Every millisecond matters

Lukáš Huvar

Lukáš Huvar

Software engineer productboard

huv1k huv1k https://huvik.dev/

Previously Vercel Prisma

Agenda

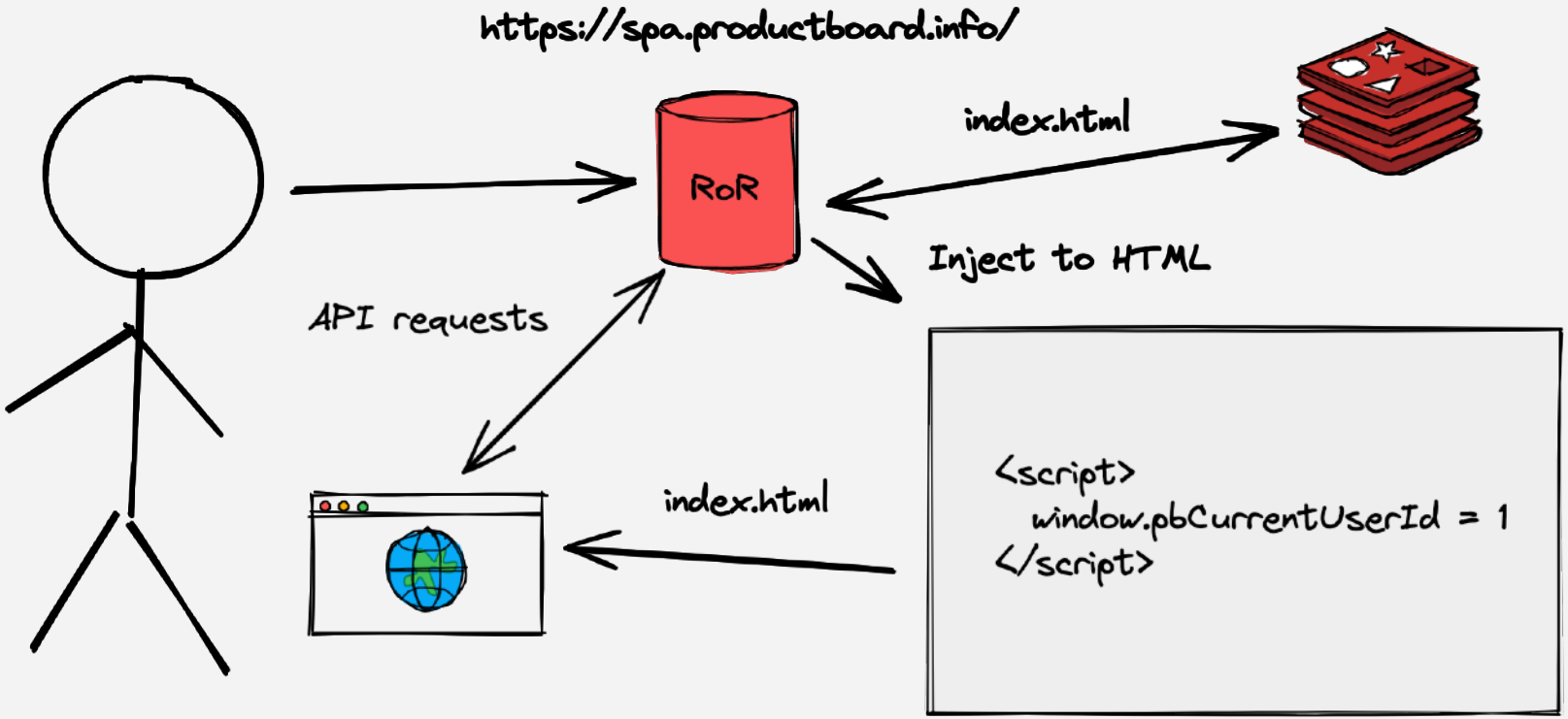
1. Business problem 👩
2. First **MVP** solution 🤞
3. Cloudflare workers 👷

Business problem 

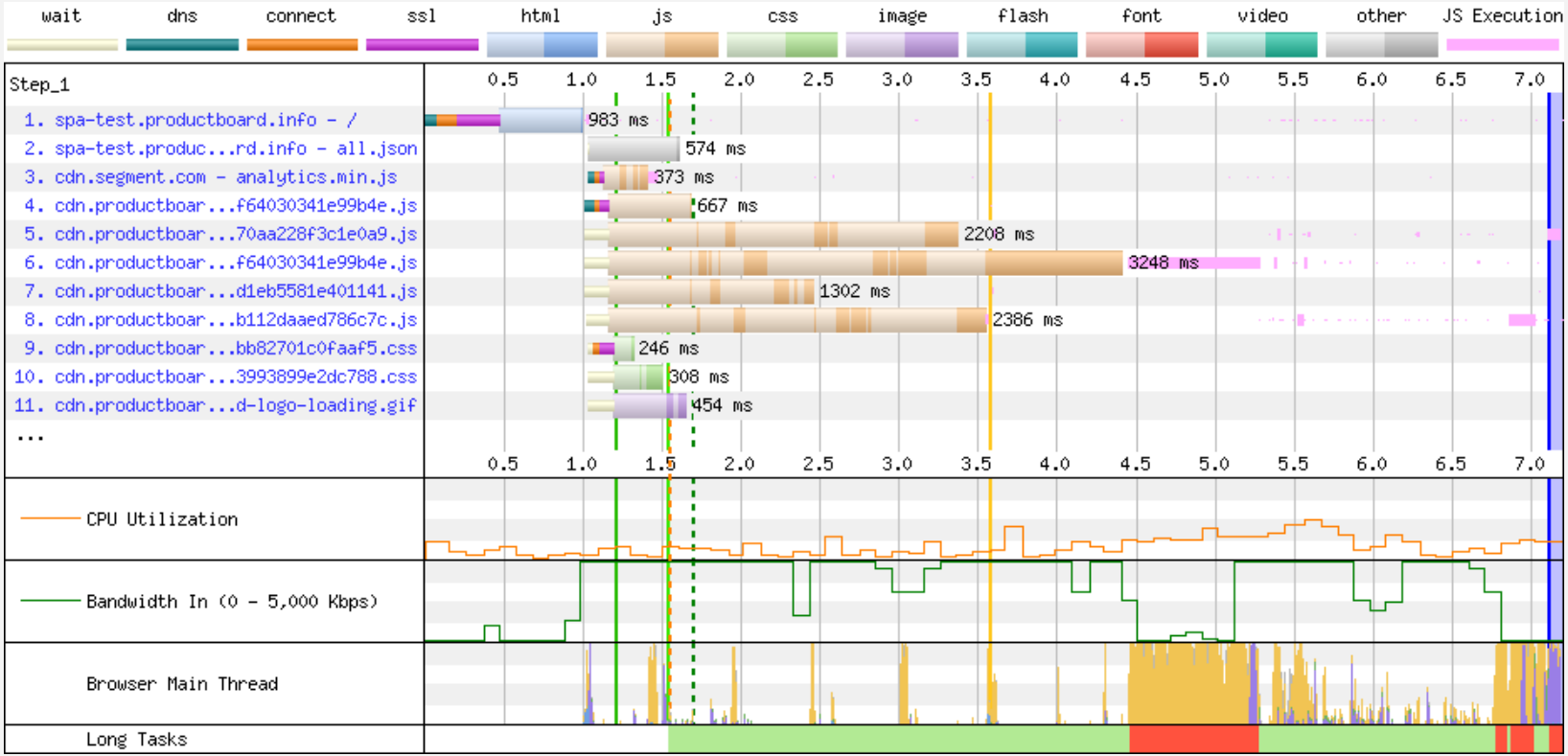
Let’s improve loading performance 🤯

- It should affect **all** customers
- Every team should focus on a different topic

