Git Productive!

A whitepaper for the 2020 Collegeville Workshop on Scientific Software, focussing on Developer Productivity.

Jason M. Gates, Joe Frye, Brent Perschbacher, Dena Vigil; *Sandia National Laboratories*

The Center for Computing Research (CCR) at Sandia National Laboratories does fundamental research and creates technologies spanning the domains of computer architecture, modeling physical phenomena, discrete mathematics, and data analytics, to name only a few. A little less than a decade ago, some colleagues of mine perceived a growing need in our center for training in the version control software git. People were comfortable with their CVS or SVN, but git was the cool new kid on the block. More than that, git's underlying model supports a variety of workflows that enable improved productivity, reliability, etc., and with the rise in popularity of services like GitHub, switching to git very much seemed like the thing to do. Everyone was jumping on the bandwagon, and then realizing after the fact that the learning curve was substantial. Enter CCR-University (just a fun name for the handful of us passionate about training) and our first *Introduction to Version Control and Collaboration with Git* course.

If you're a self-taught git kung fu master, you're probably thinking, "Why on earth do you need a course for that? There's plenty of information online just a few clicks away." That's a reasonable question, and you're perfectly right. What we've found in our experience, though, is that while some people excel at teaching themselves through documentation, many more prefer the interaction you get with a personal training. We have a wealth of knowledge to share from our years of experience, and are confident you can benefit from it when standing up your own training efforts (on any topic, not just version control) at your own institution.

Humble Beginnings

In the early days, our teaching team of five or six would serve about a dozen people per semester. For material we initially used The Carpentries' Version Control with Git course, and some of us went through the process of becoming certified Carpentries instructors. Over time, though, we began developing our own materials that were more tailored to the needs of the departments and teams we were assisting. Four one-hour classes would be held in a small conference room, and we'd hand out loaner laptops that we had configured appropriately for the course.

This is where things were when I joined the team three and a half years ago. While what we offered at that point met the initial need, the need had been growing as the years went on. Instead of, "How do I get up and running?" it was now, "How do I not shoot myself in the foot such that I need to blow it all away and clone from scratch!?" We've all been there, right? And as more teams were transitioning from their former version control system to git, we had more people looking for help to get up and running. It was time for us to expand to meet the growing needs of not just our center, but of the rest of the Sandia population as well.

Formalizing our Content

We set about productionizing our content (to use a software engineering term) by first gathering our materials into Confluence pages that participants would have access to during and after our course. In the process, we created a second course, *Intermediate Git*, intended for those who'd been using the version control software regularly for six months or so and were ready to level up their skills. For these two courses, we broke them up into five and eight modules, respectively, that were intended to take up roughly 90 minutes of class time, including plenty of audience interaction and live coding.

Over time, we built the following components into each of our modules:

- **Learning Objectives:** We provided statements up front about the learning objectives so participants would know the purpose of each module and what they should be able to understand, explain, and do by the end of it.

- **Exercises:** While presenting the material initially, participants are always just watching one of our instructors demonstrate live at the terminal, and perhaps following along. Once a concept has been introduced, then it's time for everyone to try it themselves and help their neighbors figure it out.
- **Knowledge Checks:** Before or after an exercise, we'll throw a multiple-choice question up on the screen and give folks ~30 seconds to come up with an answer. We'll then ask for a show of fingers (the number of fingers you raise corresponds to what you think the right answer is), and that quickly allows the instructor to gauge whether or not the class is tracking well. If not, slow down; if so, maybe speed up.
- **List of New Commands:** Each module contains a summary list of all the new commands introduced, along with what they do, for easy reference later.
- **Homework:** We always encourage participants to put these new skills into practice immediately and keep using them throughout the week.

Integrating Feedback

Our courses would not have reached the level of professionalism they have without mechanisms for consistently integrating feedback into what we do and how we do it. Our primary feedback mechanism we borrowed from The Carpentries: post-it notes. When participants arrive, they grab one red and one green post-it note to have on hand. When working on an exercise, if all is well, they stick the green post-it note on the top of their laptop screen. If something goes wrong, use the red one instead. This quickly gives the instructor an idea of how people are doing, and allows someone else on the teaching team to jump in and help out whoever is stuck without that derailing the rest of the class.

The post-it notes serve an additional function at the end of a class period. We ask participants to write something that worked really well, something we definitely need to keep for next time, on the green post-it note, and then something that didn't work so well, something that perhaps we should change up, on the red one. Depending on what the feedback is, it may change what we do for the very next class period, or it may change our direction for a future offering of the course. For the last class period of each course, we also hand out yellow post-it notes, and for that day only the different colors mean:

- **Green:** What was the best part of the course?
- **Red:** What was the worst part of the course?
- **Yellow:** What's something you'd hoped to learn but didn't?

