GSA Office of Technology Transformation Services

An overview of TTS DevTools

Revised 2025-02-20

Why we need DevTools

Our core problem space: Developer experience in government

Productivity

Civic tech software engineers are most productive when they can stay focused on building solutions that solve the public's problems.

Missing tools, lack of common components, and the steep climb to understand compliance block their path to getting started.

Fragmented tooling and friction from compliance limit their ability to deliver efficiently.

Compliance burden

Code is how software and security engineers build and secure digital services.

Current compliance patterns force engineers out of code and into 300+ page documents.

We are wasting cybersecurity expertise on huge text documents, leaving little time to improve security through code.

Workforce

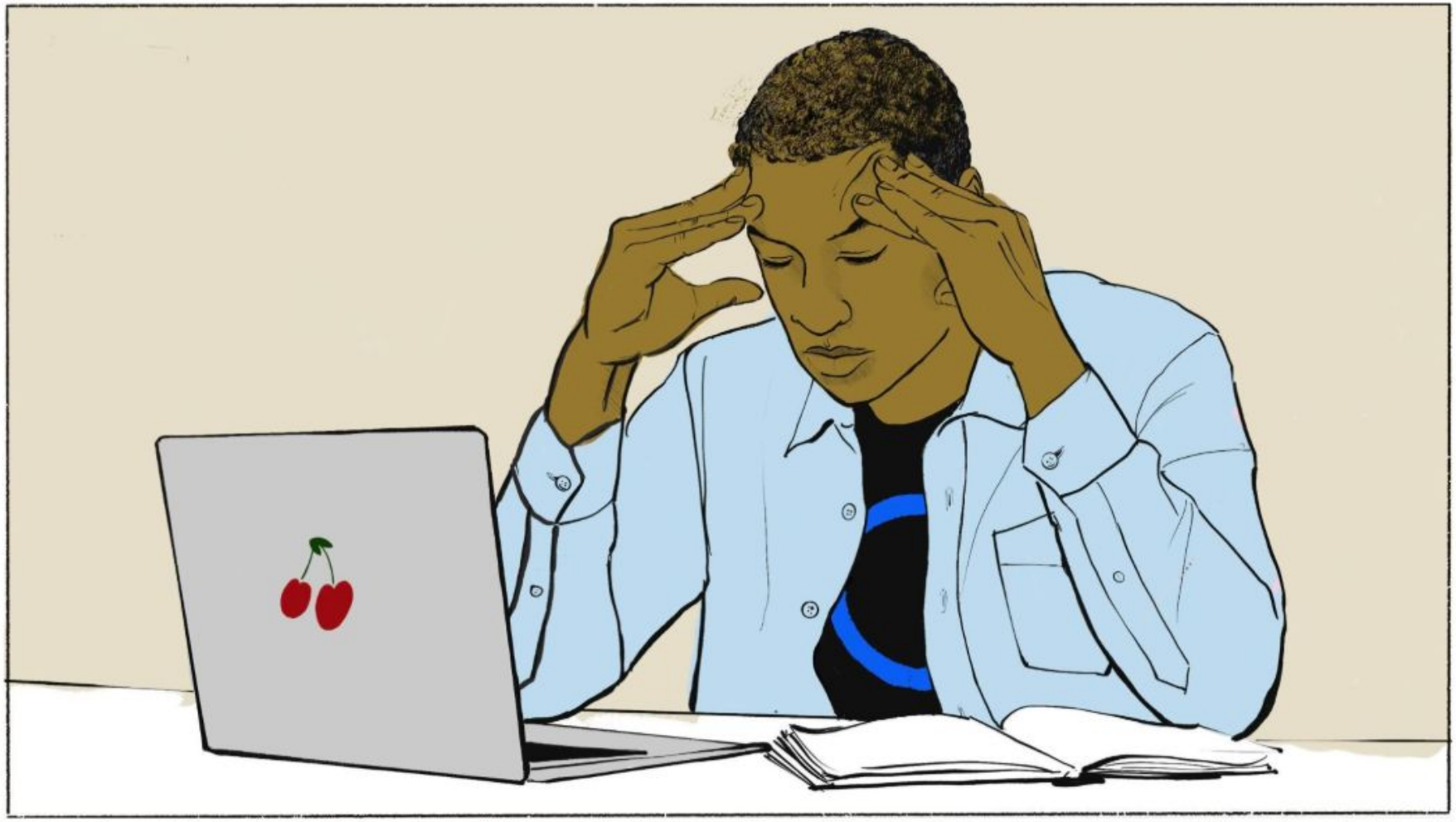
Software and security engineers are fluent with modern development tools. Forcing them to learn old methods and manual processes slows them down, and prevents government from accessing the benefits of modern approaches.

Cybersecurity SMEs are hard to hire and retain. It is counterproductive to task them with work that doesn't capitalize on their skills.

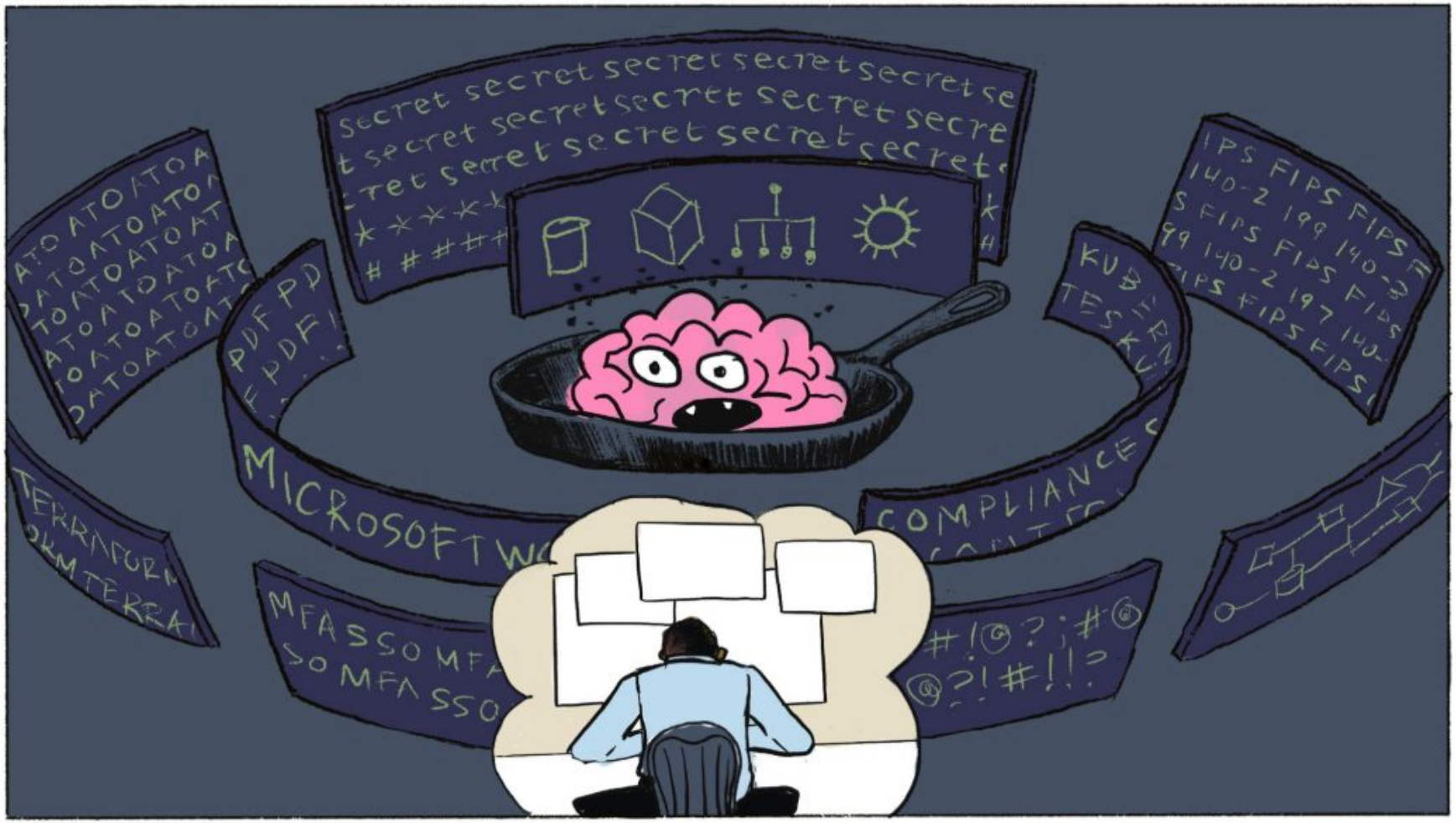
We must pave a road, making the *right way* the path of least resistance.

Time for a story.