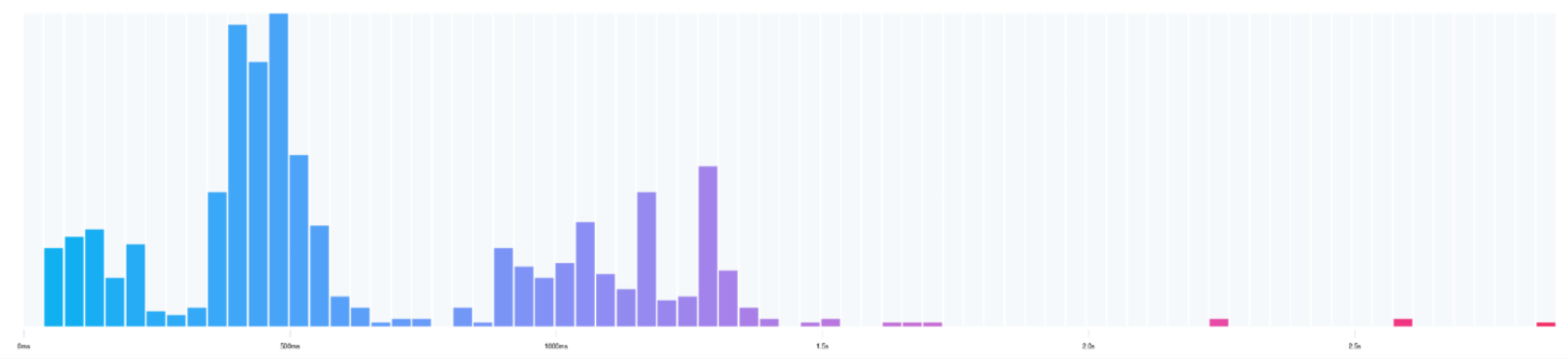
Previous setup



Where to focus?



Time to first byte before



RFC - “Remove server side rendering”

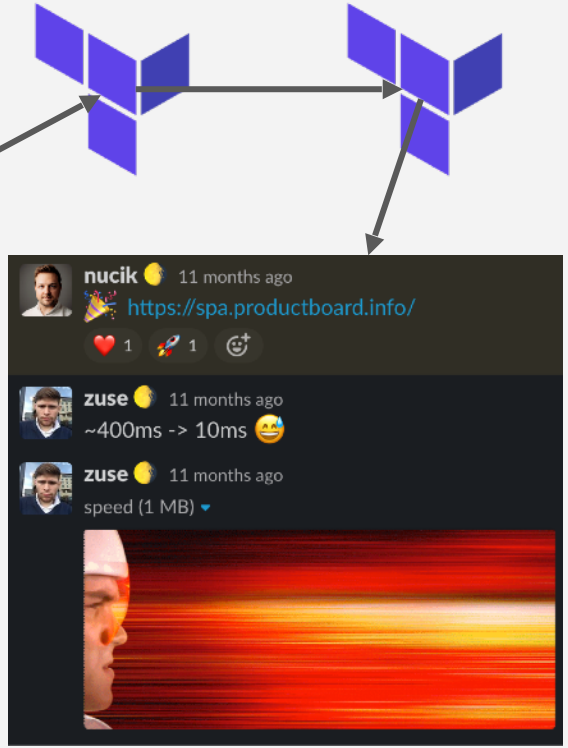
- The classic process around bigger Productboard initiatives
- Covered most problematic parts and how to solve them
- This wasn’t the best name of RFC 😅

First MVP solution 🤞

It’s easy right? Just deploy one static file to CDN

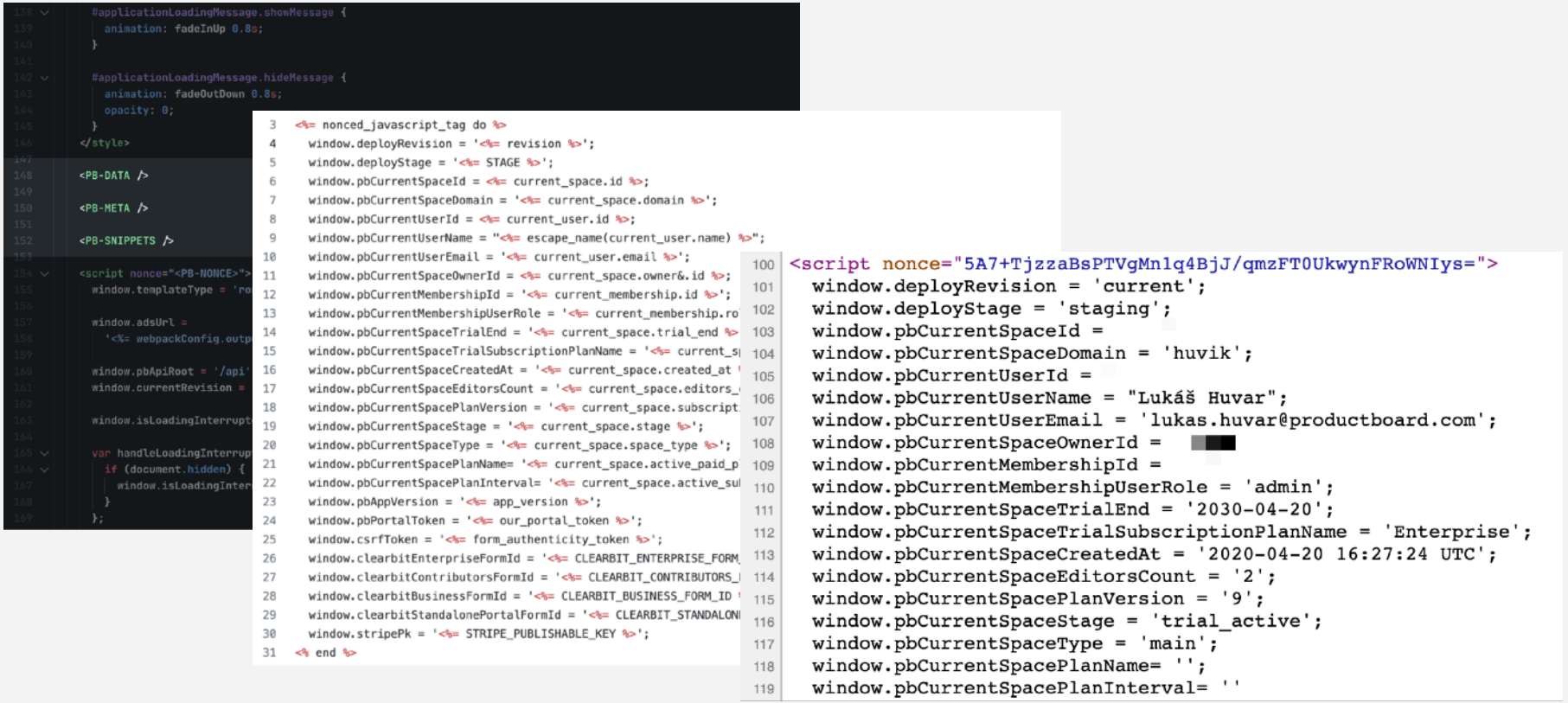
Let’s implement it

```hcl
module "pb-staging-spa" {
  source  = "terraform-aws-modules/s3-bucket/aws"
  version = "1.12.0"

  bucket = "spa-staging.productboard.net"
  acl    = "public-read"

  # Never change the value of this field ⚠
  force_destroy = false

  versioning = {
    enabled = true
  }

  website = {
    index_document = "index.html"
  }

  tags = {
    Team = "Frontend"
  }
}
```



It should work right?

