitHub App and deploy the web app. You can use this guide as a reference.
Create a new GitHub App
Register new GitHub App
Identifying and authorizing users
Post installation
Webhook
Repository permissions
Organization permissions
User permissions
Create GitHub App
Generate a private key
Generate a client secret
Copy App ID and Client ID
Install the app
Configure Supabase for caching access tokens (optional)
Deploy giscus
As a Next.js application with API routes
As a static website and separate serverless functions
Use the deployed self-hosted giscus