Powered By Technology Transformation Services | Developer Tools

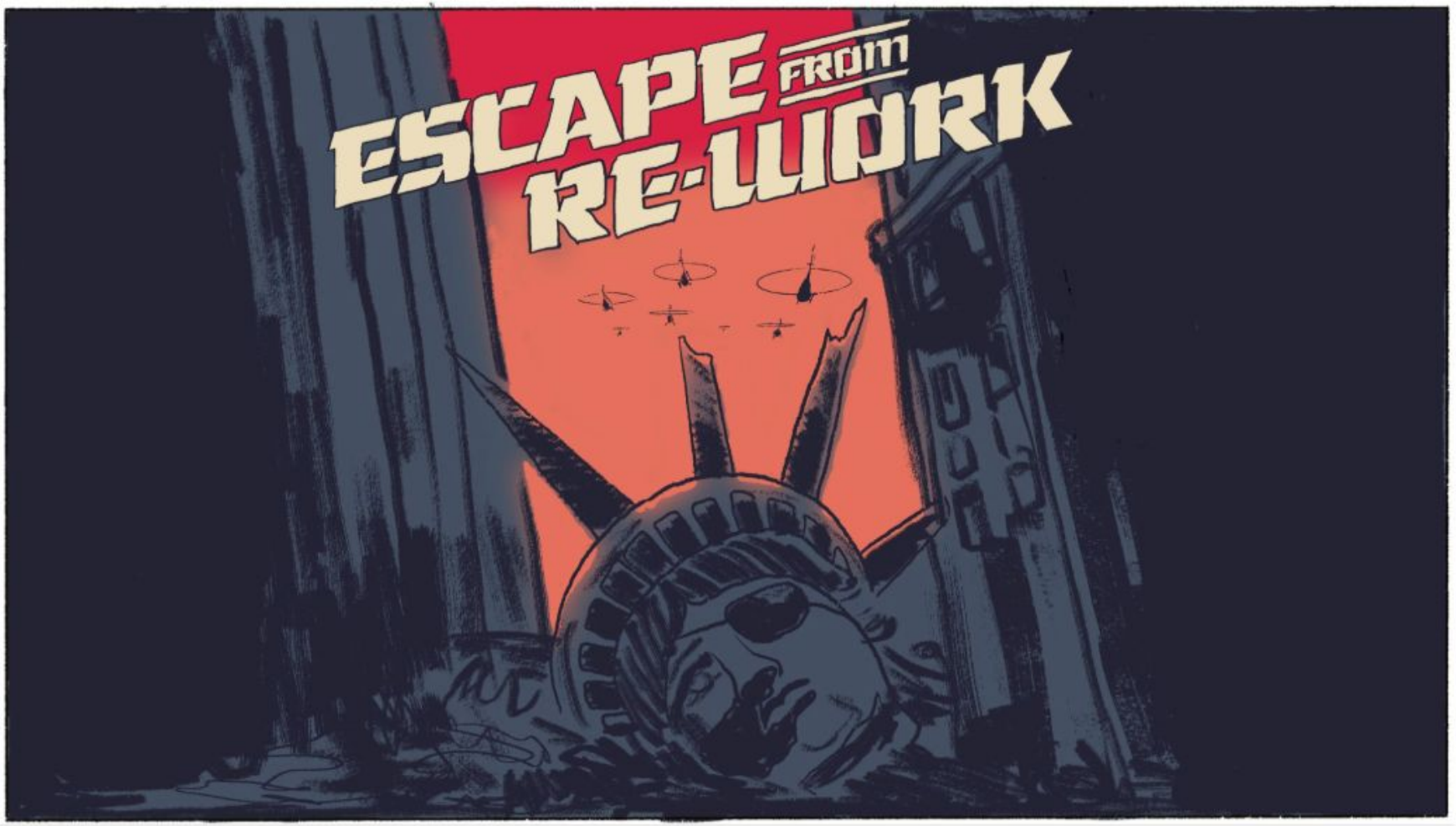


This is Lee, a developer on a TTS team. His team is launching a new shared service in partnership with its initial customer, which supplies crucial functionality and access to the American public. On the heels of a security incident, they are looking forward to leveraging the new service for a more robust, secure web presence. Lee is exhausted just thinking about it.

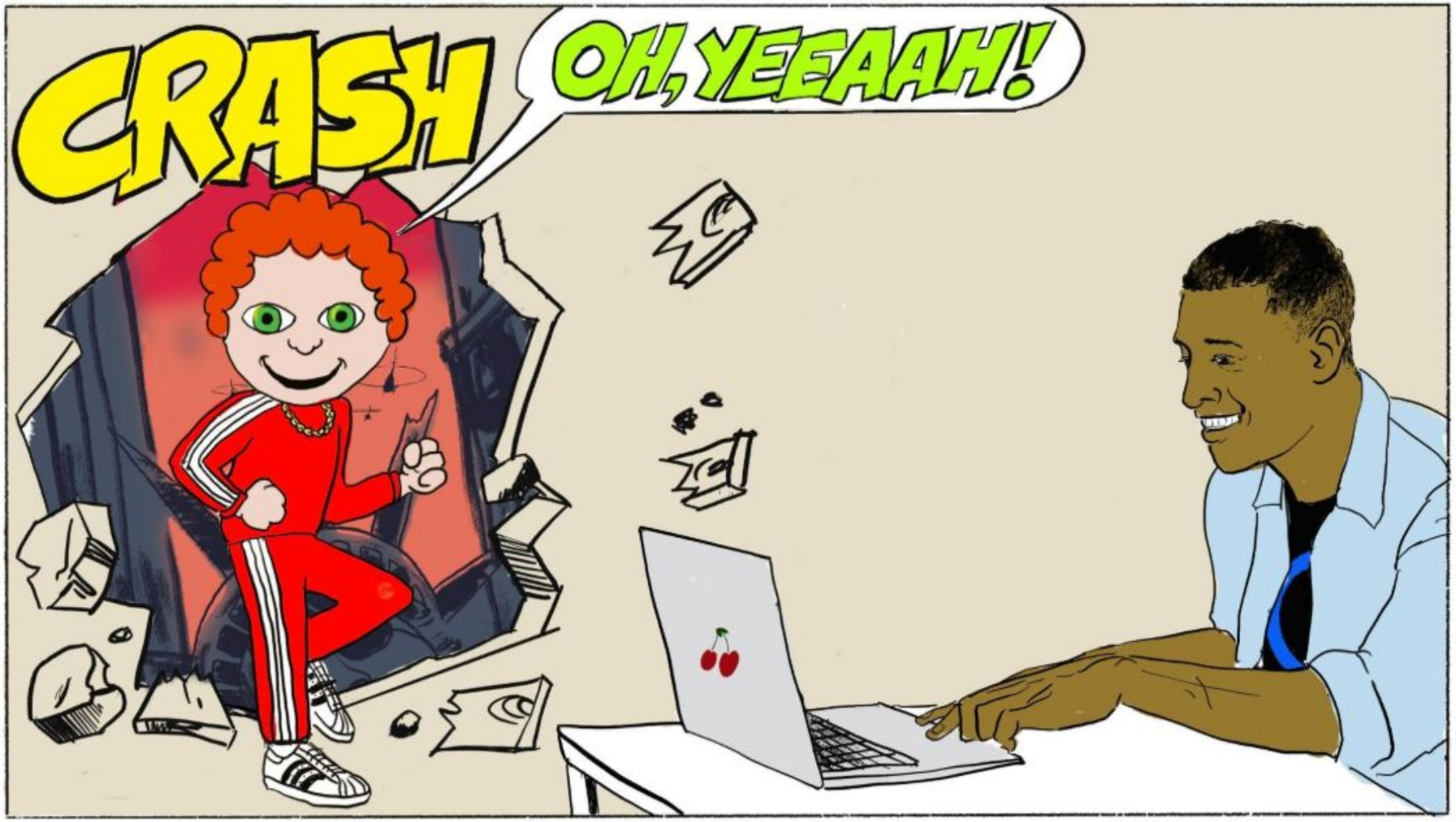


He thinks back to the complexity of his team's last deployment, during which they assembled a system out of whole cloth. He's dreading the beginning of a new digital service and the seemingly endless processes, like searching for reusable components and authorized services, and working through various levels of compliance.

It feels like, with every new project, he and his team are dropped into a dystopian landscape with fresh new obstacles, and they have to assemble their tools from their surroundings each time. They'll need to search for and glue together different parts, all while looking carefully at each one to see if it's well-supported and fits the needs of the service. Some of them are out there, but not necessarily in a centralized location that is set up for re-use. And forget about all the compliance documentation.

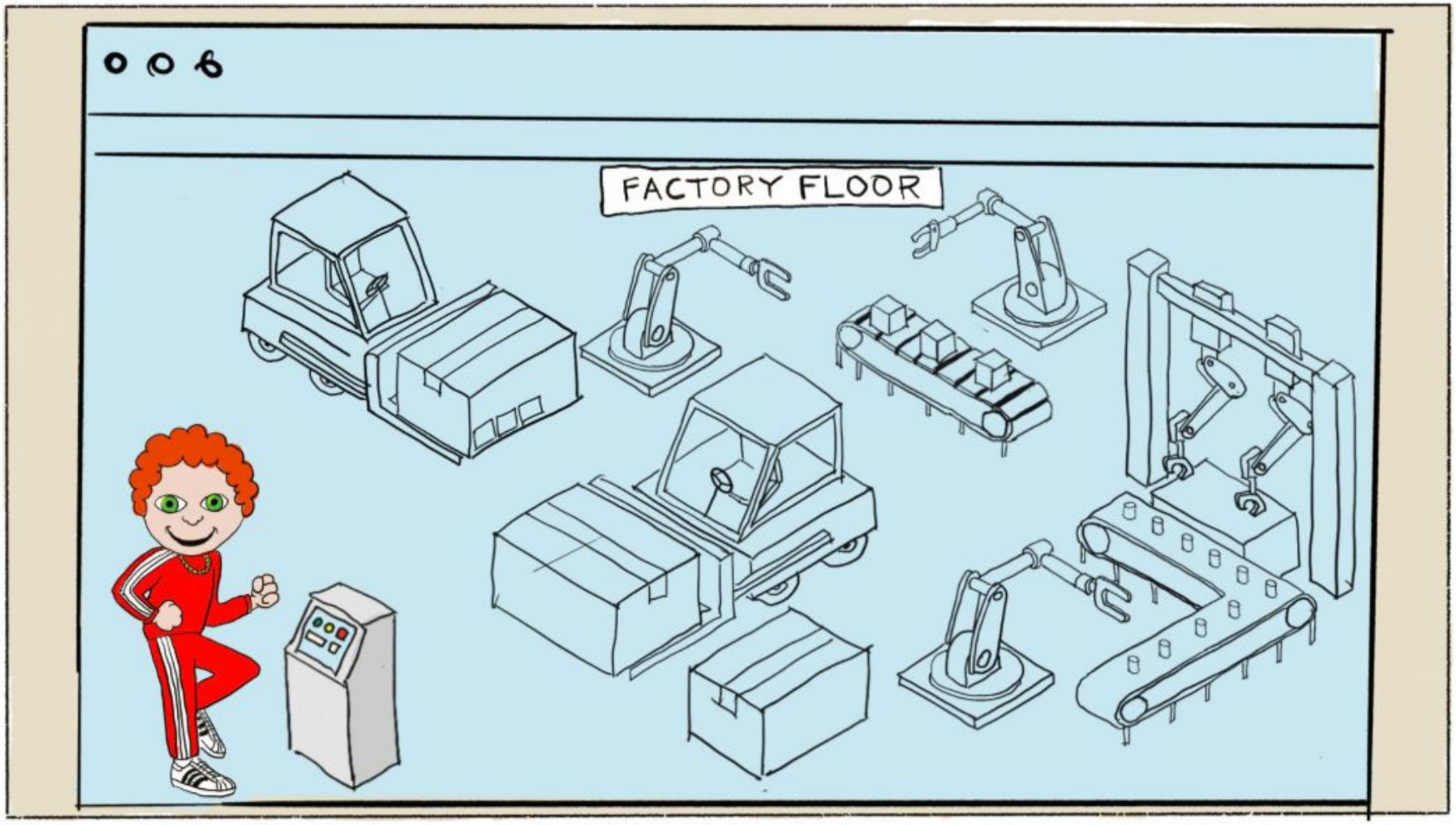


He wishes there was a better way.