Change how we load injected data



```
#applicationLoadingMessage.showMessage {
  animation: fadeInUp 0.8s;
}

#applicationLoadingMessage.hideMessage {
  animation: fadeOutDown 0.8s;
  opacity: 0;
}
</style>

<PB-DATA />

<PB-META />

<PB-SNIPPETS />

<script nonce="<PB-NONCE>">
  window.templateType = 'ro

  window.adsUrl =
    '<%= webpackConfig.outp

  window.pbApiRoot = '/api'
  window.currentRevision =

  window.isLoadingInterrupt

  var handleLoadingInterrup
    if (document.hidden) {
      window.isLoadingInter
    }
  };
```

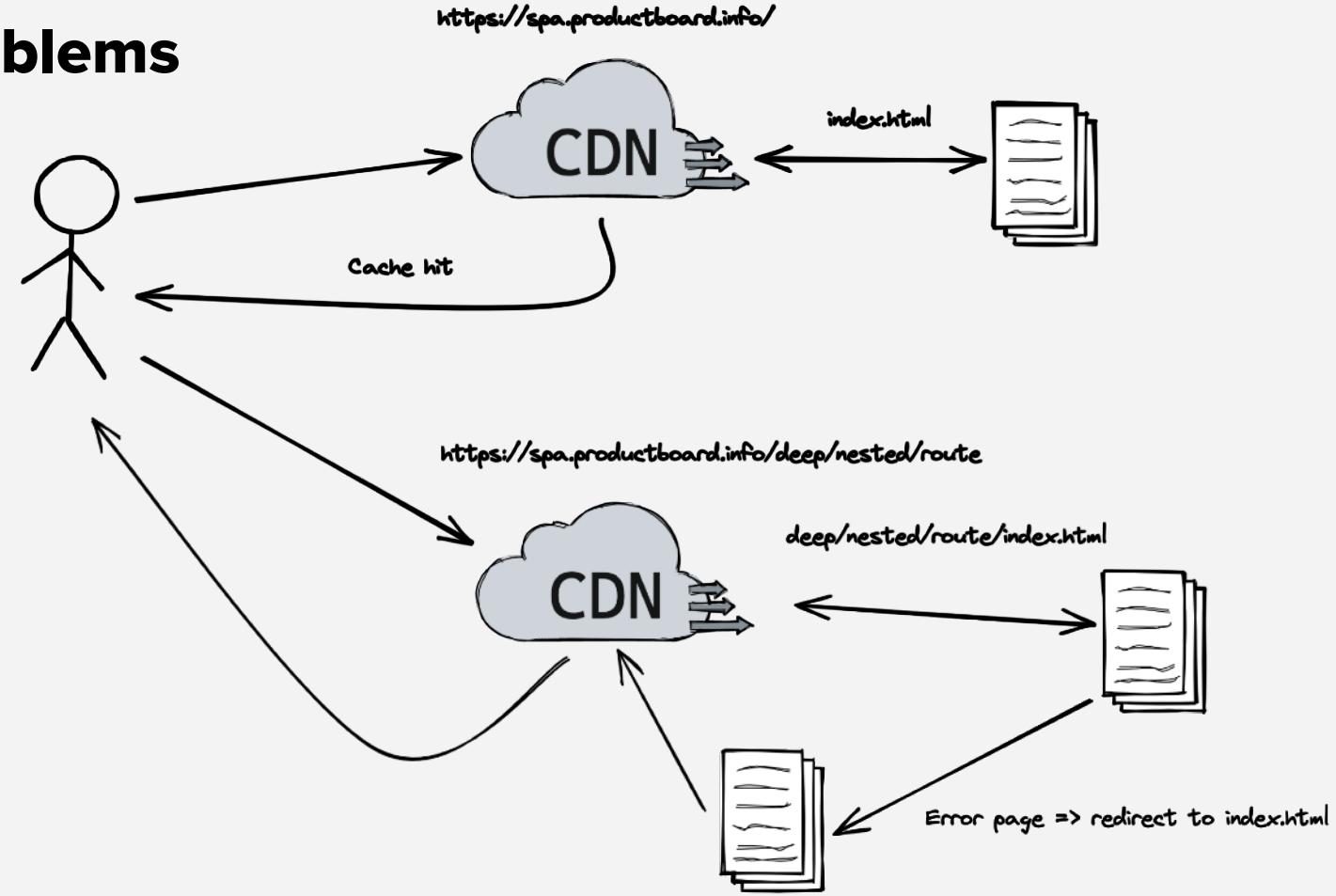
```
<%= nonced_javascript_tag do %>
  window.deployRevision = '<%= revision %>';
  window.deployStage = '<%= STAGE %>';
  window.pbCurrentSpaceId = <%= current_space.id %>;
  window.pbCurrentSpaceDomain = '<%= current_space.domain %>';
  window.pbCurrentUserId = <%= current_user.id %>;
  window.pbCurrentUserName = "<%= escape_name(current_user.name) %>";
  window.pbCurrentUserEmail = '<%= current_user.email %>';
  window.pbCurrentSpaceOwnerId = <%= current_space.owner&.id %>;
  window.pbCurrentMembershipId = '<%= current_membership.id %>';
  window.pbCurrentMembershipUserRole = '<%= current_membership.ro
  window.pbCurrentSpaceTrialEnd = '<%= current_space.trial_end %>
  window.pbCurrentSpaceTrialSubscriptionPlanName = '<%= current_s
  window.pbCurrentSpaceCreatedAt = '<%= current_space.created_at
  window.pbCurrentSpaceEditorsCount = '<%= current_space.editors_
  window.pbCurrentSpacePlanVersion = '<%= current_space.subscript
  window.pbCurrentSpaceStage = '<%= current_space.stage %>';
  window.pbCurrentSpaceType = '<%= current_space.space_type %>';
  window.pbCurrentSpacePlanName= '<%= current_space.active_paid_p
  window.pbCurrentSpacePlanInterval= '<%= current_space.active_su
  window.pbAppVersion = '<%= app_version %>';
  window.pbPortalToken = '<%= our_portal_token %>';
  window.csrfToken = '<%= form_authenticity_token %>';
  window.clearbitEnterpriseFormId = '<%= CLEARBIT_ENTERPRISE_FORM_
  window.clearbitContributorsFormId = '<%= CLEARBIT_CONTRIBUTORS_
  window.clearbitBusinessFormId = '<%= CLEARBIT_BUSINESS_FORM_ID
  window.clearbitStandalonePortalFormId = '<%= CLEARBIT_STANDALON
  window.stripePk = '<%= STRIPE_PUBLISHABLE_KEY %>';
<% end %>
```

```
<script nonce="5A7+TjzzaBsPTVgMn1q4BjJ/qmzFT0UkwynFRoWNIys=">
  window.deployRevision = 'current';
  window.deployStage = 'staging';
  window.pbCurrentSpaceId =
  window.pbCurrentSpaceDomain = 'huvik';
  window.pbCurrentUserId =
  window.pbCurrentUserName = "Lukáš Huvar";
  window.pbCurrentUserEmail = 'lukas.huvar@productboard.com';
  window.pbCurrentSpaceOwnerId =
  window.pbCurrentMembershipId =
  window.pbCurrentMembershipUserRole = 'admin';
  window.pbCurrentSpaceTrialEnd = '2030-04-20';
  window.pbCurrentSpaceTrialSubscriptionPlanName = 'Enterprise';
  window.pbCurrentSpaceCreatedAt = '2020-04-20 16:27:24 UTC';
  window.pbCurrentSpaceEditorsCount = '2';
  window.pbCurrentSpacePlanVersion = '9';
  window.pbCurrentSpaceStage = 'trial_active';
  window.pbCurrentSpaceType = 'main';
  window.pbCurrentSpacePlanName= '';
  window.pbCurrentSpacePlanInterval= ''
```

