



# John Paz

Señor Content Design



## Content Design Manager Portfolio

Updated January 2022



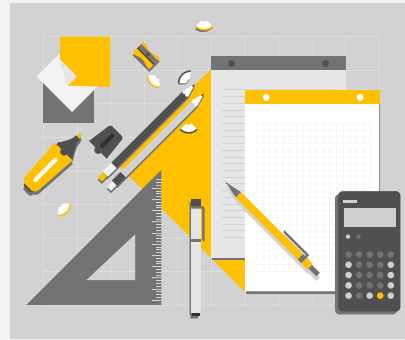
# About Me

# About Me



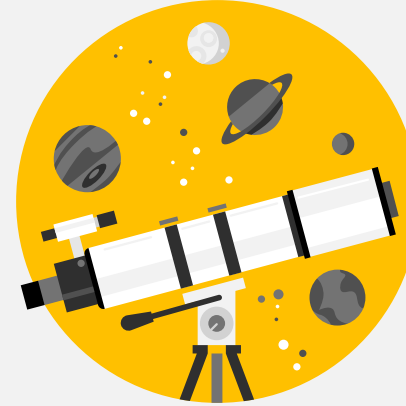
## **2+ years in leadership roles**

I embrace mentorship as mutually beneficial, and practice service leadership.



## **15 years of experience**

UI copy reviews, user research, and strategic design thinking.



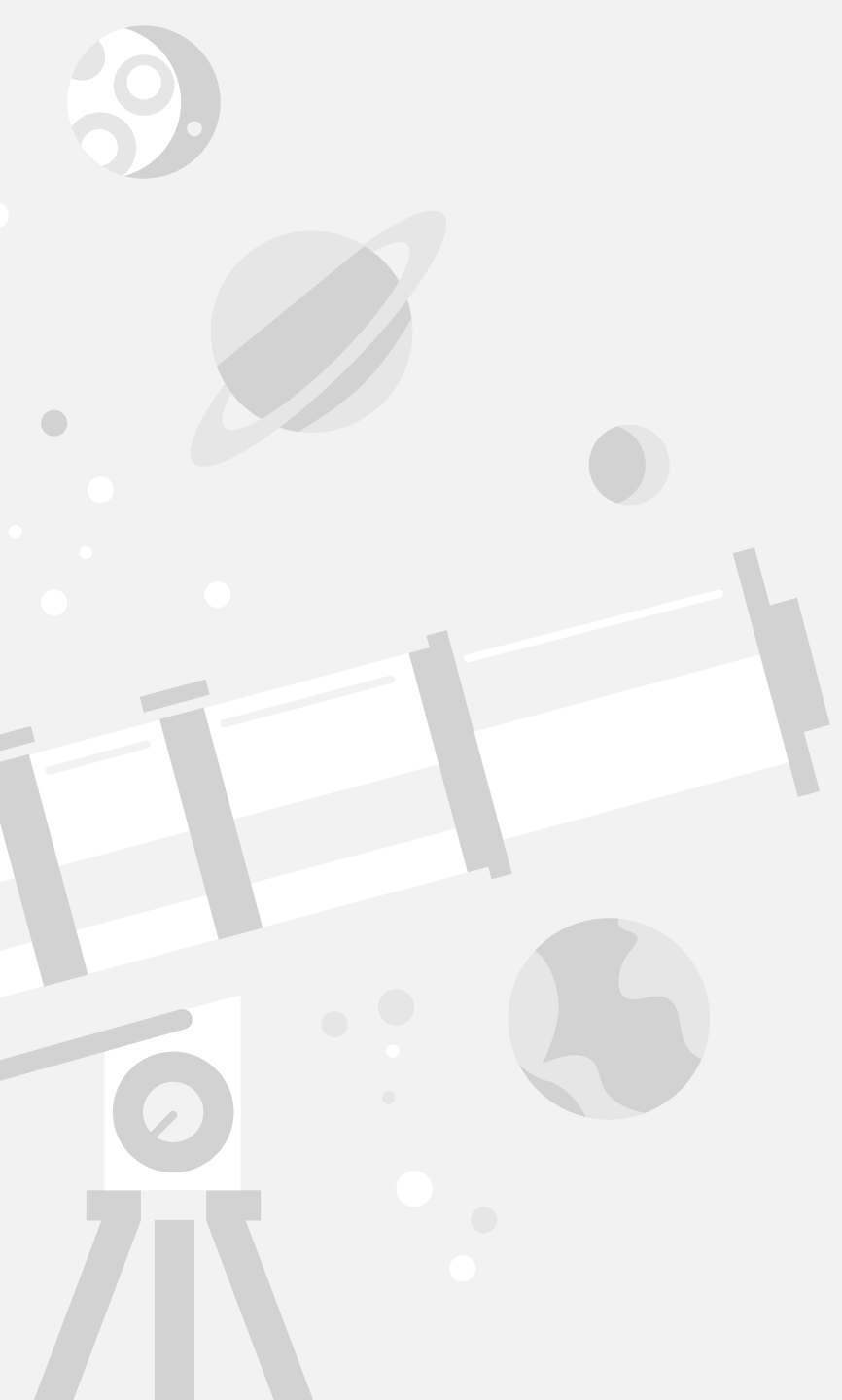
## **Great in Agile teams on the product triad**

I love getting involved early in the dev lifecycle (I'm also a Scrum Master).



## **Powerful storyteller and public speaker**

I rely on my diverse work/life experiences and storytelling ability to advocate for users.



# Work Examples



# UX Writing

## Menlo Connect Installer

2021

#### FORMAT/TOOLS

- Google Docs
- Figma
- Snag-It
- Pendo



#### UX WRITING

#### USER RESEARCH

#### DESIGN THINKING

#### DOCUMENTATION

# UX Writing Example

## Menlo Connect Installer

### Project Overview

**My contribution:** I reviewed this screen (and the others in the flow) for clarity, and I insisted on changing the name of a new enhancement for better accuracy and clarity.

### Project Goals

Add the Tamper Proofing/Uninstall Protection feature to the Menlo Connect installer and update documentation.




#### Review the interface to improve copy and flow

**Review the design mockups.** Before the installer was ready to test, review the designs in Figma and provide feedback.

**Suggest improvements to nomenclature.** What components are referred to felt problematic, make it better.

**Rewrite field labels and descriptions.** Once installer is ready for testing, propose changes to the interface.

### Team

- 1  Writer (Me)
- 2  Product Managers
- 4  Engineers (contractors)

1  Support Engineer

### Project Duration

21 days

# UX Writing Example

## Menlo Connect Installer

### FORMAT/TOOLS

- Google Docs
- Figma
- Snag-It
- Pendo



### UX WRITING

### USER RESEARCH

### DESIGN THINKING

### DOCUMENTATION

### Problem statement

The Menlo Connect installer previously did not have an interface for setting a password to prevent users from uninstalling the program.