Fortunately, Lee's team has opted to take part in the new pilot of TTS DevTools, which is expressly designed to reduce the burden of starting and operating digital services within TTS.

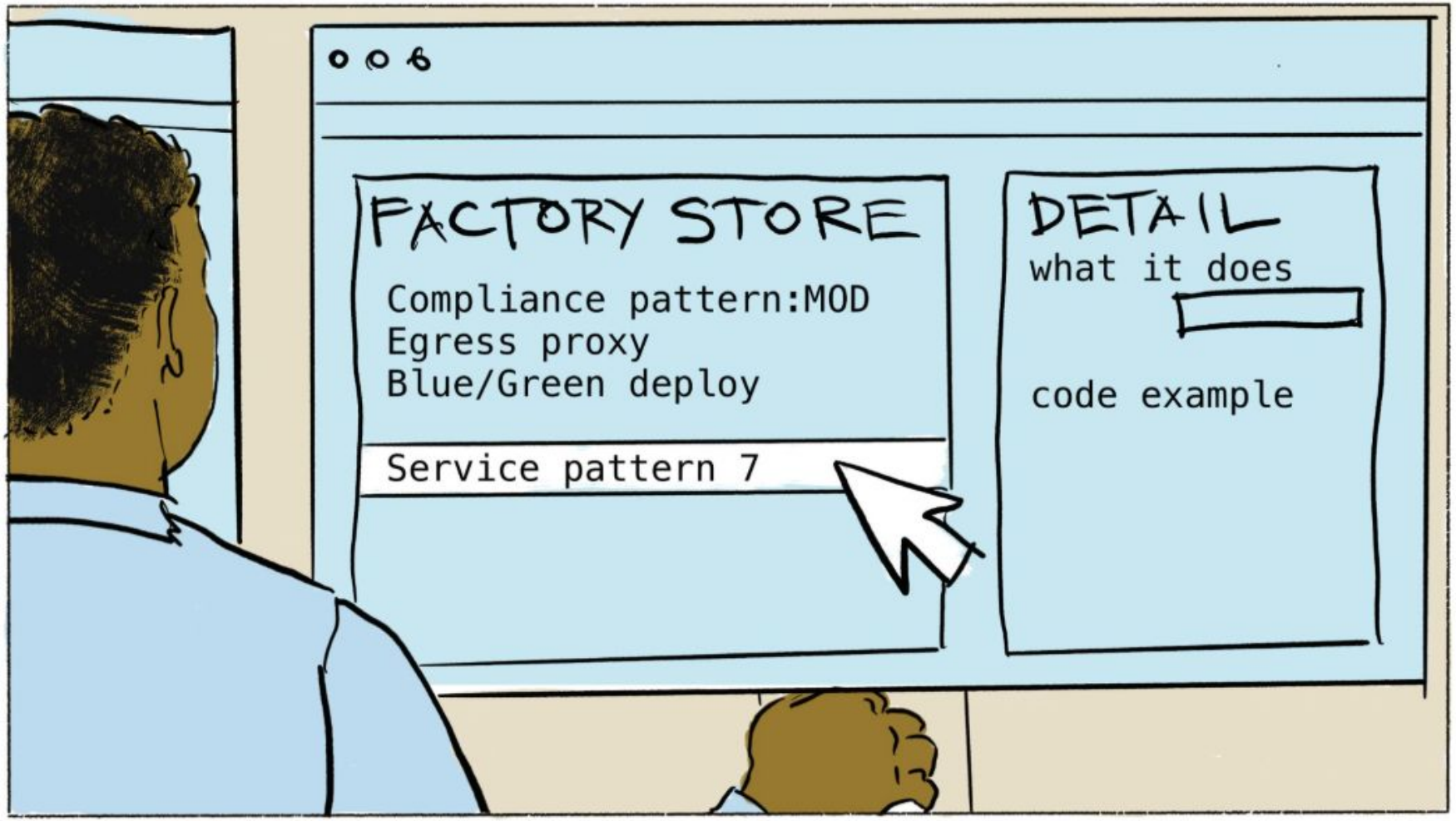
At this point, DevTools Runner crashes through the wall of Lee's home office like the Kool Aid man, and shares the DevTools vision: a clean, functional, and organized space to create and run software projects, and a running start for technologists (with a helping hand).



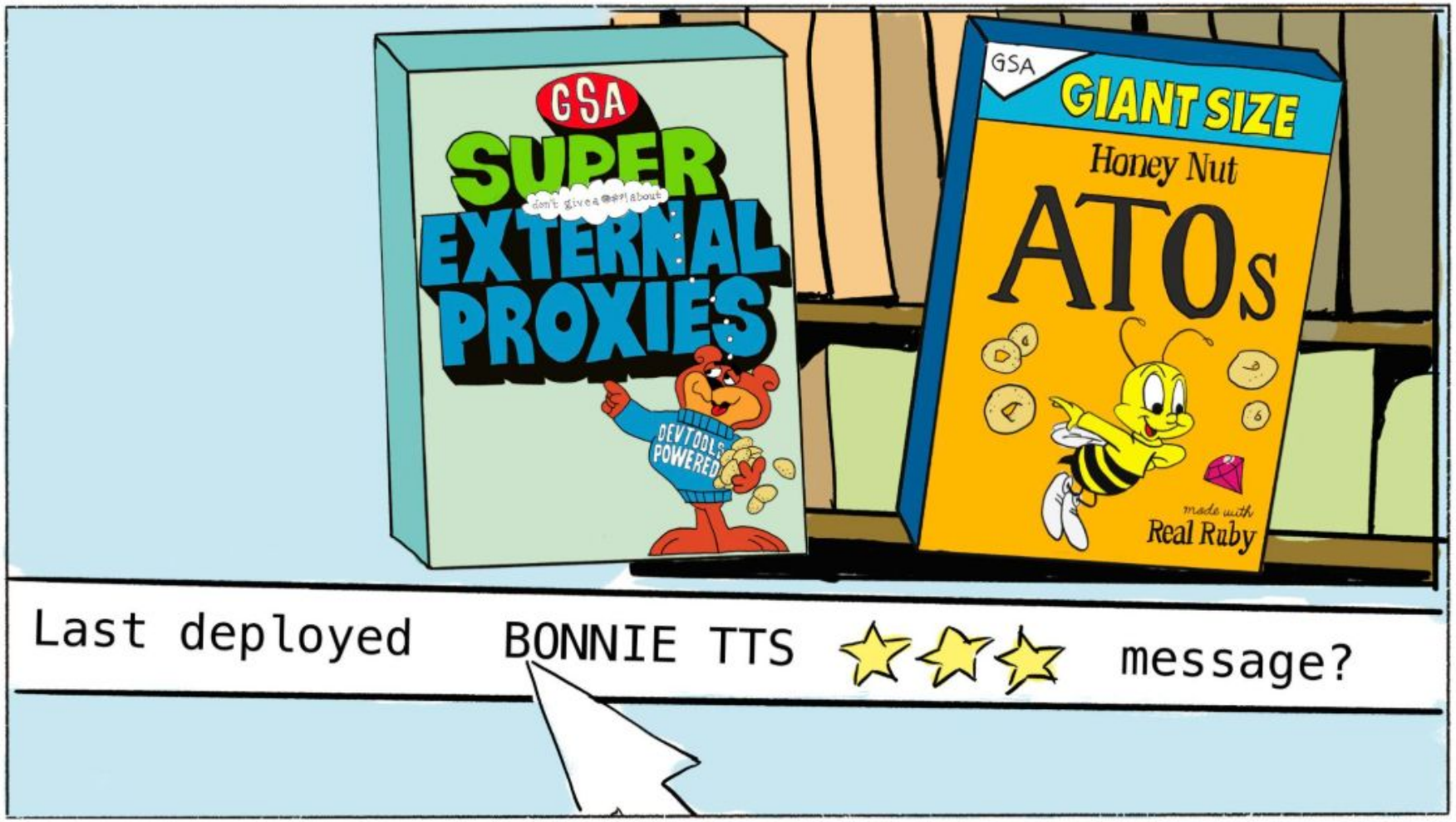
DevTools Runner introduces Lee to The Factory Floor, built on Gitlab Dedicated for Government. It has every bit of heavy machinery, tooling, and Infrastructure Lee needs, all in one dedicated place.

Next, DevTools Runner introduces Lee to The Factory Store, which is a fully-stocked catalog of software components and patterns tailored for Federal digital services. It is always open.