Security 👮

- We need to reimplement **CSP** headers from the **backend**
- Move **CSP** to frontend and use csp-html-webpack-plugin
- Inline **nonce** hashes are automatically generated
- Change headers to **meta** tags

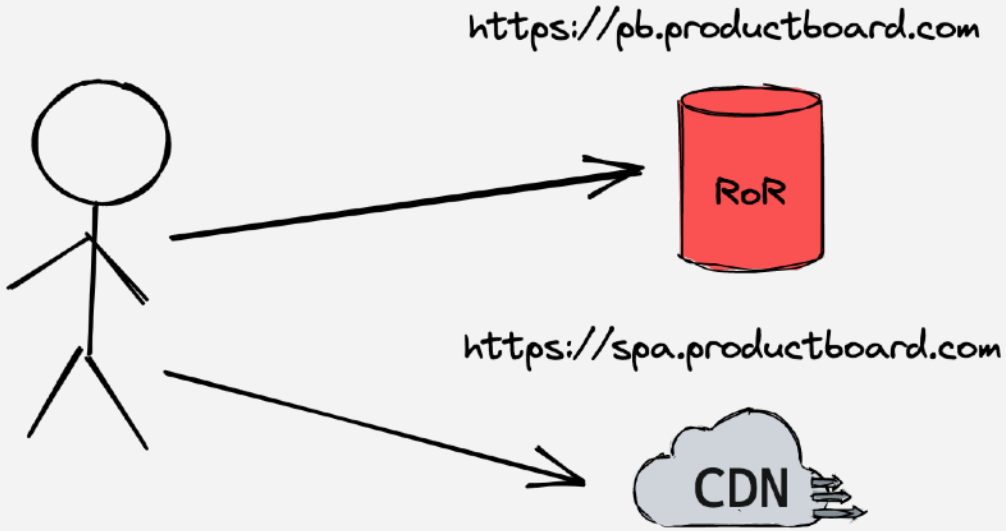
Caching problems



Revision system

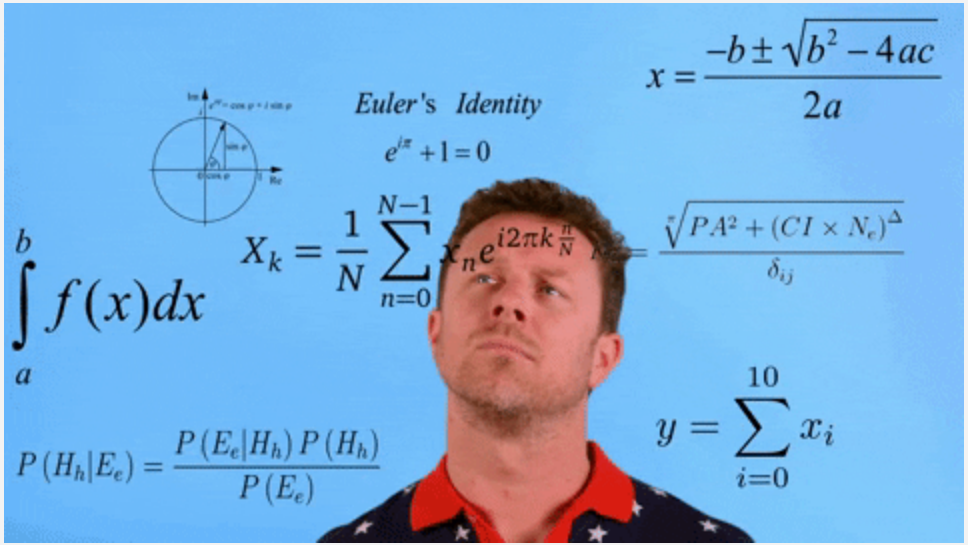
test.productboard.info/insights/trends/all-notes?rev=branch%2Ffeat%2Fdisable-jsx-handler-names

test.productboard.info/insights/trends/all-notes?rev=002f53f96acfbeb8c97dc1076fafd8d6bd31d400



What now?

- We need a **hybrid solution** that has **runtime** and **caching possibility**
- There are **multiple** solutions on the market

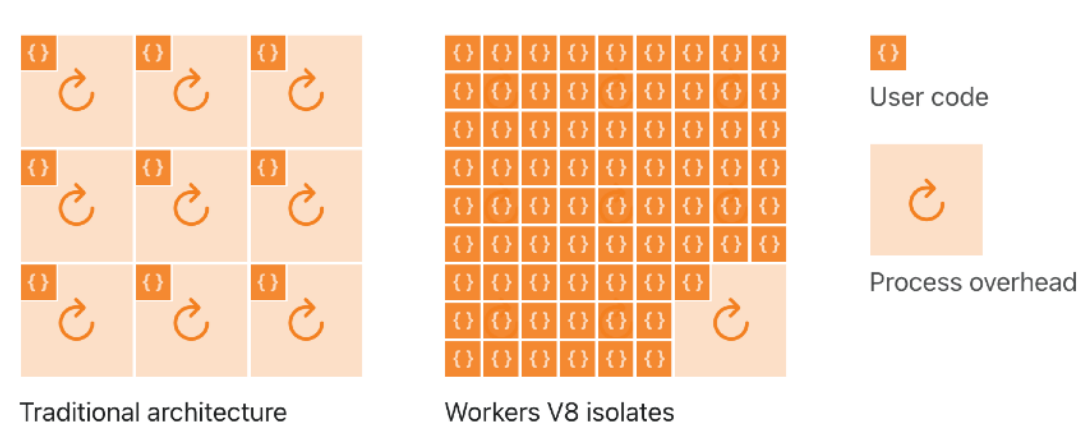


Possible solutions

Cloudflare workers 

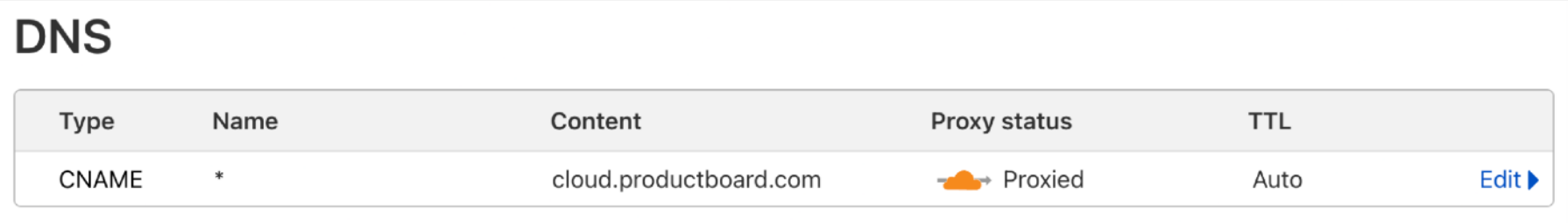
Why Cloudflare workers?