The team did not have a writer on staff to review the screens during initial iterations. Some of the names for features and fields had potential to mislead users or cause unnecessary support burden.

### Outcomes

#### Changed Tamper-proofing to Uninstall protection

- The name felt misleading; the device could still be tampered with in other ways.
- “Uninstall Protection” is more descriptive and easier to understand.

#### Rewrite field labels and descriptions

- Kept the focus of this screen in mind – helping admins set an uninstall password – and made suggestions accordingly.
- Warned users about the consequences of losing the password (but don’t scare them), link to details about password recovery.

#### Establish password recovery procedures

- Made support aware of the changes to the screens and addition of feature.
- Helped document the internal password recovery knowledge base article.

## FORMAT/TOOLS

- Google Docs
- Figma
- Snag-It
- Pendo



## UX WRITING

## USER RESEARCH

## DESIGN THINKING

## DOCUMENTATION

# UX Writing Example

## Menlo Connect Installer

### Outcomes

#### Original

Menlo Connect Setup

Tamper Proofing Protection

Please provide Tamper Proofing password...

☒ **Enable Tamper Proofing Protection** (Recommended)

Tamper Proofing Protection is a newly introduced feature in this version which protects Menlo Connect from being removed by an unauthorized user.

Tamper Proofing Password

123456

Tamper Proofing Protection uses a modern Hashing algorithm technique to protect user password from brute-force attacks with pre-computed hashes.

Back Next Cancel

#### Improved

Menlo Connect Setup

**Uninstall Protection**

Prevent Menlo Connect from being removed by an unauthorized user.

☒ **Enable uninstall protection** (recommended)

Require a password to uninstall Menlo Connect from this device. [Learn more](#)

**Uninstall protection password**

Use a combination of letters, numbers, and symbols that's memorable to your organization...

Create a strong password you can remember. Recovering this password will require contacting support.

Back Next Cancel



# UX Writing Example

## Menlo Connect Installer

### FORMAT/TOOLS

- Google Docs
- Figma
- Snag-It
- Pendo



### UX WRITING



### USER RESEARCH



### DESIGN THINKING



### DOCUMENTATION



### Process

#### Scope

##### 1. Go where the work is.

The dev teams don't always know what kind of support writers provide, so it's important to listen.

##### 2. Know the schedule.

I booked a meeting with the Director of Product Management to better understand the event and audience.

##### 3. Get access.

Access to the tools isn't always possible, so I made sure I knew which design documents and mockups to use.

#### Iterate

##### 1. Templates first.

The teams insist on using Google Docs, but none were used for copy review before. I made templates.

##### 2. Heavy on detail.

I had to go into more details than was typical to make sure the names of components was clear. Dev team were in another time zone.

##### 3. Check in early and often.

With a wobbly source of truth, I had to check in early and often to make sure I had the latest and my suggestions were valid.

#### Review

##### 1. KIT with POCs.

With so many actors with crisscrossing time zones involved, knowing who to ask, and when, was vital.

##### 2. Test when you're able.

Later in the release cycle, an RC build will become available, and I would use it to validate my suggestions were correct; proved invaluable.

##### 3. Meetings when in doubt.

As time became scarce it became necessary to schedule face-to-face reviews with devs to speed up reviews.

#### Revise

##### 1. Know what's important.

Because I was involved late in the process note everything got done; some went to backlogs.

##### 2. Get support involved.

Get Support involved. They can provide reviews and add vital context for tricky situations. They also help identify lagging problem indicators.

##### 3. Keep the doc updated.

The UI changes meant some of the doc was now outdated. I ensured the work was addressed or put in a backlog.



# **UX Writing/User Research**

## **Bitbucket Cloud CoreX Copy Audit**

2019

# UX Writing/User Research Example

Bitbucket Cloud CoreX Content Audit

## FORMAT/TOOLS

- Mural
- Confluence pages
- Jira issues
- Bitbucket Pull requests



## UX WRITING

## USER RESEARCH

## DESIGN THINKING

## DOCUMENTATION

### Project Overview

**My contribution:** I identified over 200 usability bugs and improvements, scoped the amount of effort and value for each, then aggregated and scoped the work for a team of part-time contract engineers to work on for two quarters.

### Project Goals

Identify usability issues to address quickly, focusing on content improvements within the in-app copy, which will:

#### Reduce new user churn

**Create style guides and patterns.** Establish consistency among content elements and components.

**Harden the core user experience.** UI inconsistencies contributed to customer dissatisfaction and churn.

**Create a visual journey map.** Identify the highest priority screens and dialogs every user encounters.





#### Reduce complexity perceptions

**Triage feedback and support cases.** Things were hard to find, start-up tasks opaque, and hard to find help.

**Keep accessibility in mind.** Included concerns for screen readers and visually impaired users.

**Scope, chunk, and plan dev work.** Identified quick wins, design the solution, review the changes.

### Team

- 2  Writers (Me + another)
- 1  Designer
- 3  Engineers (contractors)
- 1  Product Manager

### Project Duration

9 months

# UX Writing/User Research Example

## Bitbucket Cloud CoreX Content Audit

### FORMAT/TOOLS

- Mural
- Confluence pages
- Jira issues
- Bitbucket Pull requests



### UX WRITING



### USER RESEARCH



### DESIGN THINKING



### DOCUMENTATION



### Problem statement

Bitbucket's perceived complexity stems from issues with the content and identifying and improving content challenges could positively impact the core experience using Bitbucket. Content is inconsistent, and users are leaving Bitbucket because of usability issues, and we deduced the inconsistent copy plays a role in that.

There were reports that things were hard to find, it was difficult to complete tasks, and it was difficult to predict where to find help. Evidence of this was repeat support cases from new users, negative sentiment reports on documentation feedback forms, and new customer churn.

### Outcomes

#### Directly addressed user-reported pain points

- We aggregated backlogs of user feedback (including bugs), NPS scores, and web analytics to identify patterns of user pain.
- Some of the problems reported were years old, and lots of feedback on some of the most visited pages of all Atlassian doc.

#### Created a healthy, holistic UX design precedence

- The analysis of the audit findings helped guide decision making and goal setting on the Bitbucket design team.
- This project helped to support a renewed emphasis on Bitbucket UX design by establishing patterns and creating style guides.

#### Maximize the impact and value of audit findings

- Created clearly defined chunks of work; approximately 10 weeks worth of work, grouped into sprints.
- Components that had improvements already committed in the product roadmap were added to feature team backlogs.