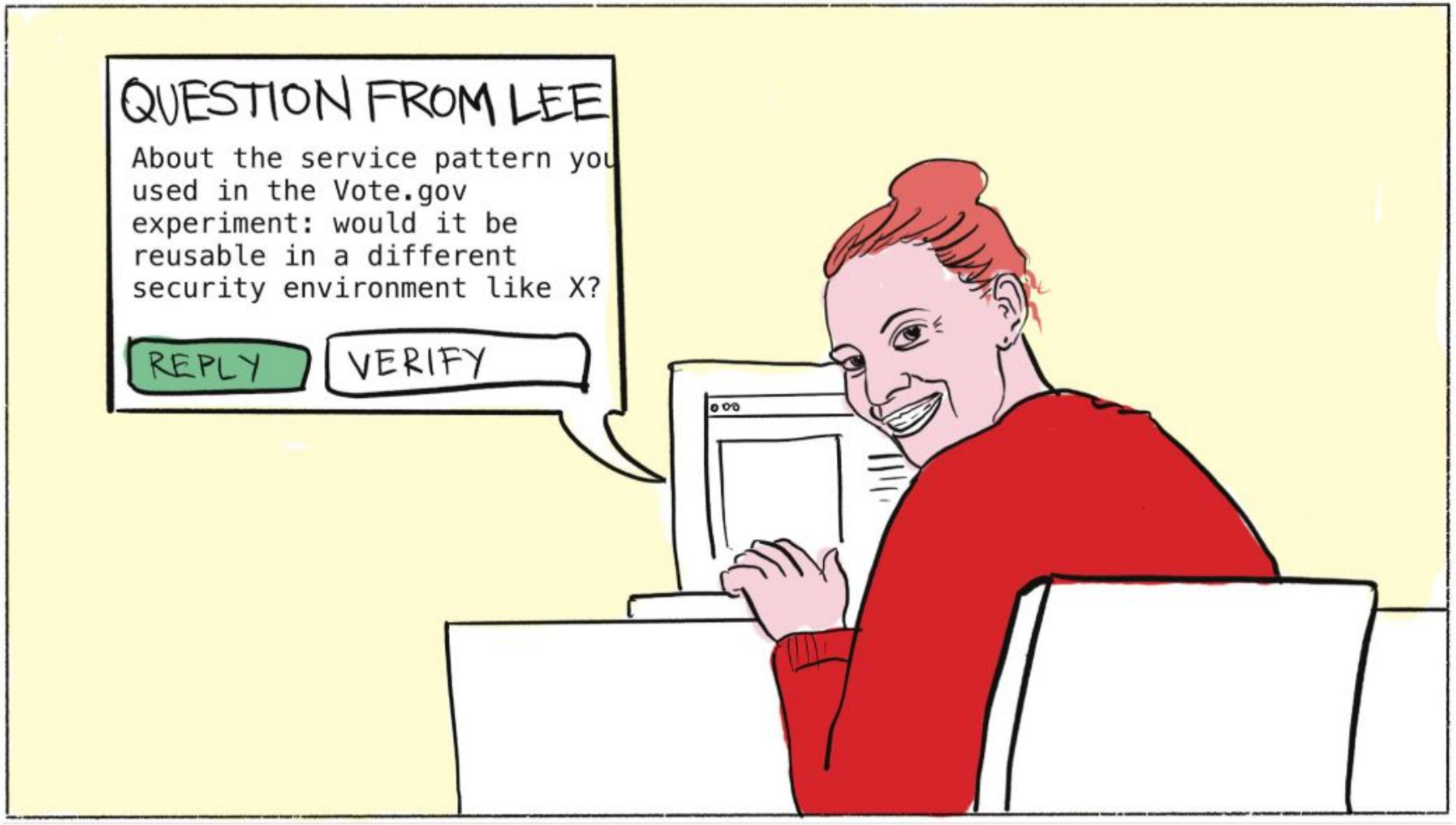
Lee goes through the onboarding process with his whole team, and can see some obvious advantages. He feels optimistic. Security guardrails and compliance check boxes are baked in, components are available off the shelf, and it is all aligned to the way teams build services in TTS. It looks like maintenance will be easier in the long run.



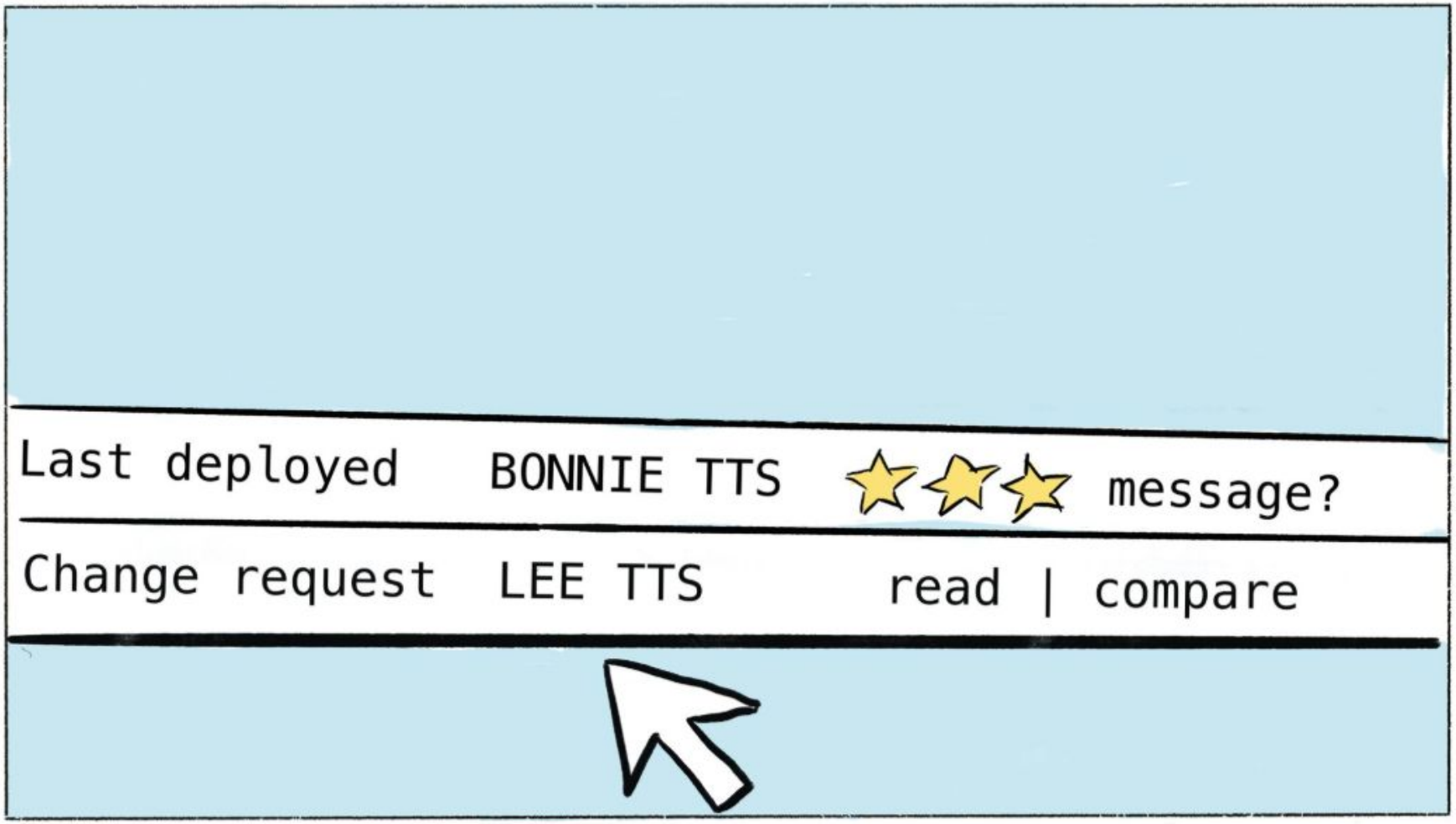
Based on the type of sensitive data the new service will be handling, Lee immediately sees a service pattern he can re-use from another TTS project.



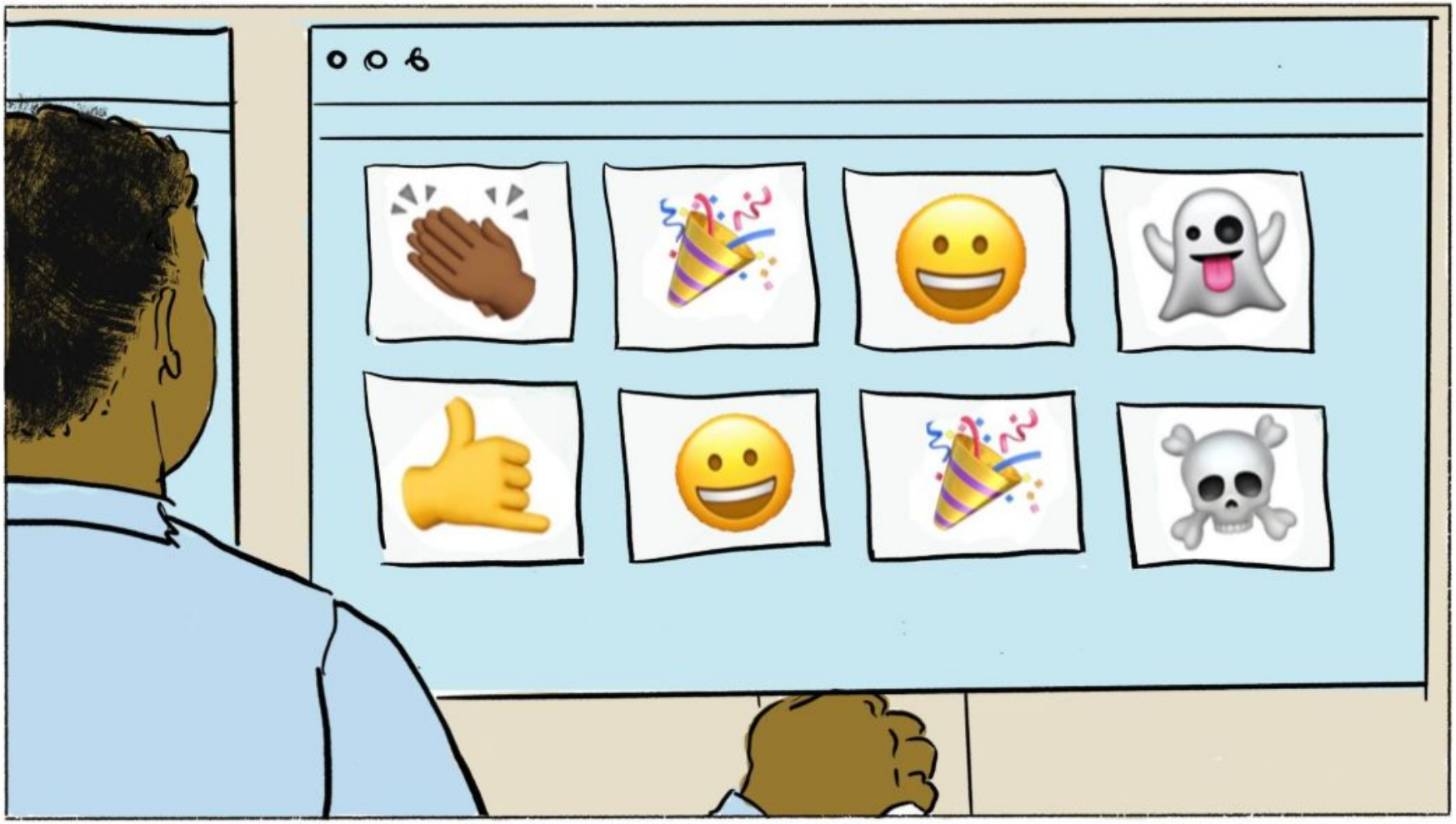
The compliance patterns, forms, and components are clearly visible in the catalog, and Lee can even preview them within the DevTools system. It's almost like pulling a box off the shelf at the store and reading the ingredients. He can scan and browse the histories of their previous deployments, and even read commentary left by other feds. Lee has a question about the compatibility of one of the patterns, which he hopes will reduce some of the immediate compliance burden on the project. He's able to leave a question for Bonnie, the last fed who deployed it.