- Serverless platform running on V8 isolates

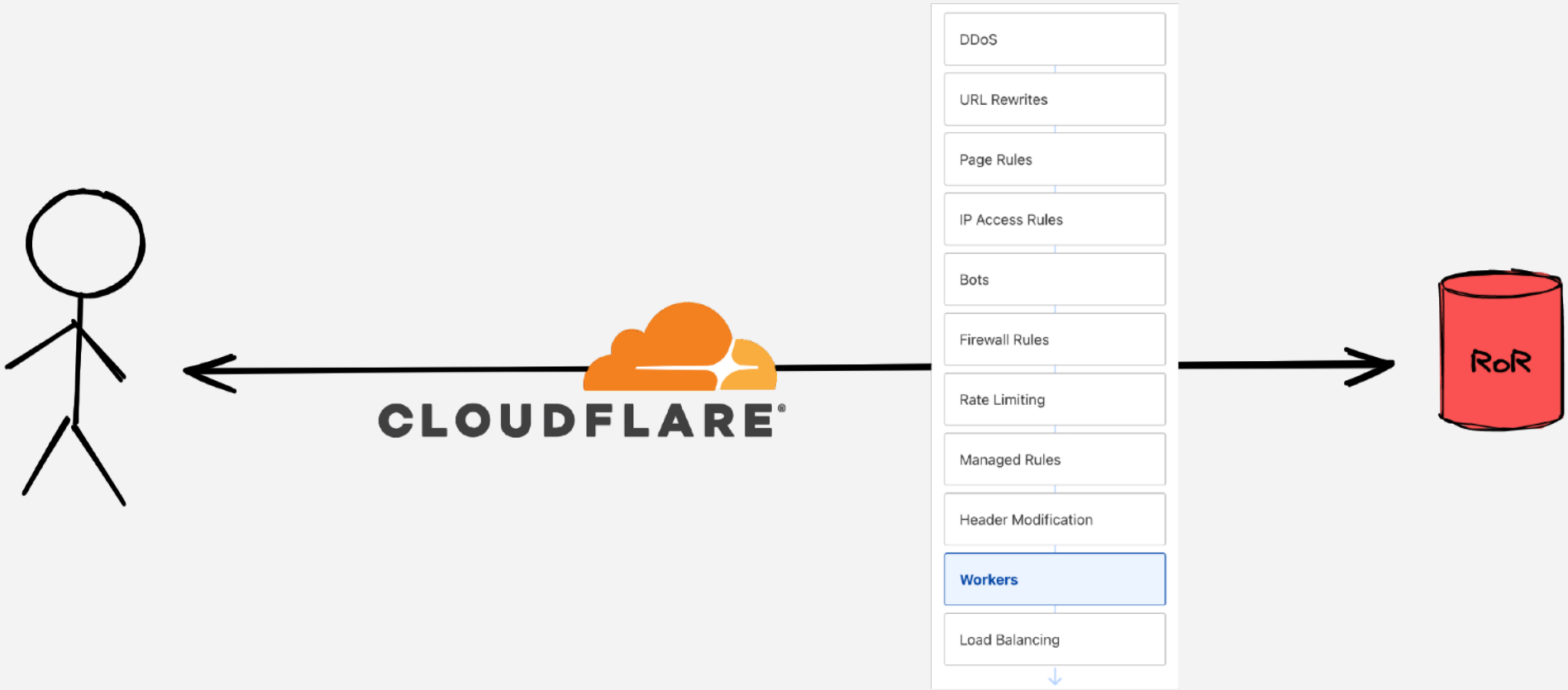


- Can run Javascript and languages compiled to WebAssembly
- No cold starts, handled during TLS handshake
- Buying more into the Cloudflare ecosystem

What does it mean to enable workers?



But what it actually means?



Matching behaviour

```
https://*.example.com/images/*
```

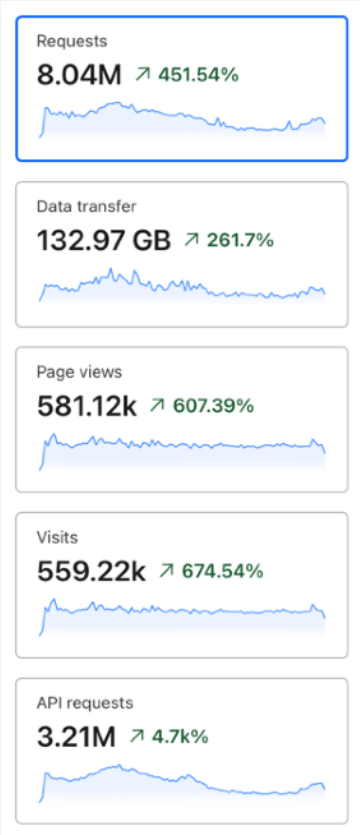
```
*example.com/images/cat.png -> <no script>
*example.com/images/*       -> worker-script
```

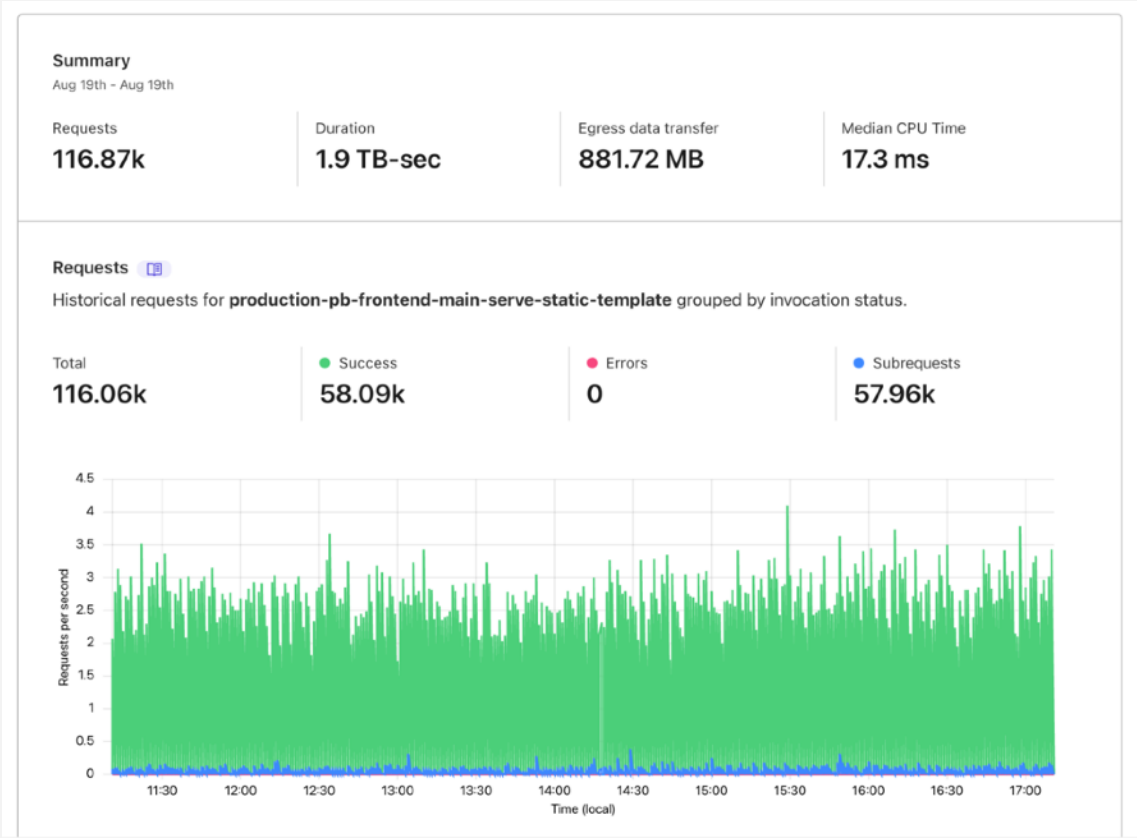
```
name = "production-pb-frontend-main-serve-static-template"
route = "https://*.productboard.com/*"
vars = {ENVIRONMENT = "production"}
```