# UX Writing/User Research Example

## Bitbucket Cloud CoreX Content Audit

### FORMAT/TOOLS

- Mural
- Confluence pages
- Jira issues
- Bitbucket Pull requests



### UX WRITING



### USER RESEARCH



### DESIGN THINKING



### DOCUMENTATION



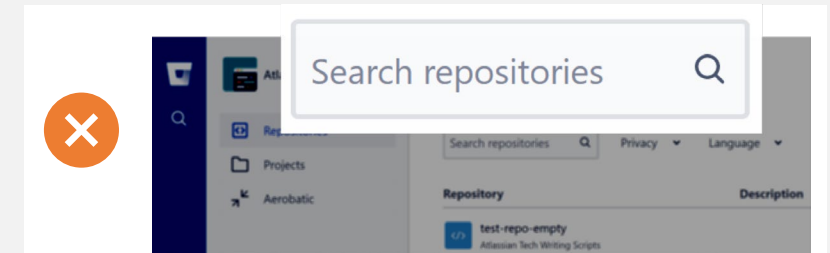
### Outcomes

### Recommendations

**Recommendations for improvement were grouped according to themes.** Chunking the work in this way helps to identify problematic patterns or general inconsistencies within Bitbucket Cloud's interface.

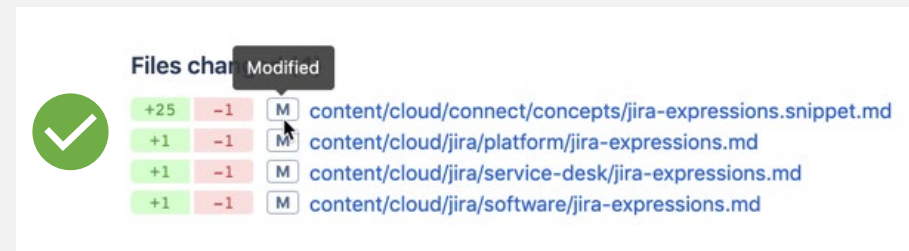
#### Mismatched expectations for 'Filter' and 'Search'

- In some parts of the UI we refer to looking for an object as "search," (code search) when the behavior is actually filtering.
- Defined the behavior and updated copy to be consistent in alerts, field labels, empty states, and error states.



#### Inconsistent use of menus, icons, and tooltips

- "More actions" menus are like a box of chocolates, you never know what you're going to get. Need to establish some patterns.
- This project helped to support a renewed emphasis on Bitbucket UX design by establishing patterns and creating style guides.



# UX Writing/User Research Example

## Bitbucket Cloud CoreX Content Audit

### FORMAT/TOOLS

- Mural
- Confluence pages
- Jira issues
- Bitbucket Pull requests



### UX WRITING



### USER RESEARCH



### DESIGN THINKING



### DOCUMENTATION



### Research Process



#### 1. Define the problem.

Audits can encompass anything, so it was important to understand intended outcomes.

#### 2. Find the time.

This project was done in addition to my other responsibilities. I had to carve out time to work on this.

#### 3. Gather resources.

We probed backlogs of user feedback and feature requests to inform our work (and not reinvent the wheel).

#### 1. Weekly meetings.

The designer and I met weekly during the journey map creation process, which was very helpful.

#### 2. Cataloging severity.

I never had continuous time for this project, so leaving detailed notes helped keep important tasks top of mind.

#### 3. Chunking.

Once themes began to emerge, grouping similar types of changes helped with velocity of the impact of changes.

#### 1. Present findings.

Stakeholder buy-in to get approval for developer time, so making findings clear was important.

#### 2. Internal blog post.

Since this was a long-running project, a narrative-style blog post helped convey the work involved and how to contribute.

#### 3. Coordinate with product.

Lots of the changes we proposed coordinated with planned feature development for greater efficiency.

#### 1. Coordinate with dev.

All the hard work would amount to little without contract engineers making the changes.

#### 2. Put metrics in place.

We had to always be asking the devs, "How do we know that worked?" to put in place ways to measure user interaction.

#### 3. Team reports.

My Jira work paid off in the form of reports on how much work went into this project, recognizing everyone's effort.

2019

# UX Writing/User Research Example

## Bitbucket Cloud CoreX Content Audit

### FORMAT/TOOLS

- Mural
- Confluence pages
- Jira issues
- Bitbucket Pull requests

### Outcomes

- Outlines the core experiences all Bitbucket Cloud users go through.
- I collaborated with a designer to help create this infographic. We met one a week to discuss and iterate on the document.
- We used Mural to create the document: collaborating asynchronously online, it became a living document we could depend on.

### User Journey Map



### UX WRITING

### USER RESEARCH

### DESIGN THINKING

### DOCUMENTATION



# UX Writing/User Research Example

## Bitbucket Cloud CoreX Content Audit

### FORMAT/TOOLS

- Mural
- Confluence pages
- Jira issues
- Bitbucket Pull requests



### UX WRITING

### USER RESEARCH

### DESIGN THINKING

### DOCUMENTATION

### Outcomes

**Tracked the audit with Confluence.** I worked my way through the interface taking screenshots and evaluating areas for improvement.

**Prepare phase**

Created by John Doe  
Last updated Jul 26, 2019 • 7 min read • 10 Analytics

**COMPLETE** This section of the audit focuses on the Prepare stage screens: Code search, Source browser, and Commit detail.

• **DOD-58** Conduct copy audit for Prepare phase screens **DONE**

**Bitbucket User Journey Phase**

Stage: Prepare

Steps: 1. Search 2. Source browser 3. Commit detail

**Key features**

- Code search
  - GLOBAL - Code search/Sidebar
    - ACCOUNT - Code search
- Source browser
  - REPOSITORY - Source browser
    - W-README, no Description
  - Loading add-on
  - More action dialog
  - Watch repo dialog (not watching)
  - Watch repo dialog (watching)
- Branch/tag selector/filter
  - Tag filter ideal state
  - No tags/empty state
  - Error state - No branches
  - Empty state
  - Partial
- Clone dialog
- Commit detail
  - REPOSITORY - Commit detail
    - Comments - Partial
    - Side-by-side diff
    - Side-by-side Partial
    - Run pipeline
    - Pull requests
  - Delete comments dialog
    - Comment on deleted comment