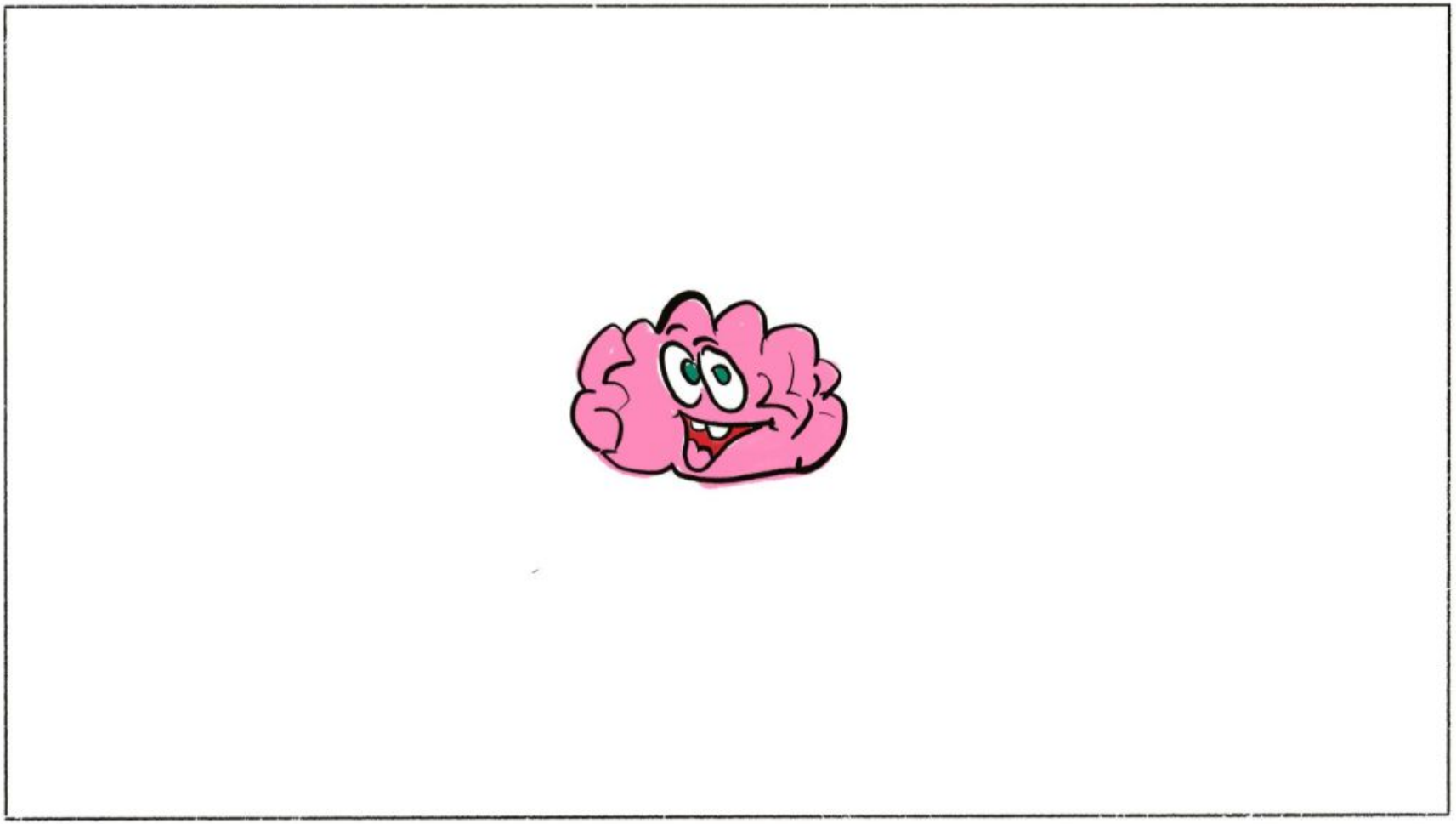
Bonnie isn't logged into the platform at the moment, but the question reaches her through the Slack integration, and she is able to answer Lee's question there. She verifies his concerns, and suggests an edit that was successful during her team's work with Vote.gov. Bonnie's comment is displayed within the factory floor setting, and becomes an artifact that is visible within the system.



Lee sees her answer to his question, and decides to use this component within the framework his team has chosen, after making some tweaks based on Bonnie's suggestions. He opens a change request with his tweaks, leaving an explanation of the changes, so that the next fed can clearly see what the new version does. The change is merged and Lee updates his code to use the new component..



Lee and his team assemble a working demo for the next meeting. They are able to pull this together pretty quickly, since all communication, commits, and deployment are happening within the same system, and all components are readily available off the shelf. Lee and his team are happy because the experience was awesome. The American Public will benefit from the service as agencies adopt it to add critical functionality and improve customer experience..

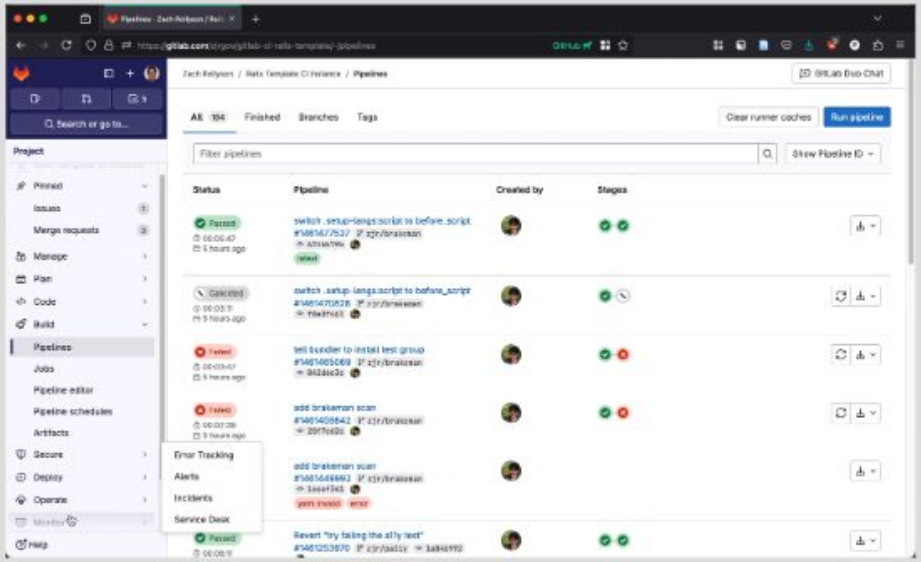


This is Lee’s brain on DevTools.

Factory Floor

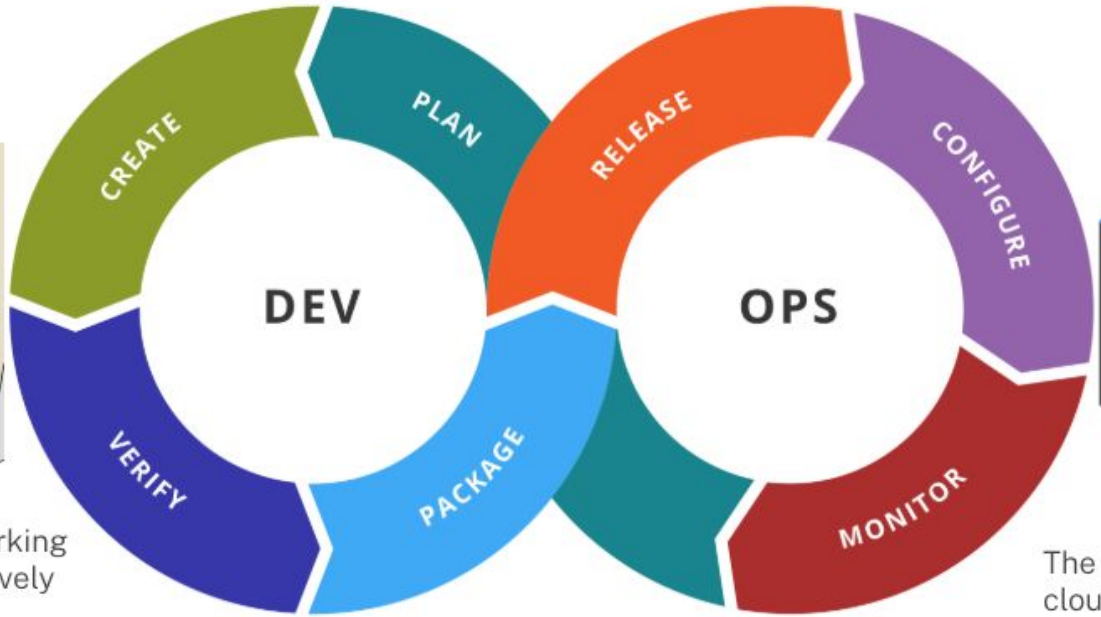
What if developers had the secure software development tools they need, authorized at a Moderate Impact level, and able to fill the gap between their local workstation and production?