Later in the week our teaching team would hold a retrospective to go over how things went. We would offer constructive criticism to the instructor to improve the delivery of the same module next time. Some of us have backgrounds in education, while others don't, so there would always be good conversation here. Additionally we'd go through the feedback on the sticky notes and brainstorm how to incorporate it into the course content or delivery. Continuous improvement became a hallmark of how we operate.

Scaling Up Delivery

It was clear that there were more folks showing up at Sandia with git questions than the two dozen people we could train each year, so we had to figure out how to scale up our offerings. Step 1: Reserve the biggest conference room on site. Step 2: Tell people to bring their own laptops. This has the added benefit of improving the learner experience, because they're working on their own machines as they like them configured, and they'll have everything set up nicely after the course ends. Step 3: Videoconferencing support. We've had a bit of a love-hate relationship with with video conferencing over the years.
We love how it can connect us to Sandians across the US, or to those in Albuquerque who can't make it out to where we hold our classes. At the same time, each tool has its quirks, but rather than dwell on them, let's talk through how to run a hybrid class with half the participants with you in person and the other half online.

If you're going to go this route, you're going to need one member of your teaching team dedicated to to your video conferencing system throughout the meeting. That means one less helping hand for problems in the room, so it may mean needing to expand your teaching team. It's definitely worth it to have someone in the room dedicated to monitoring the audio/video connection, plus all the conversation that happens in the chat. In practice it seems like one helper online can cover three to four times as many participants as a helper in the room.

How do you handle post-it notes virtually? Pick a red and green emoji to use in their place and watch the emojis roll in when working an exercise. What about knowledge checks? When you ask for a show of fingers, just have remote learners type their answer in the chat. It's not the exact same experience as being with us in person, but there doesn't seem to be too much lost, and participants appreciate the convenience of being able to join from wherever they are. By opening up our advertising to all of Sandia, and taking the steps mentioned above to scale up to the increased demand, we grew to the point of serving roughly 200 Sandians annually, plus all those they share our materials with. That's hundreds of people a year who carve 8–12 hours out of their busy schedules to devote to developing and mastering new skills in our courses.

This year we're in the midst of another substantial shift in how we do training. We grew to the point where we couldn't grow further, and as a teaching team we don't have time to develop all the other courses we see could meet the growing needs around us. To that end, we're transitioning our courses into self-paced online learning modules. If you're familiar with the concept of massive open online courses (MOOCs), that's where we're headed. We'd like all Sandians (and eventually anyone) to be able to access and work through our content whenever, wherever, and at whatever pace works best for them. We're hoping to realize this dream via Moodle, an open-source education platform. Stay tuned for how this next shift continues to improve developer productivity in the coming years.

Discussion

Okay, so we have plenty of experience offering training in version control, but what impact has that had on developer productivity? Let me share a brief story. Some months ago I was sitting in on a meeting with the Advanced Simulation & Computing (ASC) DevOps team, where they were reporting specifically on the various steps they've taken to build sustainability into their efforts. When they got to talking about workflow, I was pleased to hear things like

- All work starts as an issue in GitLab.
- That issue becomes a feature branch.
- That feature branch becomes a merge request.
- That merge request gets reviewed and approved by at least two other team members.
- That merge request undergoes continuous integration testing and can be automatically deployed (CI/CD).

When the meeting was over, I reached out to one of the team members, who I recognized from our *Intermediate Git* class a few semesters back, and said, "Hey, it was really cool to see you guys took our suggestions to heart." His response: "Your lessons were truly invaluable to getting me up to speed and helping share a working paradigm with my teams."

That's one little vignette, but we hear feedback along those lines from folks across the labs frequently. We often meet people and later find out they were pointed to our materials by a past participant, and are grateful for how it's improved their daily workflow. Additionally, I can point to a handful of people who have taken our courses and then taken those skills and materials back to their teams. Indeed, it *is* challenging to track the spread of knowledge and best practices in this real-world setting, but we've seen first-hand the progress people are making. In your own training efforts, make sure you design for repeatability, just as you would when designing software. Make yourself a checklist to ensure each module has all the essential elements. Build a culture of continuous improvement into your teaching team, perhaps using something like a productivity and sustainability improvement plan. Consider hybrid delivery means or something like a flipped classroom to spread your reach. Whatever you decide to do in this arena will pay dividends for years to come.

Acknowledgements

This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government. Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. SAND2020-6643 C