**Code search**

**GLOBAL - Code search/Sidebar**

| State | Screens | Analysis   |
|-------|---------|--|
| IDEAL |         | <b>QUICK WINS</b> <ul style="list-style-type: none"><li>1. Add a footer so users know when they've reached the end of the list.</li><li>2. "View all code matches" seems like it could be improved.</li></ul>  |
| EMPTY |         | <b>QUICK WINS</b> <ul style="list-style-type: none"><li>1. Placeholder text could be more useful, less marketing-y.<ul style="list-style-type: none"><li>a. Perhaps could include a sample of syntax to help?</li></ul></li><li>2. "Search for code" feels awkward and out of place, and it doesn't feel clear that it's different than standard search.</li></ul> |

### Audit Findings

| State   | Screens | Analysis   |
|---------|---------|--|
| EMPTY   |         | <b>QUICK WINS</b> <ul style="list-style-type: none"><li>1. Placeholder text could be more useful, less marketing-y.<ul style="list-style-type: none"><li>a. Perhaps could include a sample of syntax to help?</li></ul></li><li>2. "Search for code" feels awkward and out of place, and it doesn't feel clear that it's different than standard search.</li></ul>   |
| LOADING |         | <b>QUICK WINS</b> <ul style="list-style-type: none"><li>Feels super snappy and doesn't have a loading state.</li></ul>   |
| ACTIONS |         | <b>QUICK WINS</b> <ul style="list-style-type: none"><li>1. We could and should standardize CTAs look/feel and content.</li><li>2. "View all repositories" needs to be changed to be more accurate. "View matching repositories" perhaps.</li></ul>   |
| PARTIAL |         | <b>QUICK WINS</b> <ul style="list-style-type: none"><li>1. Way too much info below the fold. The bottom should reflect Confluence's, with a CTA to do an "Advanced search."</li><li>2. Spacing feels extra tight.</li><li>3. How might this change when Workspaces is fully realized?<ul style="list-style-type: none"><li>a. Searching in the context of a Workspace feels potentially really useful.</li></ul></li></ul> |
| EMPTY   |         | <b>QUICK WINS</b> <ul style="list-style-type: none"><li>1. Placeholder text could be more useful, less marketing-y.<ul style="list-style-type: none"><li>a. Perhaps could include a sample of syntax to help?</li></ul></li><li>2. "Search for code" feels awkward and out of place, and it doesn't feel clear that it's different than standard search.</li></ul>   |

### Scoped Dev Work

**Tracked requirements for dev in Jira.** I made mockups and provided the changes to make as requirements within the issues.

| Key     | T | Summary  |
|---------|---|--|
| BUS-294 | 🔴 | Pipelines filters panel: fix filters panel position, fix the user's avatar (instead of the user's photo, we get an avatar with initials) |
| BUS-281 | 🔴 | Pipelines filters panel: replace router Link with a custom component (AFP)   |
| BUS-298 | 🔴 | Fix spacing between "recent repositories" and repo cards   |
| BUS-209 | 🔴 | Global PR page - workspace tab: there is a random dropdown behind the table, and cannot switch filters                                   |
| BUS-203 | 🔴 | On the repo sidebar, the "Pipelines" and "Deployments" icons do not become blue when selected.   |
| BUS-248 | 🔴 | On file viewer, file row fill is not consistent when then viewport is taller than # rows   |
| BUS-258 | 🔴 | Code search in repo UI updates   |
| BUS-215 | 🔴 | "Code search in this repository" ends up in an error state, not an empty state   |
| BUS-212 | 🔴 | Repo/Source filter bar UI fix  |
| BUS-266 | 🔴 | Update disabled state in code search in source browser   |
| BUS-220 | 🔴 | Code Search - fix responsive behaviour   |
| BUS-223 | 🔴 | Source annotate - various fixes  |
| BUS-267 | 🔴 | The background page scrolls when there is a Jira issue bento overlay   |
| BUS-268 | 🔴 | Jira issue summary does not truncate in Your Work dashboard and Jira issues tab  |
| BUS-229 | 🔴 | Pull requests: Selected lines do not maintain highlighted state when refreshing or opening URL in a new tab                              |
| BUS-280 | 🔴 | Increase width of "SSH/HTTPS" dropdown select to 96px  |
| BUS-238 | 🔴 | Source: clicking on files after clicking on search filter does not work.   |
| BUS-296 | 🔴 | Updated create repository page (this page is under FF): field labels aren't aligned with corresponded fields                             |
| BUS-42  | 🔴 | Same diff multiple draft comments & browser warning for unsaved comments.  |





# **Developer Documentation**

## **Bitbucket Cloud API Proxy Module**

# Developer Documentation Example

## Bitbucket Cloud API Proxy Module

### FORMAT/TOOLS

- Jekyll static site generator
- Postman
- Confluence pages
- Visual Studio
- Markdown
- Bitbucket pull requests

### Project Overview

Bitbucket Cloud's API proxy module was a planned change as a reaction to GDPR and would impact all platform add-on developers. It changed Bitbucket's inherent object model, and how developers authenticate calls.

### Project Goals

Describe how to use the API proxy module to optimize integrations and adhere to new GDPR requirements.

#### Document the API module

**Add new sections:** The developer doc was out of date. Needed to update the page tree to add content.

**Uncover missed dependencies:** Started as internal tool, it was not ready for production yet.

**Learn to contribute to dev docs:** This was my first role working on dev docs. Learn new authoring environment.




#### Address documentation gaps

**Catalog and track gaps:** These changes cascaded to other parts of using the API, which required updating other docs.

**Schedule dev work:** Some gaps required changes to codebase, required coordination with product management.

**Draft the missing docs:** Drafted outlines and rough drafts of the missing docs to be reviewed and implemented later.

### Team

- 1  Writers (Me)
- 1  Product Manager
- 3  Engineers

- 1  Software Architect

### Project Duration

4 months



### UX WRITING



### USER RESEARCH



### DESIGN THINKING



### DOCUMENTATION



# Developer Documentation Example

## Bitbucket Cloud API Proxy Module

### FORMAT/TOOLS

- Jekyll static site generator
- Postman
- Confluence pages
- Visual Studio
- Markdown
- Bitbucket pull requests

### Problem statement

- The change was reactionary to GDPR, and there were some architecture requirements that were originally unknown, and the project took much longer than expected.
- One piece of doc wasn't going to be enough, as this new functionality didn't just add a single module, it added several other complimentary, but entirely separate, features that had to be explained.
- This was my first time contributing to developer documentation.

### Outcomes

#### New module included lots of code examples

- Like most developer content, it needed to include lots of examples that required me to pair with a developer to create and test.
- Paired with a developer to describe set of examples that will demonstrate all the vital functionality without confusing people.

#### Documentation gaps were addressed or indexed

- Added several other complimentary, but entirely separate, pieces of doc for features that had yet to be explained.
- Ended with a tight, comprehensive, piece of content that helped devs get the most from their integration with Bitbucket.