What if security guardrails were baked in and high level security dashboards allowed efficient oversight and vulnerability management?



Development as imagined

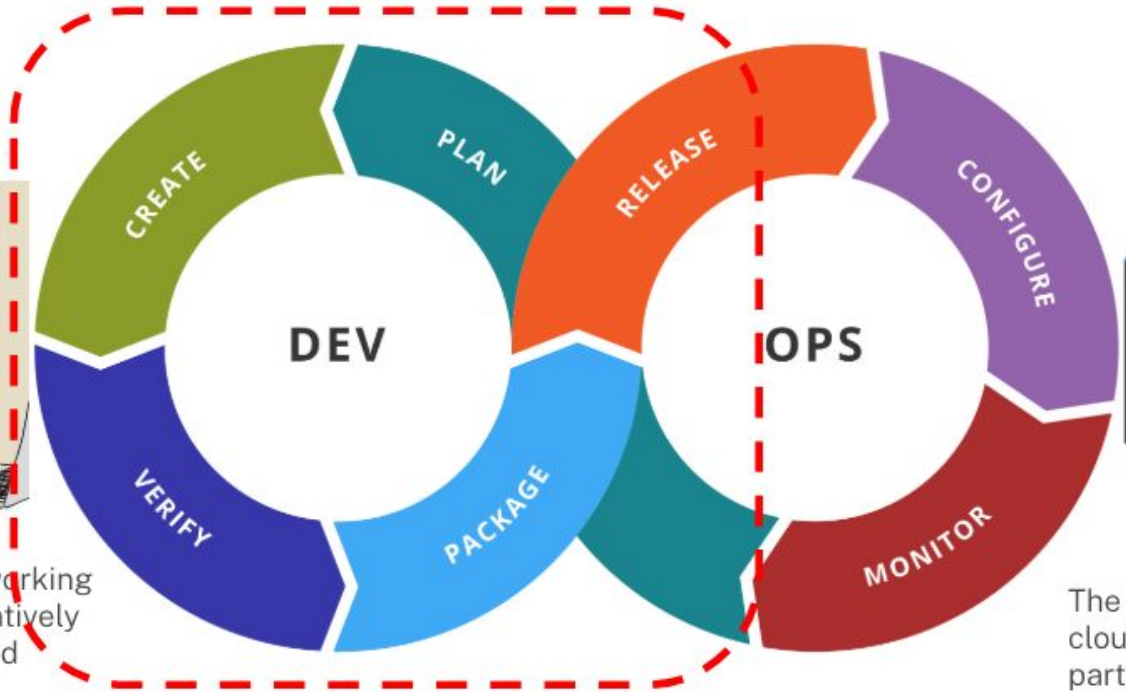
A civic technologist, working as part of a team, iteratively plans, writes, tests, and releases code.



The code is deployed to cloud.gov where is lives as part of a digital service providing value to the public!

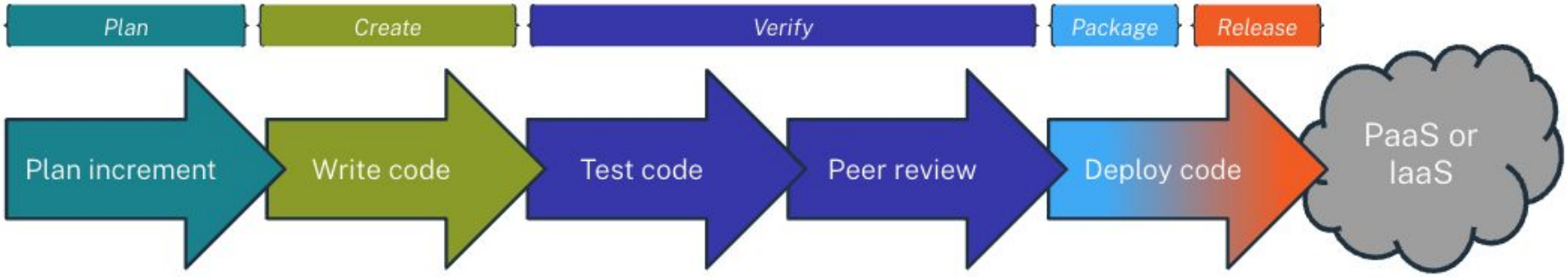
Development as imagined - The gap

A civic technologist, working as part of a team, iteratively plans, writes, tests, and releases code.

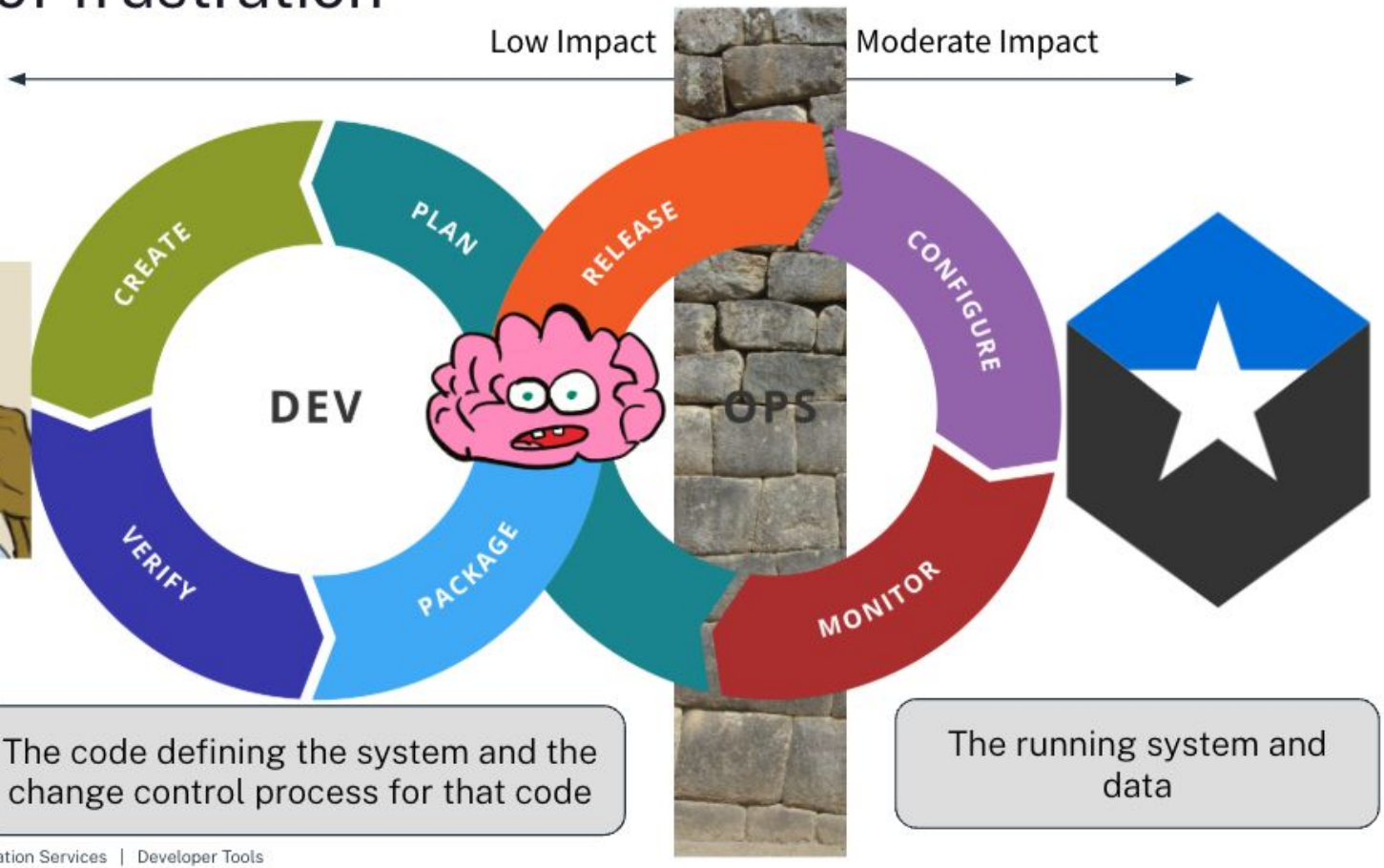


The code is deployed to cloud.gov where is lives as part of a digital service providing value to the public!