Routes whitelisting

```
 resource "cloudflare_worker_route" "bypass-api" {
 resource "cloudflare_worker_route" "bypass-gw" {
resource "cloudflare_worker_route" "bypass-integrations" {
  zone_id = data.cloudflare_zones.productboard-com.zones[0].id
  pattern = "https://*.productboard.com/integrations/*"
}
```

Cloudflare proxy enabled





Revision system

Key	Value	
001b0e3faa396082c460eea69e9fe8fb57b731c6	<!DOCTYPE html><html lang="en" prefix="og: https://ogp.me/ns#"><head><meta http-equiv="Content-Security-Poli...	View •••
002c44621e9cc03e1bcc6870d81519735b2fd9cf	<!DOCTYPE html><html lang="en" prefix="og: https://ogp.me/ns#"><head><meta http-equiv="Content-Security-Poli...	View •••
003d59b37941d99ce731dc12e07218ee72b4401f	<!DOCTYPE html><html lang="en" prefix="og: https://ogp.me/ns#"><head><meta http-equiv="Content-Security-Poli...	View •••
004946ca2cda84b34249217e0396c0b0d05f0c8d	<!DOCTYPE html><html lang="en" prefix="og: https://ogp.me/ns#"><head><meta http-equiv="Content-Security-Poli...	View •••

```
1  resource "cloudflare_workers_kv_namespace" "production" {
2    title = "production-templates"
3  }
4
```

- **:current** - most recent revision template
- **:revision** - current revision hash
- **:version** - deployed worker version

```
addEventListener('fetch', (event) => {
  event.respondWith(handleEvent(event))
})

async function handleEvent(event) {
  const { rev } = parseUrl(event.request.url)

  // Check if we hit cache
  const cache = caches.default
  let response = await cache.match(event.request)

  if (!response) {
    const revisionTemplate = rev ? await TEMPLATES.get(rev) : null

    const staticTemplate = revisionTemplate
      ? revisionTemplate
      : await TEMPLATES.get(':current')

    response = new Response(staticTemplate)

    // Save response to cache
    event.waitUntil(cache.put(event.request, response.clone()))
  }

  return response
}
```

Caching requirements

- We want to cache all request to **one response**
- Skip cache for revisions
- Use web standards like **ETags**
- Users should always hit hot cache

Caching all requests

```
addEventListener('fetch', (event) => {
  event.respondWith(handleEvent(event))
})

async function handleEvent(event) {
  const { rev } = parseUrl(event.request.url)

  // Check if we hit cache
  const cache = caches.default
  const cacheKey = new Request(
    `https://pb.productboard.com/_main_cache/${TEMPLATE_REVISION}`,
    event.request,
  );
  let response = await cache.match(cacheKey)

  if (!response) {
    const revisionTemplate = rev ? await TEMPLATES.get(rev) : null

    const staticTemplate = revisionTemplate
      ? revisionTemplate
      : await TEMPLATES.get(':current')

    response = new Response(staticTemplate)

    // Save response to cache
    event.waitUntil(cache.put(cacheKey, response.clone()));
  }

  return response
}
```

Skip cache for revision

```
addEventListener('fetch', (event) => {
  event.respondWith(handleEvent(event))
})

async function handleEvent(event) {
  const { rev } = parseUrl(event.request.url)

  // Check if we hit cache
  const cache = caches.default
  const cacheKey = new Request(
    `https://pb.productboard.com/_main_cache/${TEMPLATE_REVISION}`,
    event.request,
  );
  let response = rev ? null : await cache.match(cacheKey)

  if (!response) {
    const revisionTemplate = rev ? await TEMPLATES.get(rev) : null

    const staticTemplate = revisionTemplate
      ? revisionTemplate
      : await TEMPLATES.get(':current')

    response = new Response(staticTemplate)

    // Save response to cache
    if(!rev) {
      event.waitUntil(cache.put(cacheKey, response.clone()))
    }
  }

  return response
}
```