#### Great-looking graphics

- Insisted on the graphics to help portray the information flow.
- Architect drew out a crude line diagram I iterated into a polished into a production ready publishable graphic.



### UX WRITING



### USER RESEARCH



### DESIGN THINKING



### DOCUMENTATION



2018

# Developer Documentation Example

## Bitbucket Cloud API Proxy Module

### Outcomes

### Proxy Module Doc

You can see this piece of documentation at:

[developer.atlassian.com/cloud/bitbucket/proxy-module/](https://developer.atlassian.com/cloud/bitbucket/proxy-module/)

The screenshot shows the Bitbucket Cloud Developer documentation page for the API proxy module. The page has a blue header with navigation links: Guides, Reference, and Resources. The main content area is divided into sections: Introduction and Basics, Security, and Learning. The Introduction and Basics section is currently selected and expanded, showing a list of topics: Integrating with Bitbucket Cloud, Getting started, Frameworks and tools, and Install an app from your site. The Security section includes Security overview, Understanding JWT for apps, Authentication for apps, OAuth 2.0, and Query string hash. The Learning section includes Patterns and examples, API proxy module (highlighted), Object hydration, and Application properties. The main content area displays the API proxy module documentation, including a code example for the proxy module configuration and a section titled 'How it works' which includes a diagram of the data flow.

Bitbucket Cloud Developer

Guides Reference Resources

Latest updates

INTRODUCTION AND BASICS

- Integrating with Bitbucket Cloud
- Getting started
- Frameworks and tools
- Install an app from your site

SECURITY

- Security overview
- Understanding JWT for apps
- Authentication for apps
- OAuth 2.0
- Query string hash

LEARNING

- Patterns and examples
- API proxy module
- Object hydration
- Application properties

API proxy module

The API proxy module creates a layer between your API and Bitbucket to provide the heavy lifting for authentication, parameter substitution, object hydration, and other services. This should make the API integrations and apps you build into Bitbucket Cloud more efficient and easier to build.

You create the proxy connection when you map a URL pattern to a specific destination URL in your app descriptor. The simplest form of this is:

```
1 "modules": {
2   "proxy": {
3     "/proxy-example/{target_user}/{repository}": {
4       "destination": "/proxy-example{name={repository.owner.display_name}}", // required
5       "methods": { ... }, // optional
6       "conditions": { ... }, // optional
7       "scopes": { ... } // optional
8     }
9   }
10 }
```

Here `/proxy-example/{target_user}/{repository}` is the path available on Bitbucket, and would be the path you would use in calls to an `API` object, for example. The destination URL is a path served by your server, and maps to a specific Bitbucket resource, in this case the display name of the repository owner.

Before you begin

It's important to understand these core concepts before you jump in to using the API proxy module.

- Authentication between Bitbucket and Connect apps
- Authorization, including conditions and scopes
- Context parameters in path segments

Looking for reference doc or examples?

**Reference:** Find the elements, definitions, and properties of the API proxy module in the [API proxy module reference docs](#).

**Examples:** Example configurations that demonstrate how to use the proxy module in your app descriptor can be found in the [Examples](#) section.

How it works

The following diagram depicts the basic operation of proxy module, from authenticating the client call to hydrating the response object.

The diagram illustrates the data flow of the proxy module. It shows three main components: CLIENT, BITBUCKET PROXY, and MYAPP.COM. The flow is as follows: 1. CLIENT sends a request to BITBUCKET PROXY. 2. BITBUCKET PROXY sends a request to MYAPP.COM. 3. MYAPP.COM sends a response back to BITBUCKET PROXY. 4. BITBUCKET PROXY sends a response back to CLIENT. The diagram is labeled 'Data flow of the proxy module'.



### UX WRITING



### USER RESEARCH



### DESIGN THINKING



### DOCUMENTATION



2018

# Developer Documentation Example

## Bitbucket Cloud API Proxy Module

### FORMAT/TOOLS

- Jekyll static site generator
- Postman
- Confluence pages
- Visual Studio
- Markdown
- Bitbucket pull requests

### Outcomes

- **Originally part of the proxy module doc:** This entire document was meant to be part of the API proxy module documentation; it's actually longer!
- **Required extensive paring with a dev:** Over several meetings we established requirements for the ideal set of examples to demonstrate all the vital functionality without confusing people.

You can see this piece of documentation at:

[developer.atlassian.com/cloud/bitbucket/proxy-object-hydration/](https://developer.atlassian.com/cloud/bitbucket/proxy-object-hydration/)



### UX WRITING



### USER RESEARCH



### DESIGN THINKING



### DOCUMENTATION



### Obj Hydration Doc

Bitbucket Cloud Developer

GuidesReferenceResources

Q

Latest updates

INTRODUCTION AND BASICS

Integrating with Bitbucket Cloud

Getting started

Frameworks and tools

Install an app from your site

SECURITY

Security overview

Understanding JWT for apps

Authentication for apps

OAuth 2.0

Query string hash

LEARNING

Patterns and examples

API proxy module

Object hydration

Application properties

PRIVACY GUIDELINES

User privacy guide for app developers

Profile visibility

BUILDING BLOCKS

App context

App descriptor

Conditions

Context parameters

Access to context parameters

OTHER CONSIDERATIONS

Atlassian Design Guidelines

Ataskit

Atlassian Marketplace

Cloud app licensing

Last updated Jan 7, 2021

Rate this page: ☆☆☆☆

Response object hydration

Object hydration allows your app to respond to proxied API requests with a minimal amount of data for a resource and Bitbucket will add the full object from the Bitbucket database. This API proxy service will automatically fill in any missing elements of certain Bitbucket data types returned by a remote service. This allows remote services to keep track of only the absolute minimum information to uniquely identify an object.

For example, if your app were meant to get details of a single repository it only has to store a way to identify the repository, which can be as little as a repository UUID. Without using the proxy module, a simple request using a repository UUID returns a partial repository response object, without being hydrated, that would look like this:

```
1 {
2   "type": "repository",
3   "uuid": "{aa5acc33-e5f7-43e9-883d-50325fc68ca8}"
4 }
```

Using the proxy module, when the response is passed to Bitbucket's API proxy, the proxy:

1. Decodes the JSON from the remote service.
2. Scans the object graph for type elements defined in Bitbucket.
3. For each type, queries the database and substitutes it for the full object.
4. Serializes the resulting object, producing layouts consistent with other, core APIs.