Development as done today



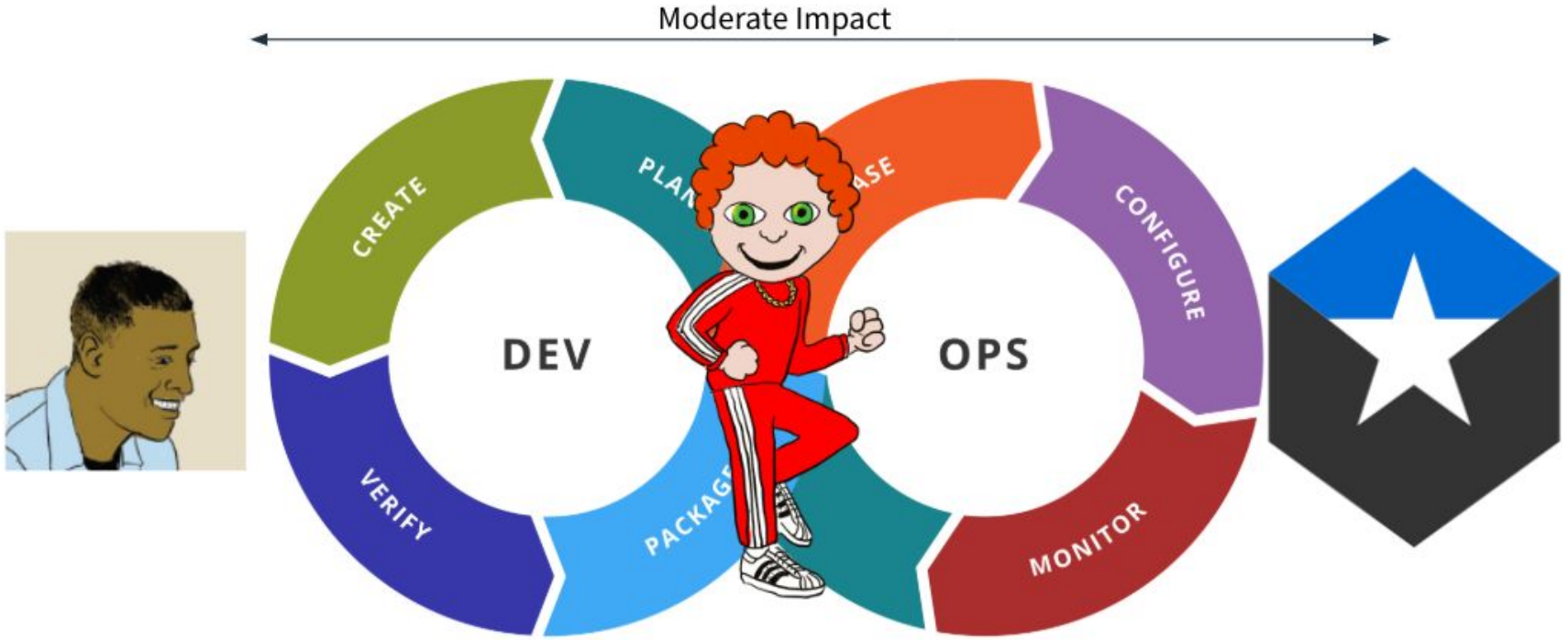
The wall of frustration



The code defining the system and the change control process for that code

The running system and data

Secure software delivery without arbitrary walls

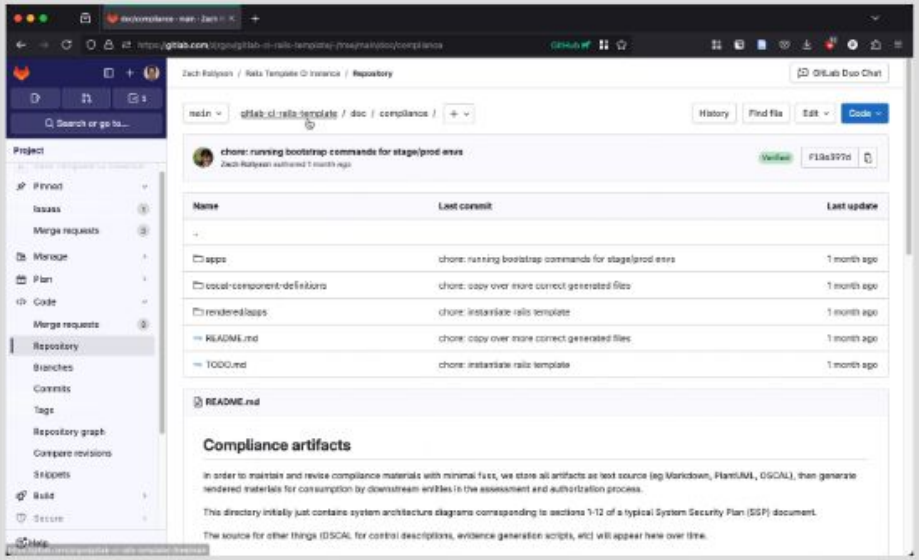


The code defining the system and the change control process for that code and the live system and data all at the same FIPS 199 Moderate Impact level

Factory Store

What if teams could get a running start on every project they start, using shared components assembled in an opinionated way, ready to deploy to cloud.gov, with compliance documentation built-in?

What if updates to components were presented automatically to teams while allowing the teams to decide if and when to update those dependencies?



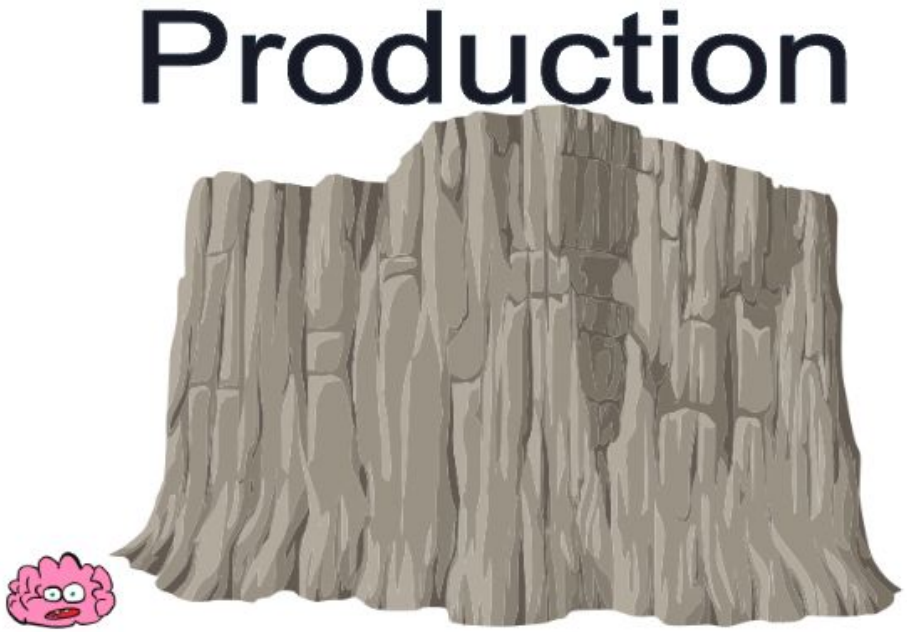
Building a new service now

Questions:

- ➔ Where will it run?
- ➔ How will I deploy it?
- ➔ How will I implement tricky controls like egress filtering?

Work:

- ➔ Start the application code
- ➔ Start the infrastructure code
- ➔ Start the CI/CD automation code
- ➔ Integrate the separate tools
- ➔ Try to find up-to-date compliance templates (and oh so much more compliance work...)



DevTools shrinks the mountain and brings stairs

Less total work up front:

- ➔ New projects start with basic requirements like a README, security policy, and change controls
- ➔ Application code starts with common requirements like multilingual support and is coded to use USWDS
- ➔ Automation for code quality, accessibility, and security testing is in place and running
- ➔ Infrastructure as code to deploy smoothly to cloud.gov is included

Less work during implementation:

- ➔ Additional DevTools components can be fit into the project with minimal effort
- ➔ Agency required integrations for logging, tooling, and more can be streamlined
- ➔ DevTools components represent well tested patterns that are kept up to date and less likely to come with surprises

Compliance as Code

What if teams could use common software development processes to document, test, and continuously attest to controls? (Instead of authoring Word documents.)

What if teams could invite compliance staff left into the development process and could automatically ship OSCAL compliance artifacts to their agency OCISO or to FedRAMP?


```
{
  "assessment-results": {
    "import_ap": {
      "href": "trestle://assessment-plans/rspec/assessment-plan.json"
    },
    "metadata": {
      "last_modified": "2024-09-16T15:31:52+00:00",
      "oscal_version": "1.1.2",
      "title": "Automated Testing Results",
      "version": "2024-09-16T15:31:52+00:00"
    },
    "results": [
      {
        "description": "Results for tests:\n\n  × /documents POST /create with valid parameters logs the use of this privilege
        "end": "2024-09-17T13:16:30+00:00",
        "findings": [
          {
            "description": "ac-6.9: \"logs the use of this privileged function\" (./spec/requests/documents_spec.rb:87)",
            "implementation_statement_uuid": "c2db0c86-cfa5-4fb5-9f92-66a066bf5604",
```

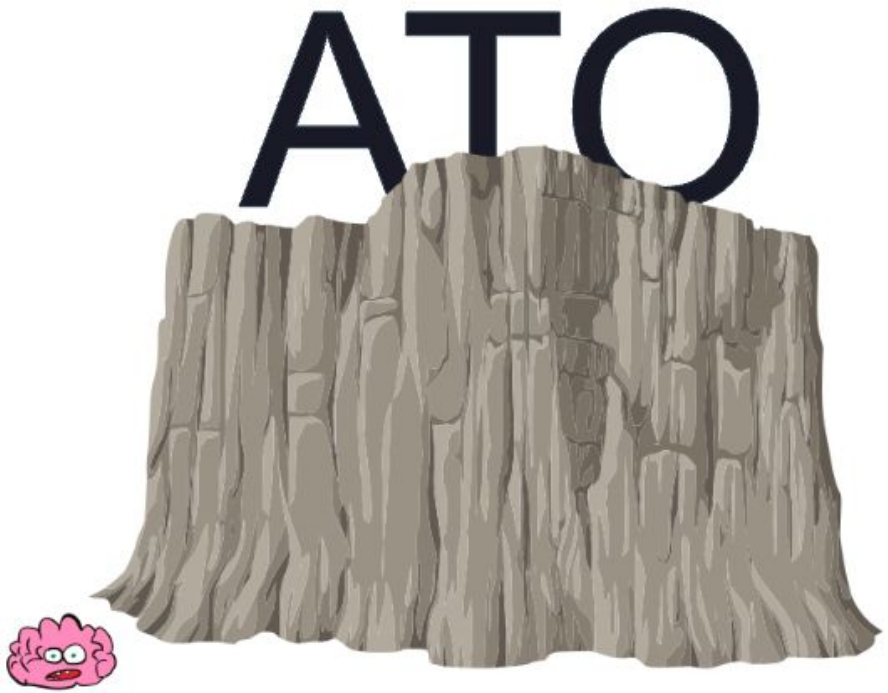

Securing an initial Authority to Operate (ATO)