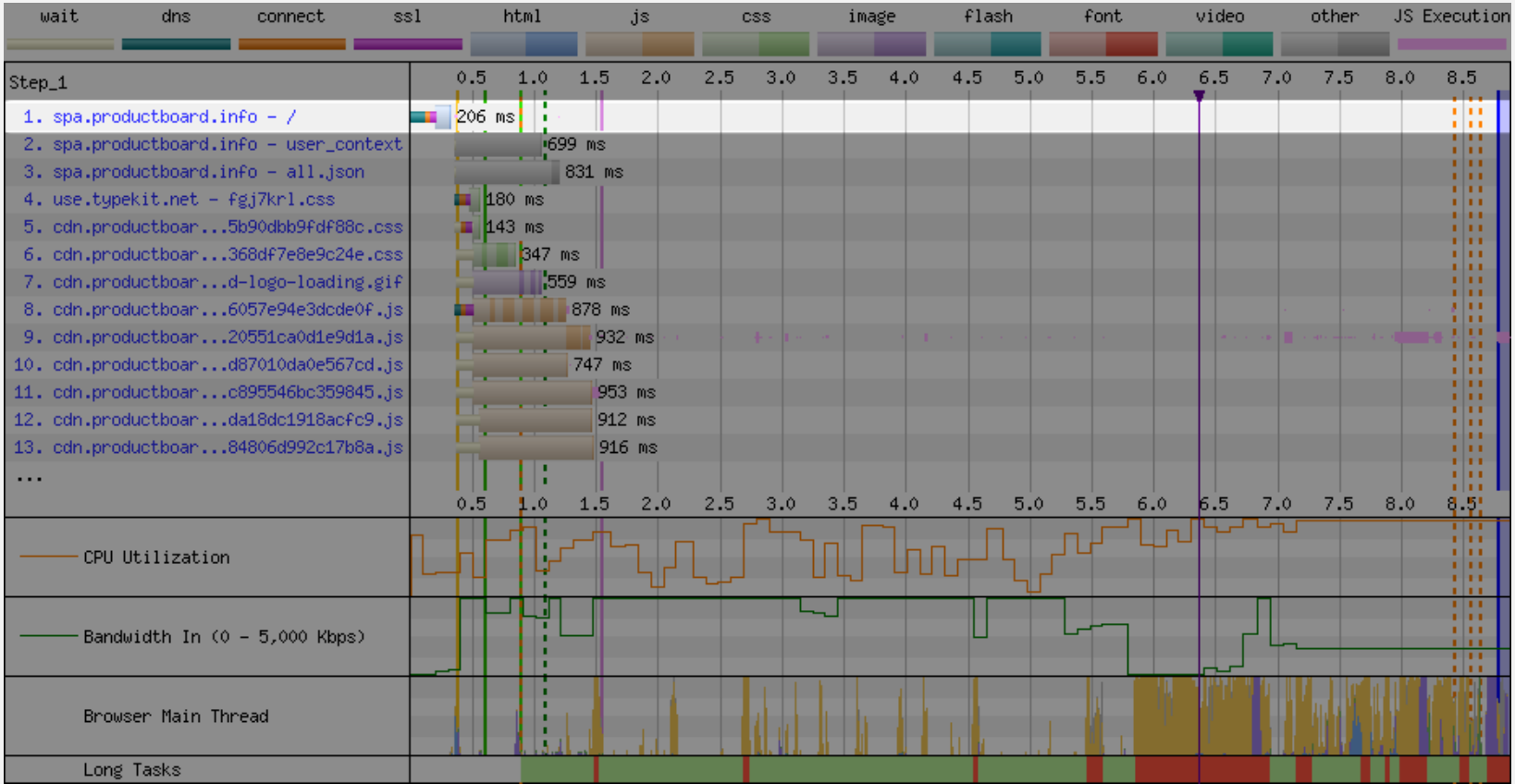
Using ETags for

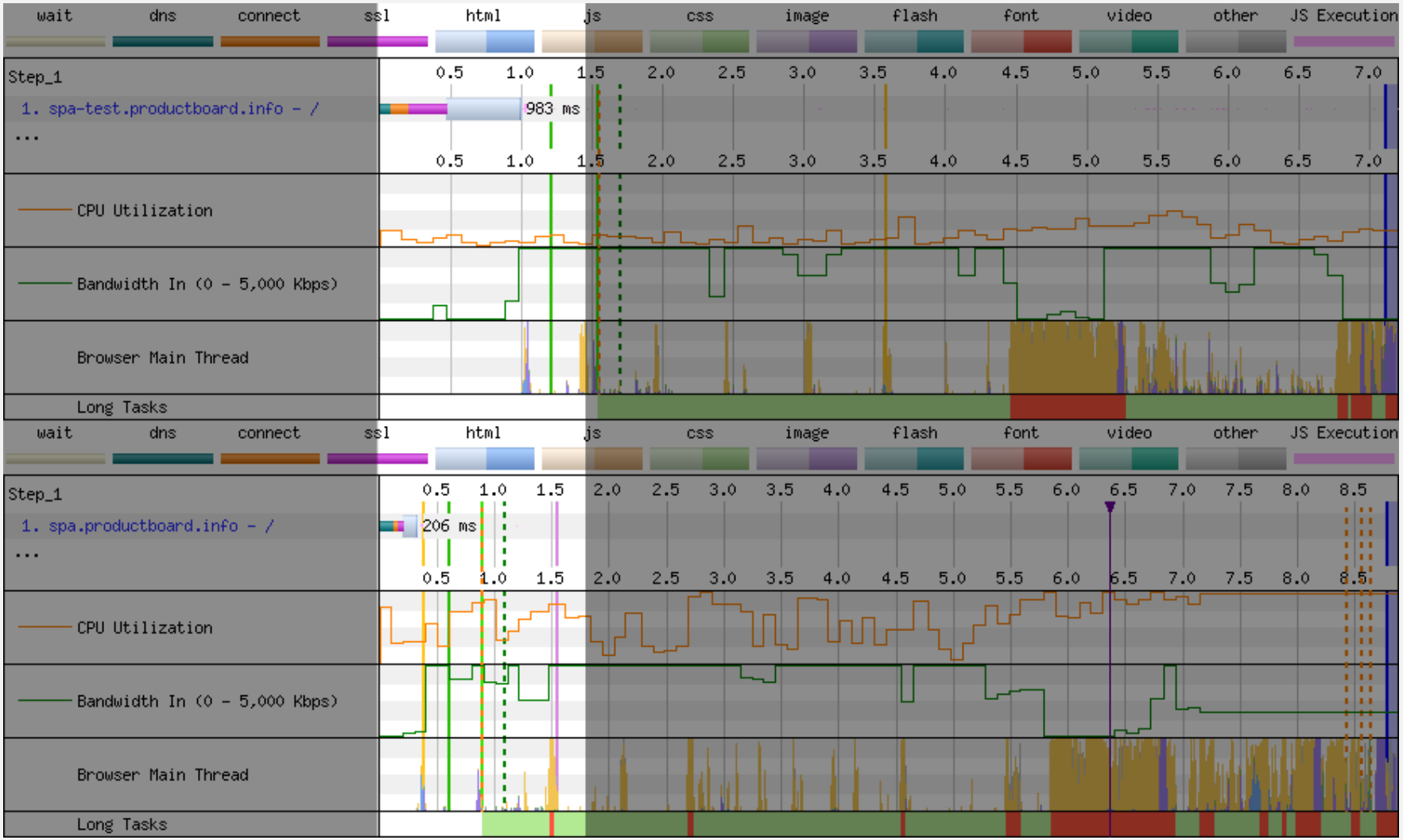
```
addEventListener('fetch', (event) => {
  event.respondWith(handleEvent(event))
})

async function handleEvent(event) {
  const { rev } = parseUrl(event.request.url)
  const ifNoneMatch = event.request.headers.get('if-none-match')

  // Check if we hit cache
  const cache = caches.default
  const cacheKey = new Request(
    `https://pb.productboard.com/_main_cache/${TEMPLATE_REVISION}`,
    event.request
  )
  let response = rev ? null : await cache.match(cacheKey)

  if (!response) {
    const revisionTemplate = rev ? await TEMPLATES.get(rev) : null

    const staticTemplate = revisionTemplate
      ? revisionTemplate
      : await TEMPLATES.get(':current')

    response = new Response(staticTemplate, {
      headers: {
        etag: formatEtag(TEMPLATE_REVISION),
      },
    })

    // Save response to cache
    if (!rev) {
      event.waitUntil(cache.put(cacheKey, response.clone()))
    }
  } else {
    if (ifNoneMatch === formatEtag(TEMPLATE_REVISION)) {
      response = new Response(null, {
        status: 304,
        statusText: 'Not Modified',
      })
    }
  }

  return response
}