The hydrated response coming from the API proxy provided to the client would look like the entire response body from the repository endpoint.

```
1 {
2   "scm": "git",
3   "website": "",
4   "has_wiki": false,
5   "uuid": "{aa5acc33-e5f7-43e9-883d-50325fc68ca8}",
6   "links": {
7     "watchers": {
8       "href": "https://api.bitbucket.org/2.0/repositories/atlassian_tech_writing/teams-in-sp-
9     },
10    "branches": {
11      "href": "https://api.bitbucket.org/2.0/repositories/atlassian_tech_writing/teams-in-sp-
12    },
13    "tags": {
14      "href": "https://api.bitbucket.org/2.0/repositories/atlassian_tech_writing/teams-in-sp-
15    },
16    "commits": {
17      "href": "https://api.bitbucket.org/2.0/repositories/atlassian_tech_writing/teams-in-sp-
18    },
19    "clone": [
20      {
21        "href": "https://bitbucket.org/atlassian_tech_writing/teams-in-space-tutorial-cont-
22        "name": "https"
23      },
24      {
25        "href": "git@bitbucket.org:atlassian_tech_writing/teams-in-space-tutorial-content.1
26      }
27    ]
28  }
```

Capturing the parent object of the object you're capturing

For objects that belong to another object -- like how a repository belongs to a user or team -- **the parent object may need to be resolved in order to resolve a child object**

ON THIS PAGE

Capturing the parent object of the object you're capturing

Response objects that can be hydrated

Issue schema

Snippet schema

Pull request schema

Repository schemas

Account schemas

Project schemas

Issue comment schema

Commit comment schema

Pull request comment schema

Build schemas

Commit schema

Snippet commit schema

Commit file schema

Commit directory schema

Rendered schemas

Error schema

# Developer Documentation Example

## Bitbucket Cloud API Proxy Module

### Documentation Process

#### Scope

**1. Listen, ask questions:**

The original project was daunting, so I had to listen and do a lot of background reading.

**2. Create an outline:** This project had a lot of interdependent and interweaving parts, an outline helped stay organized.

**3. Shared understanding:** Resist the urge to 'figure it out later' when the initial assignment only consists of 'document this' for requirements.

#### Iterate

**1. Get out of the hole:**

Instead of iterating alone, it was important to get reviewers eyes on the work as soon as possible.

**2. Raise concerns early:** As soon as the scope starts to creep, raise the red flag and ask for support or push back the deadlines.

**3. Feedback from outside the team:** Sometimes your teammates are too close to the problem and share biases. Look for reviews from other teams.

#### Review

**1. Schedule stakeholder reviews:** The folks reviewing my work get busy, so I make sure to schedule their reviews..

**2. Be specific:** Long pieces of documentation like this can be daunting to review. Be precise about what should and shouldn't be reviewed.

**3. Test it yourself:** Though it can add to turnaround time, testing procedures yourself is the only way to ship instructions with genuine confidence.

#### Measure

**1. Use your resources:** Not everyone reads release notes, so it can help to advertise new docs in community forums.

**2. Set a future date:** We don't always stop to smell the roses. We also don't plan to stop and measure work in the future but doing so ensures you will.

**3. Check social media:** If the product is publicly available and popular chances are someone online has shared an opinion of it. Find them.



#### FORMAT/TOOLS

- Jekyll static site generator
- Postman
- Confluence pages
- Visual Studio
- Markdown
- Bitbucket pull requests