Powered By Technology Transformation Services | Developer Tools

Simply climb the mountain in one step

Step 1:

Gather documentation and control templates, implement controls and document them in an ever growing Word document, try to keep up with changes in the system, try to get feedback from your ISSO and ISSM, think you're done, find out you are not even half done, try to copy language from other SSPs in the agency, find out they are all inconsistent, reinvent the wheel multiple times, prepare for assessment, find out about 10% of your controls need changes in implementation and/or documentation, *don't look down... do not look down*, finish assessment, address numerous findings (many of which could have easily been avoided with the right information up front), and finally reach the top: **A signed ATO!**



DevTools shrinks the mountain and brings stairs

Less total work up front:

- ➔ Agency or FedRAMP SSP templates in code
- ➔ Prefilled sections based on leveraged/inherited controls
- ➔ Implementation guidance available where it is most useful
- ➔ Continuously generated OSCAL SSP with PDF alternative

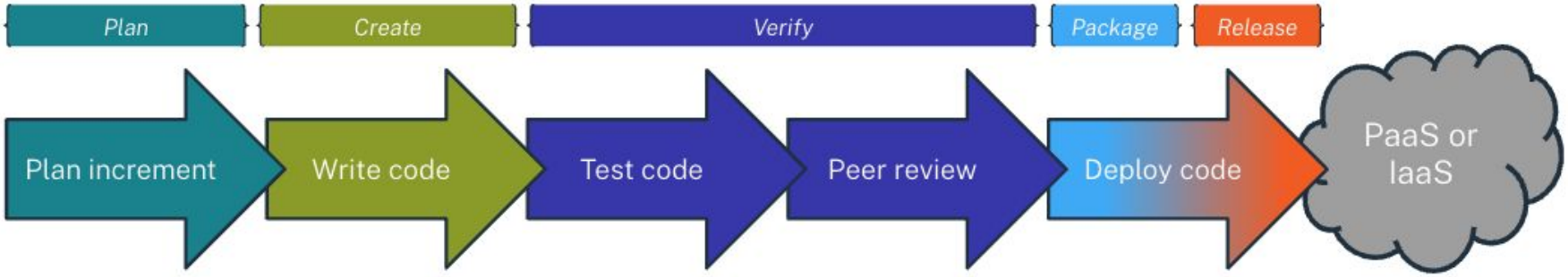
Less work during implementation:

- ➔ Code to address a control, tests that attest to meeting the control, and control documentation are all iteratively developed in the code base
- ➔ Implementers, who have the most system context, directly engaged with the control language
- ➔ No need for a "compliance translator" to copy information into the SSP and other documents

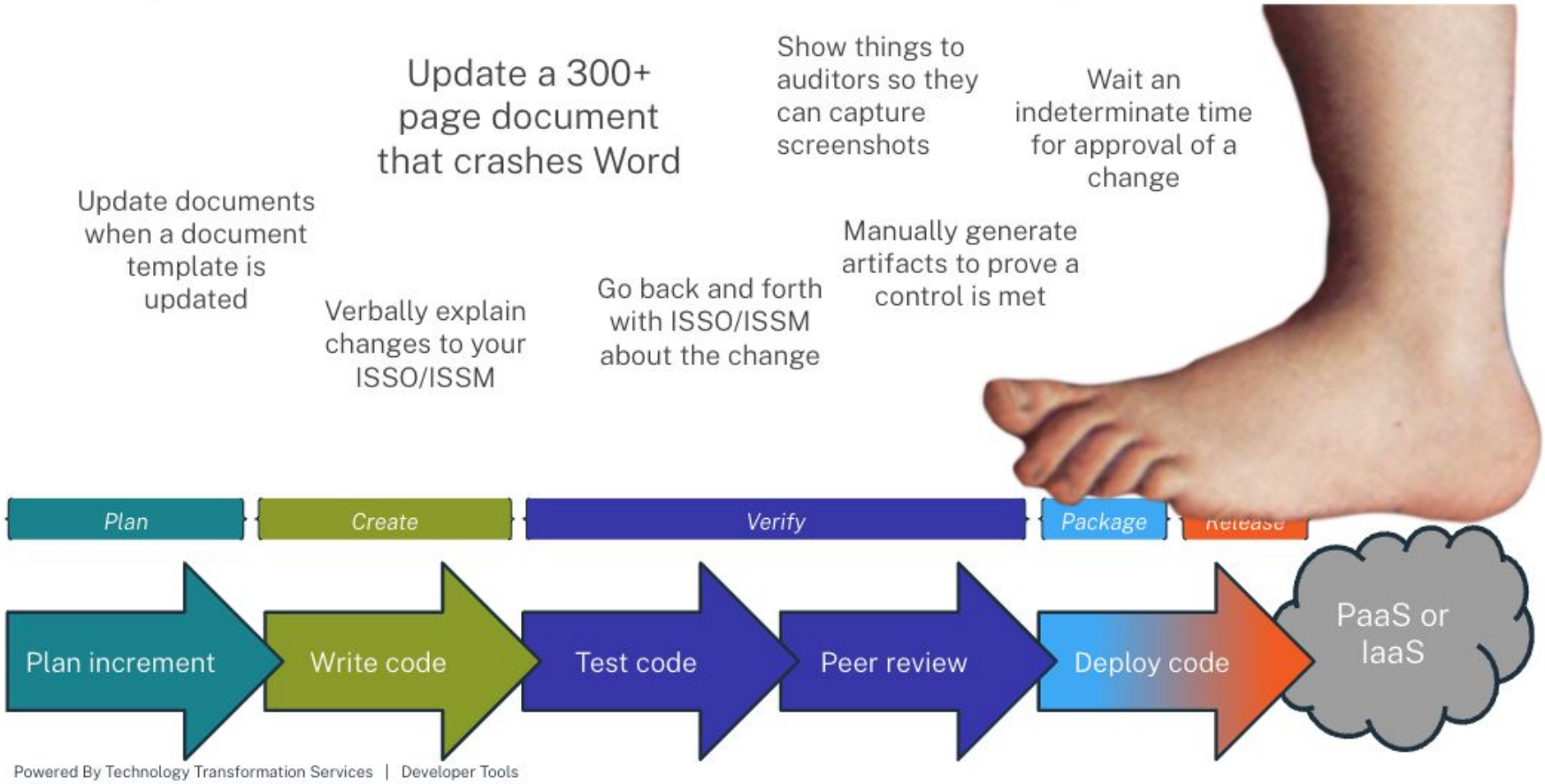
Keeping up with compliance

Powered By Technology Transformation Services | Developer Tools

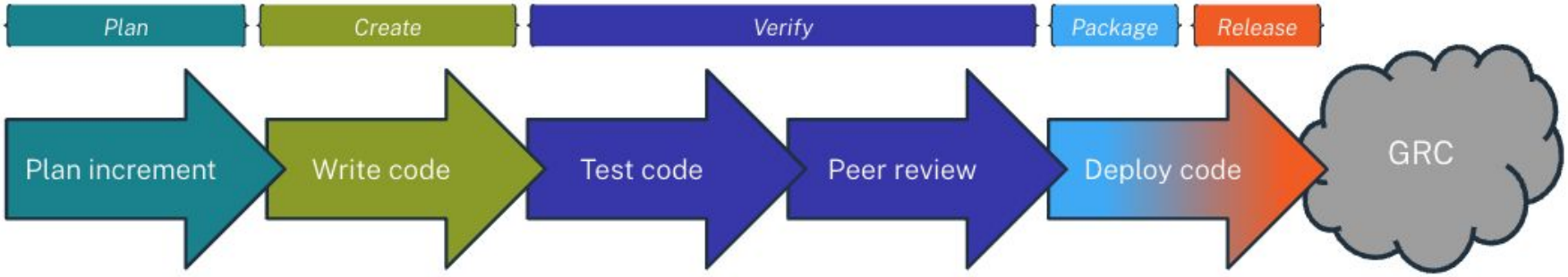
How compliance fits into the development lifecycle



Compliance doesn't fit into the development lifecycle



Developer centered compliance as code



Compliance becomes just another facet of the issue.	Developers write code, matching compliance documentation, and tests to ensure the control is met.	Compliance tests run and documentation is validated.	Compliance staff review changes and provide feedback following normal peer review processes.	Updated OSCAL artifacts (updated SSP, assessment results, etc) and alternate PDFs are generated.	OSCAL is pushed to the agency or **FedRAMP** governance, risk, and compliance (GRC) service.

How DevTools products map to the problem space

	Productivity	Compliance burden	Workforce
Factory Floor	A complete toolset to develop and collaborate within, improving focus and reducing friction.	A common working space centered on code with security oversight interfaces to collaboratively manage risk.	New hires get the tools they need on day one, reducing the stress of getting started.
Factory Store	Components dramatically reduce startup times and ongoing burden, with less total code to write and maintain.	OSCAL baselines and pre-filled common controls reduce start up times, with less compliance documentation to write and maintain.	Paved roads keep the focus on solving problems for the public, increasing satisfaction for civic technologists.
Compliance as code	Tools and components fit within the development process, improving flow.	Compliance docs and tests live with the code and generate artifacts automatically, allowing security skills to be focused on practical security instead of generating documents.	Allows for a more gradual guided path to navigate ATO processes, reducing feelings of being overwhelmed.

Measurement of impact and adoption with customers

How will we know how we're doing?

QUALITATIVE: We'll ask.

- Discovery and onboarding interviews with initial benchmarks
- Ongoing individual temperature checks
- Office Hours, regular communal feedback gathering

QUANTITATIVE: We'll track.

- Counting inherited, implemented, and documented controls as a baseline, we'll compare customer burden with and without DevTools (What did the customer **not** need to do)
- Additional quantitative metrics will focus on success within the system

ITERATION: We'll experiment.

- Exploring use of tools like DORA-4, NASA TLX for cognitive load, or the "SPACE" model for building developer productivity metrics
- Looking at existing signals and making sure they tie to outcomes, and can be linked to improvement