Memorandum

To: Dr. Kim Campbell

From: Kali Rossi

Date: August 14, 2026

Subject: Final Status Report

Technical writing at Tanium

Tanium is a cybersecurity and systems management company. Their platform gives IT and security teams real-time visibility into every endpoint on their network, including laptops, servers, and other devices, so they can manage, patch, and secure them all from one place. Even organizations with millions of devices can get that information back in seconds, not hours.

I'm working as a technical writing intern on Tanium's documentation team. In the second half of my internship, my main job duties have continued to be researching an AI documentation project and working day-to-day as a technical writer, from drafting content, to finalizing it after reviews, to publishing it to help.tanium.com. Along the way, I've kept learning about topic-based authoring, helped with content migration, and gotten a better sense of Tanium's corporate structure through the internship program.

Below, I'll walk through what I've learned in each of these areas, then share two work samples: a Deploy guide update and a Jump Gate guide update. Because Tanium is a cybersecurity company, some details, like specific tools or the exact topic of my research project, are confidential, so I'll keep those parts a little vague.

Job duties

Research project: presentation

In the first half of the internship, my research centered on identifying an AI-driven solution that could improve my team's workflow. I'm not able to share specifics about the research topic itself, but the work involved independent research along with interviews with most of the documentation team about their day-to-day processes, to understand where an AI tool could realistically help.

Building on that foundation, I spent the second half of the internship compiling everything I'd gathered from the interviews and general research into a five-minute final presentation for the R&D team at Tanium, an audience of about 200 people. Before presenting, I did a practice run with our internship HR manager, which helped me refine the delivery.

One thing I had to think through carefully was audience: as a technical documentation intern presenting mostly to developers, I couldn't take the same approach I might have with a documentation-focused group. I drew on techniques from my technical communication coursework in grad school, and a teammate's advice to "tell a story" ended up being especially helpful in shaping how I structured the material. To keep the audience engaged, I kept the pace moving and included a live demo.

Day-to-day work

While I'm not working on my research project, I typically help with our content migration or work through real fixes to improve our content. I still haven't worked an entire sprint with a real team, but I've continued working through real Jira tickets with my mentor, and I've gained more independence in that process. I'll receive a ticket, figure out what needs to be fixed, find the affected files, and make the edits before publishing to our website.

One change from earlier in the internship is that I can now go straight to developers with questions, rather than having them filtered through my mentor, within reason. Throughout this process, I still have to stay aware of the company's version control procedures and style guide rules, and I now get reviews from both my mentor and the relevant developers before anything goes live.

Mentorship

My mentor, Kevin, has continued to be a huge asset throughout the second half of the internship. We still meet a few times a week so I can update him on my work, and he continues to take the time to check what I've done and answer any questions I have. When something's more complex, he sets aside extra time to walk through it with me.

Beyond the technical side, Kevin has kept teaching me how to navigate a corporate environment. That guidance has shaped me into not just a better technical writer, but a better employee and team member overall.

This mentorship hasn't been limited to Kevin. I've also had the chance to work more closely with my manager and the rest of the team as the internship has gone on, and that support has stayed consistent. Everyone I've worked with has made clear I can come to them with questions, whether technical or about my career more broadly, and they've followed through every time. Overall, I've found the whole team easy to work with and genuinely invested in helping me grow as a technical writer.

Work environment

I work out of Tanium's Addison, TX office, though the company is headquartered in Bellevue, Washington. I'm on a hybrid schedule, splitting my time between three days in the office and two days remote. My whole team, though, is located elsewhere: most work remotely, and a few work out of different offices entirely.

Despite being spread across locations and time zones, the team has continued to work remarkably well together throughout the internship. We have team meetings to sync up when needed, but for the most part, everyone works with a lot of autonomy.

Being the only one on my team in the Addison office has turned out to be one of the more valuable parts of the internship. It's pushed me out of my comfort zone to talk to employees from other teams, and over the course of the summer I've gotten to know people on the Customer Experience team, in data science, and in solutions engineering. That's given me a broader view of the company than I probably would've gotten otherwise.

What I'm learning

Topic-based authoring

I've mostly encountered topic-based authoring through content migration, but it also comes up regularly in my day-to-day Jira ticket work, whether I'm making basic updates or larger edits to a guide. We're moving content from MadCap Flare to Heretto, and the two handle authoring differently: Flare uses XHTML, while Heretto is built on DITA, which is much stricter about formatting. Our new platform is also more particular about chunking content into individual topics, so as I review pages, I've had to figure out where a topic should be split. I've learned to identify each piece as a task, concept, or reference topic and structure the content accordingly. Heretto has also been trickier with version control than Flare, though I've enjoyed the learning curve.