#### UX WRITING

#### USER RESEARCH

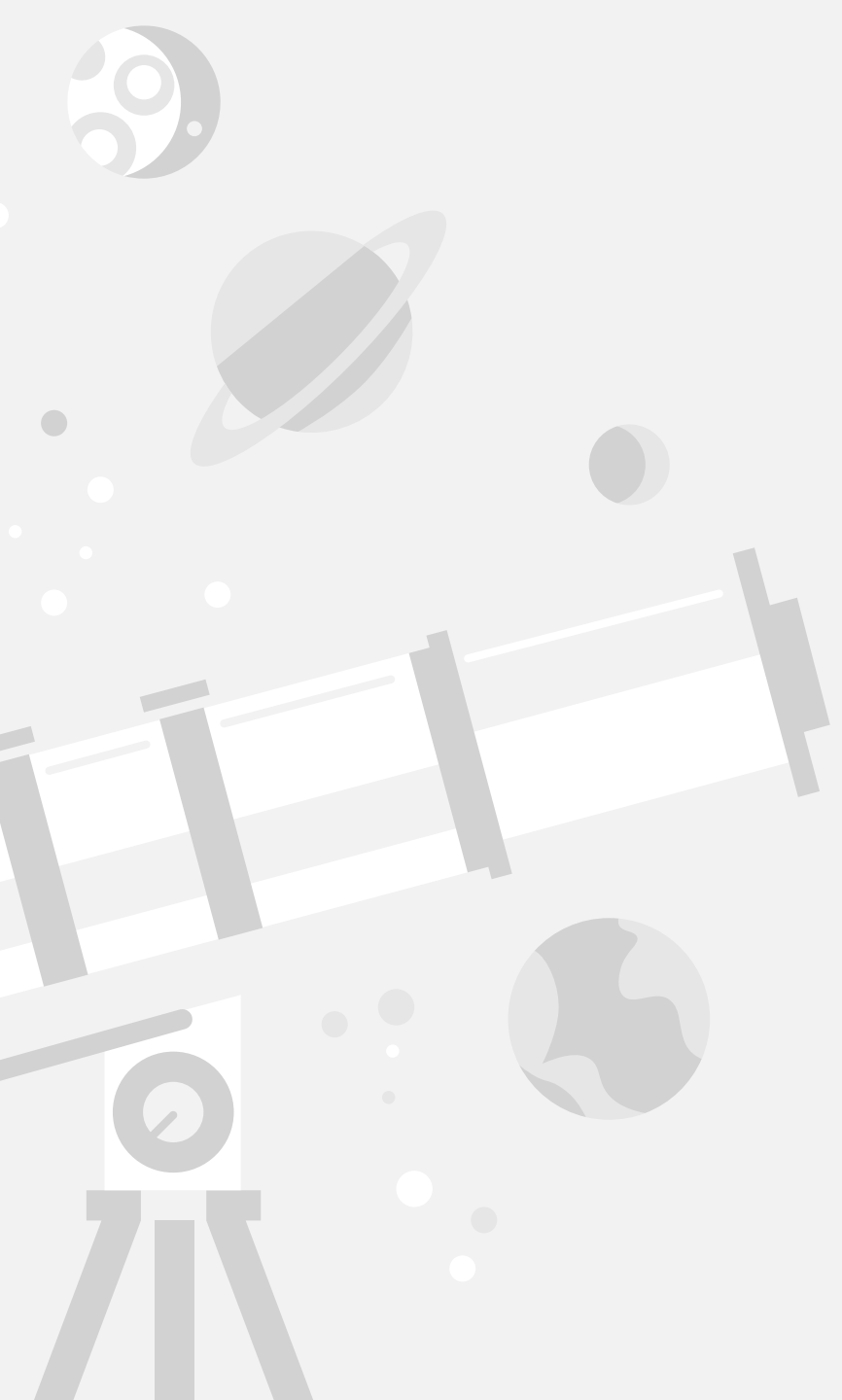
#### DESIGN THINKING

#### DOCUMENTATION



# **More About Me**

A few quick things to note



# Public Speaking



# More About Me

## Public Speaking

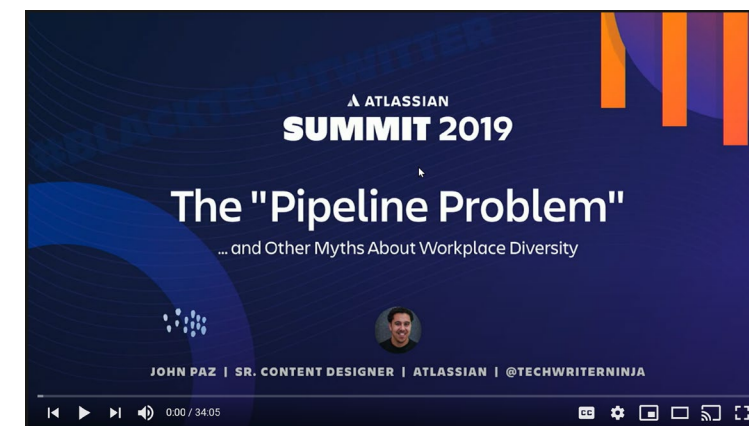
- **Telling stories is my favorite pastime.** I may be a Technical Writer/Content Design by day, but I'm a creative writer. I bring my passion for telling stories from the page to the stage. My experiences working in tech are usually at the center of my talks.
- **It all started with a 'Pipeline.'** My signature talk started with a simple Tweet about what being Black in tech looks like. All these years later and that experience is still driving people to my talks online, and to invite me to give the talk all over the world.
- **Talks for all stages, big and small.** I didn't start by giving talks to big audiences, and I still love to give more intimate presentations to smaller audiences about a wide range of topics, including: Agile, innovation, Tech Writing, Content Design, UX Writing, job hunting, and diversity.

## Atlassian Design Week 2019



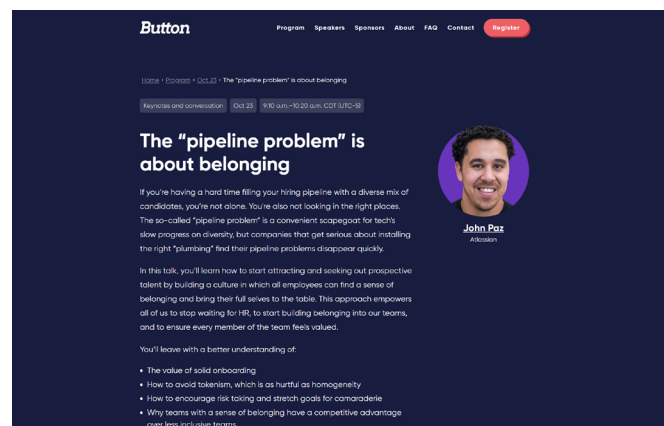
[paz.tips/john-presentation-building-belonging](https://paz.tips/john-presentation-building-belonging)

## Atlassian Summit 2019

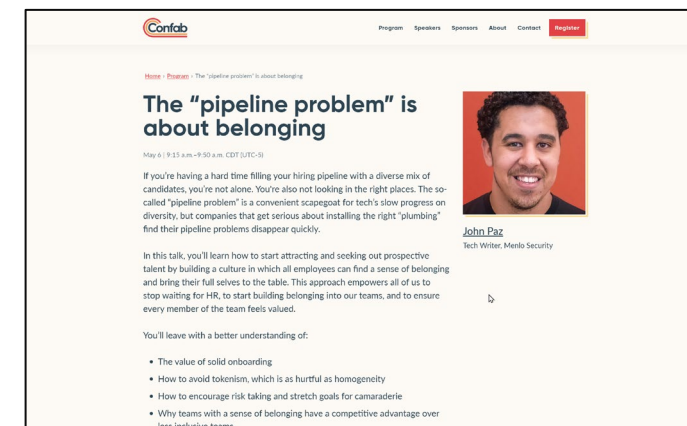


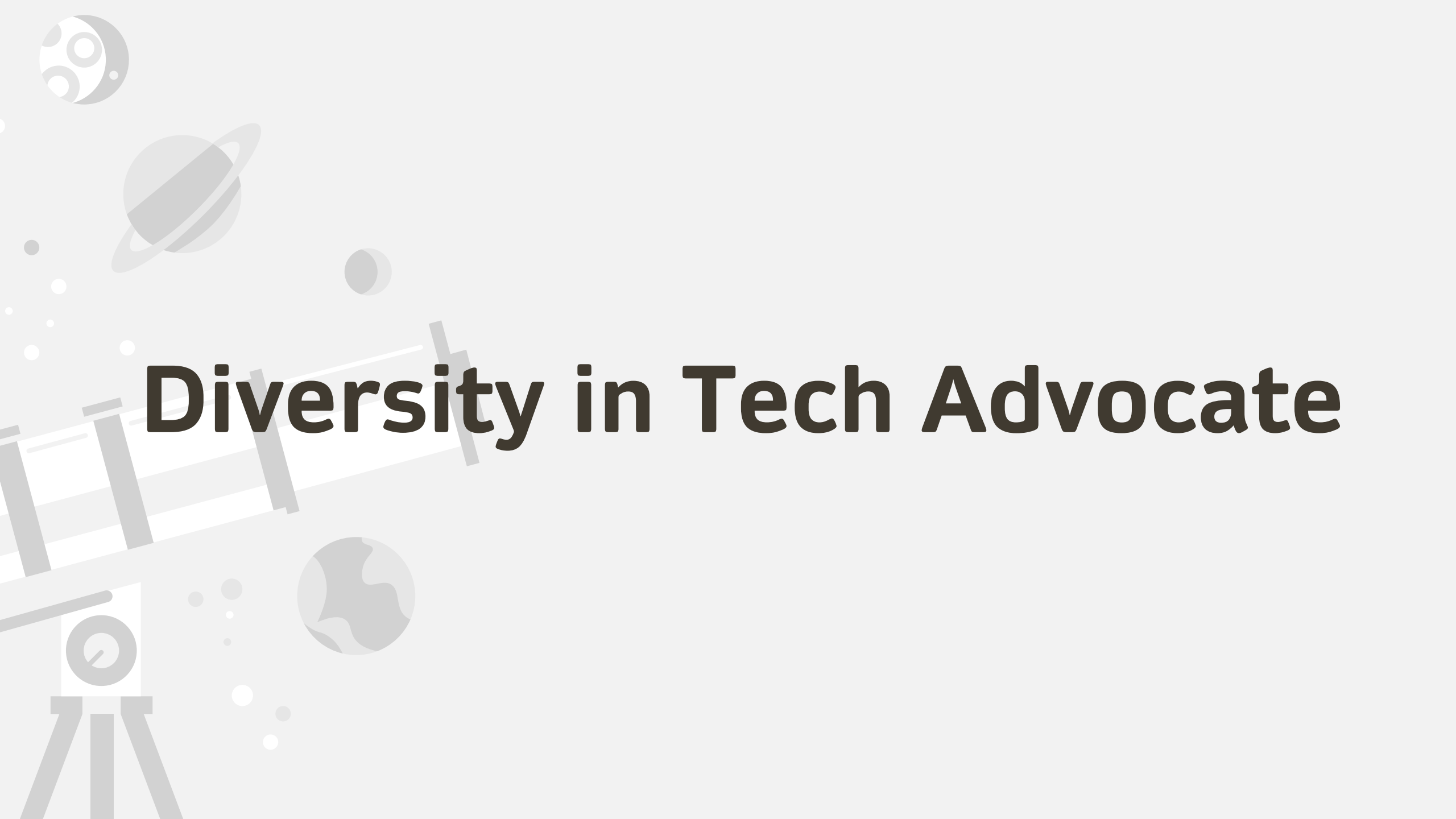
[paz.tips/pipeline-problem-summit-2019](https://paz.tips/pipeline-problem-summit-2019)