```

Warming cache

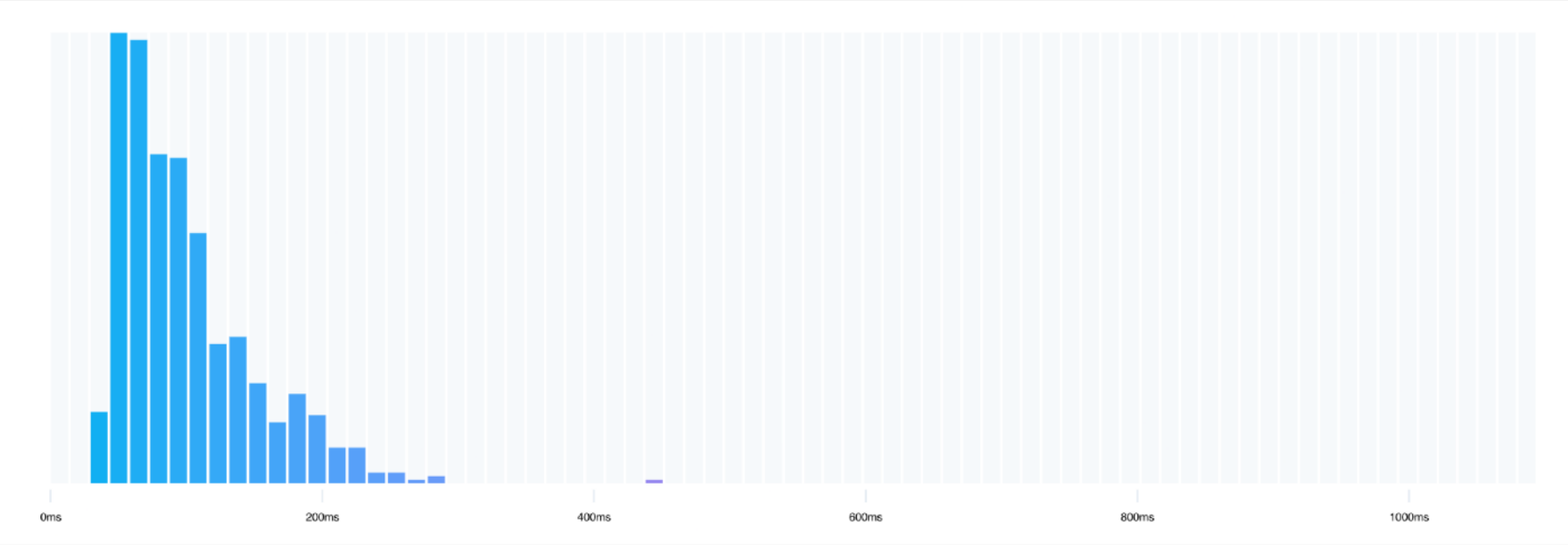
```
$ curl https://pb.productboard.com/
```

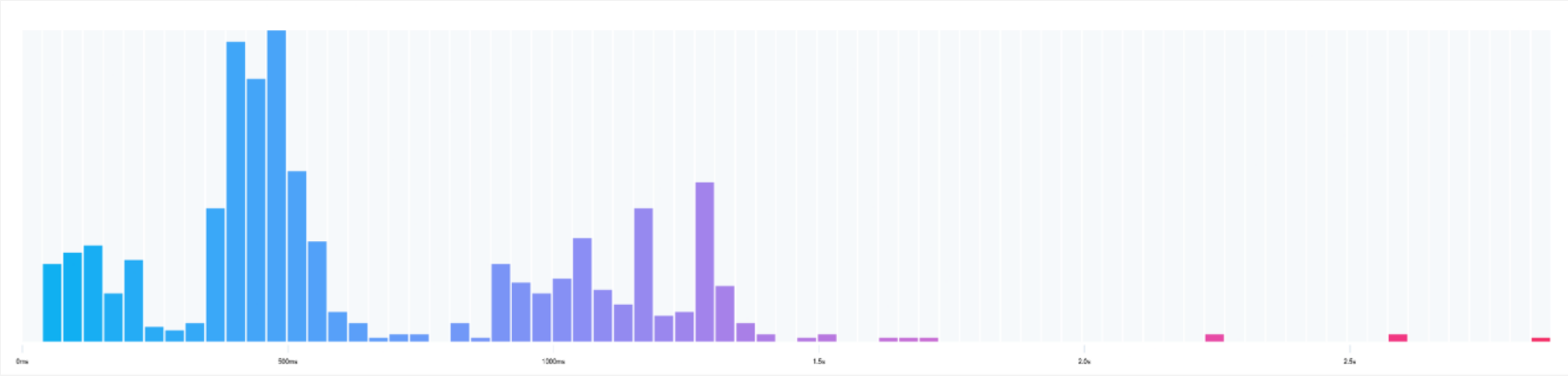
New timeline

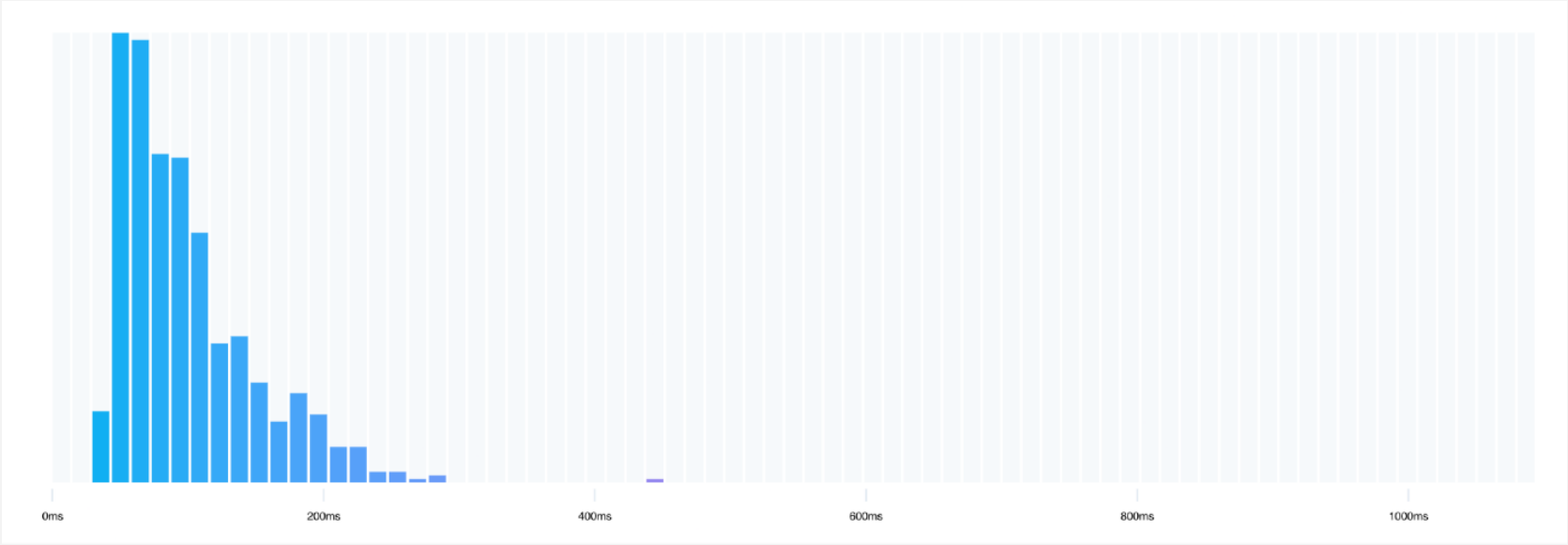



wait
dns
connect
ssl
html
js
css
image
flash
font
video
other
JS Execution
Step_1
1. spa-test.productboard.info - /
...
983 ms
0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5 6.0 6.5 7.0
CPU Utilization
Bandwidth In (0 - 5,000 Kbps)
Browser Main Thread
Long Tasks
wait
dns
connect
ssl
html
js
css
image
flash
font
video
other
JS Execution
Step_1
1. spa.productboard.info - /
...
206 ms
0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5 6.0 6.5 7.0 7.5 8.0 8.5
CPU Utilization
Bandwidth In (0 - 5,000 Kbps)
Browser Main Thread
Long Tasks

Time to first byte after




0ms
500ms
1000ms
1.5s
2.0s
2.5s


0ms
200ms
400ms
600ms
800ms
1000ms

Learnings

- Observability is key
- Changing whole network infrastructure is not easy
- Computing on the edge is future

Summary

- We reduced **TTFB** on **average by 80%**
- New serverless platform in **Productboard**
- Buy more into the **Cloudflare** ecosystem
- Great improvement in delivering new features using Cloudflare workers

Q & A

Lukáš Huvar

Software engineer productboard

huv1k huv1k https://huvik.dev/