Content migration

The documentation team is moving their content from Flare to Heretto, and I've been able to help with the transition. Earlier on, I was mainly reviewing content after the migration had been completed by another technical writer, but I've since taken on more of the migration work myself. I migrated most of our ServiceNow integration guides on my own, handling both the migration itself and the content review afterward.

Corporate structure

One thing I appreciate about Tanium is the investment the company makes in its interns. In this program, we frequently communicate with our internship director, Bella Ayala. Bella hosts a few meetings for interns every week: Mondays, we have a session about something related to the corporate world, like onboarding, networking, or performance reviews. Fridays, we have a Q&A session with someone in a leadership position, such as a founder, CEO, or CPO.

These sessions are a reminder that Tanium treats intern growth as a priority, not an afterthought. The company sees interns as future talent worth investing in, not just summer help. That kind of investment has made me feel like a genuine part of the company and like I'm being set up for long-term success this summer.

Work Products

Sample 1 – Deploy update

This was a real Jira ticket, but the ticket itself didn't spell out exactly what needed to change. I had to search Slack channels, engineering repositories, and our test environment to piece together what needed to be updated in the deploy user guide to account for a new confidence score feature on our platform.

Once I identified the doc-impact, I drafted the updates, staged them, and created a Notion review page, which I sent to a developer and a product manager for feedback. After applying their feedback, I published the new guide in MadCap Flare.

This process gave me a much clearer sense of what doc-impact research looks like when the answer isn't handed to you directly in the ticket. I had to track down the right information across several different sources before I could even start drafting.

Sample 2 – Jump Gate update

Unlike my other work samples, this update wasn't tied to a single Jira ticket. Instead, I worked through an entire sprint board to identify several updates needed in the Jump Gate guide, which made it trickier than my other samples since it wasn't focused on one specific feature.

The process itself followed a similar pattern to the deploy update: I researched the doc-impact for each relevant item, drafted the changes, staged them, and created a Notion review page for a developer and product manager to review. After incorporating their feedback, I published the updates, though a few of the changes aren't live yet since the corresponding features haven't been released.

Working from a sprint board instead of a single ticket pushed me to think about doc-impact more broadly, since I had to evaluate a whole set of potential changes at once rather than focus on a single fix.

Sample 1 – Deploy confidence score update

Overview: Software package confidence score

Software package confidence score

A confidence score is the probability that a software package deployment succeeds. Tanium calculates this score using deployment metrics collected from previous deployments of that package across all Tanium customer environments. For more information, see Tanium Console User Guide: Confidence score.

Tanium does not calculate confidence scores for the following packages:

- Gallery packages not usable without customization, such as **Microsoft InPlace upgrade** packages
- Remove Only or Audit Only packages

For more information, see View confidence score of software package.

Managing software: Create a software package (step 7)

7. (Optional) If you select **Install** or **Update** in the **Deploy Operations** section, you can define **Tracked Executables**. For custom software packages, use this section to specify the executables that the package deploys. Tanium monitors these executables for performance degradation, which feeds into the confidence score's **Endpoint Health** metric.
    a. Click **Add** for each executable.
    b. Enter the name of the executable, such as `application.exe`.

Sample 2 – Jump Gate update

Jumping to endpoints: Jump to an endpoint (step 6)

6. The RDP session opens in a separate browser window.

NOTE

- When you use the ephemeral account for the first RDP session on an endpoint, it takes up to a few minutes for Windows to set up the new user, and onboarding dialogs might appear after Jump Gate signs you in. Jump Gate reuses the same temporary account for subsequent RDP sessions to the same endpoint within the same request, so setup does not repeat until you start the first RDP session for a new request.
- Windows-based session limitations apply for RDP sessions in Jump Gate. For example, desktop versions of Windows typically allow only a single session, and Windows Server session limits depend on Windows licensing and group policies. All other Remote Desktop policy settings and behaviors apply to RDP-based jump sessions as well.
- If your browser blocks the popup, the session opens inline on the **Request Details** page instead.
- Domain controllers do not support the **Use ephemeral account** option for RDP sessions. If you select this option on a domain controller, Jump Gate displays an error. Use the **Enter credentials** option instead.

TIP

RDP sessions support clipboard redirection. You can copy text on your computer and paste it into the RDP session. Copying content from the RDP session to your local computer is enabled by default and can be controlled by an operator using the **Enable clipboard copy** setting. For more information, see Configure Jump Gate module settings.

Approving jump requests and viewing activity: session monitoring note

View session recordings and command history

You can view session recordings and command history in the **Session Monitoring** section.

NOTE

The **Session Monitoring** section appears only if you created the request or have **Jump Gate Requests read** permission for the request's content set. Having only the **Jump Gate Requests approve** or **Jump Gate Requests close** permission does not grant access to session recordings and command history.