## Button (2020) keynote speaker



## Confab (2021) keynote speaker





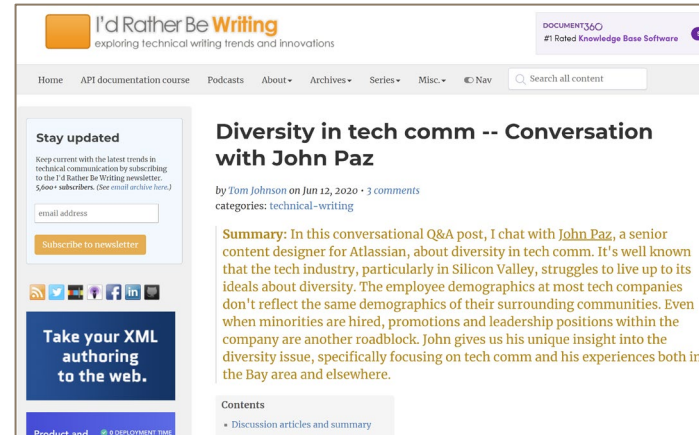
# **Diversity in Tech Advocate**

# More About Me

## Diversity Advocate

- **I make a conscious effort to expand my influence to increase my impact.** The messages from my presentations generate other opportunities for industry blogs and podcasts.
- **My unique perspective.** My personal identity consists of a few underrepresented groups, so I get requests for interviews from lots of tech-related or tech-comm-related blogs. I love to be interviewed! I am honored to act as a spokesperson for diverse voices working in tech, and it puts me in direct contact with people in my field I highly respect.
- **Emphasizing the benefits of mentorship to improve diversity and equity in tech.** I make a conscious effort to drive the conversations within tech companies to think deeply about how to be more equitable and increase representation of underrepresented people working in tech.

## I'd Rather Be Writing (2020)



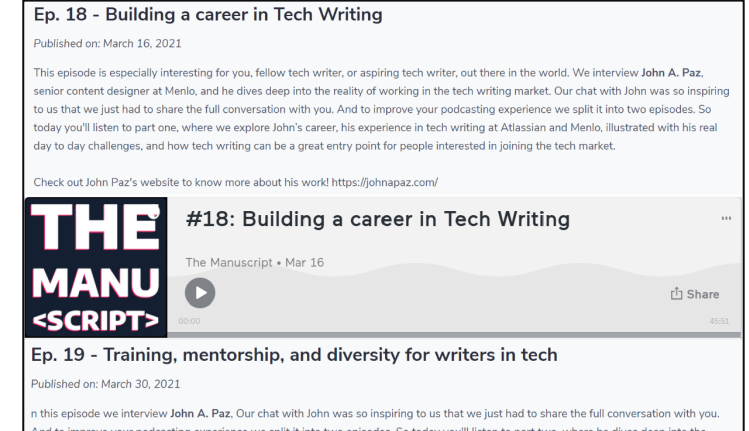
[paz.tips/john-interview-idratherbewriting2020](https://paz.tips/john-interview-idratherbewriting2020)

## People of Color in Tech (2019)



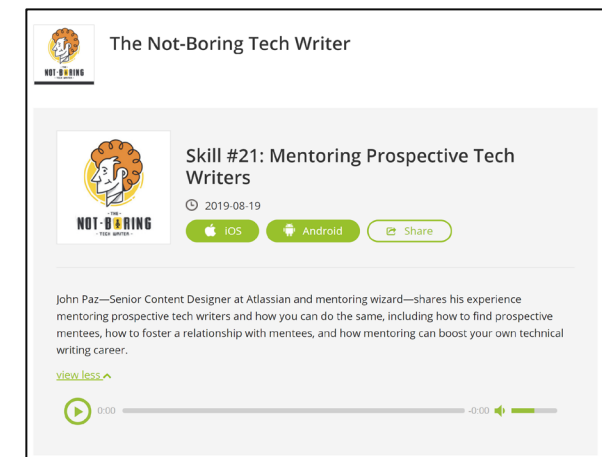
[paz.tips/john-interview-pocit2019](https://paz.tips/john-interview-pocit2019)

## The Manus<Script> Podcast (2021)



[paz.tips/john-interview-themanuscript2021](https://paz.tips/john-interview-themanuscript2021)

## The Not-Boring Tech Writer (2019)



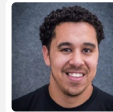
[paz.tips/john-interview-thenotboringtechwriter2019](https://paz.tips/john-interview-thenotboringtechwriter2019)

# Get in touch

Look for more of my work online



[paz.tips/johnpaz-linkedin](https://paz.tips/johnpaz-linkedin)



[paz.tips/johns-online-portfolio](https://paz.tips/johns-online-portfolio)



[paz.tips/twitter-mrcontentdesign](https://paz.tips/twitter-mrcontentdesign)

## Content Design Portfolio

Updated July 2022