DRAFT

Initial Findings & Opportunities

Discovery Phase
May 7, 2021
Week 5

18F In partnership with Wisconsin

Today’s goals

Recap (5 mins)

Modernization approach (5 mins)

Initial findings & opportunity areas (20 mins)

Next steps

Input and discussion (15 mins)
Provide input on initial findings/opportunities
Review initial direction, redirect if needed

An incredible DWD team

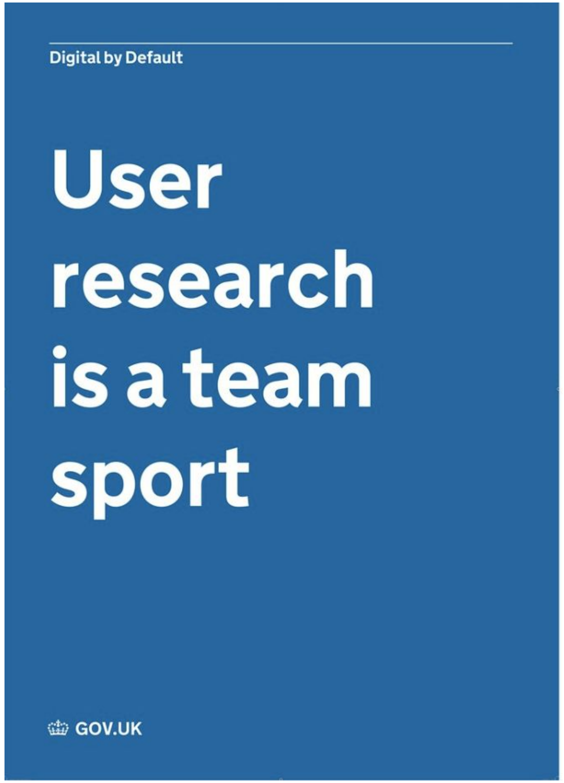
DWD

Department of Workforce Development

Upcoming milestones

5/10-5/14	May 18 & 19	5/24 – 5/28	5/31 – 6/11	6/15 – 6/30	7/1 – 7/31	7/30 – 8/31
Prioritize UI outcomes, Build Roadmap, draft solicitation	**De-risking Acquisition Workshop**	**Review RFQ & Release draft for comment by 5/28, open for 2 weeks**	**Final review and adjustments to RFP**	**Publish final RFP**	**Receive questions & proposals**	**Interviews & selection**
• Drafted RFQ • Drafted roadmap • Build vs. buy discussion	• Refine Draft RFQ • Alignment across team on roadmap & prioritized components	• 2 weeks of vendor feedback				

From research and synthesis → product roadmap


Digital by Default

User research is a team sport

GOV.UK

Research Phase:
User Interviews
Systems Mapping
Research Synthesis

In Progress
Product Strategy & Risk evaluation
User Story Development
Prioritization of intended outcomes

Coming up next:
Roadmap finalization
Build vs. Buy Analysis*
Finalized, initial solicitation

Methodology

- User Experience research involves one-on-one interviews to find recurring patterns and themes among the user groups
- Heuristic in UX research is to **conduct 5-8 interviews with a user group to find 80% of the patterns**
- **Conducted 25 user interviews** with staff experts, claimants, employers

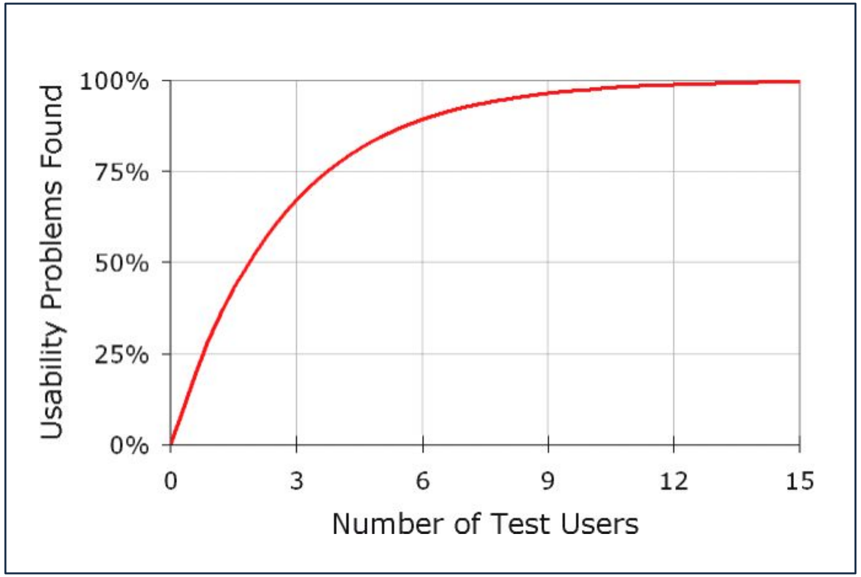


Chart from the Nielsen Norman Group
https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/

Research considerations

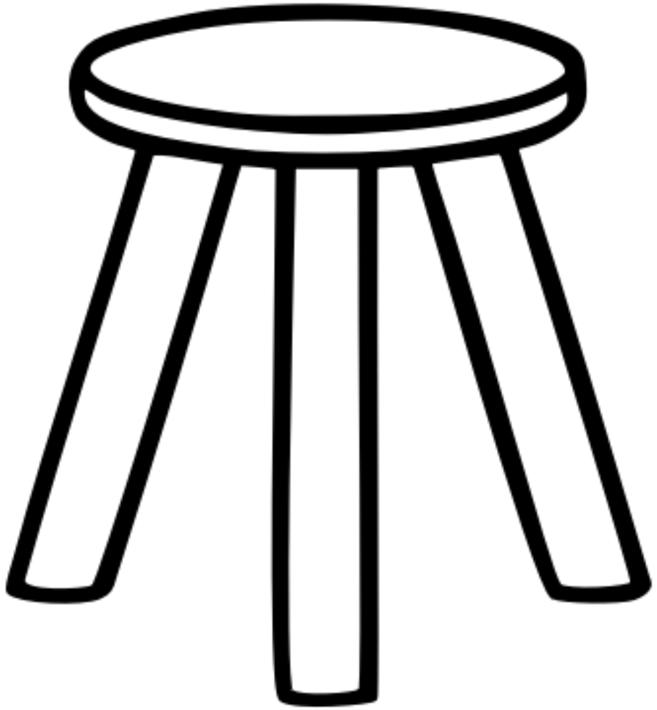
- These findings may not be new or surprising — this work serves to capture your institutional knowledge for future vendors
- Initial round of research — in a user-centered design practice, research is continuous throughout the work
- This builds upon past modernization efforts — more investigation is needed to determine what can be reused from those efforts
- Fewer interviews with claimants and employers than staff

Approach to modernization

1. **Modular pieces of work**
   (balancing customer needs and technical feasibility)
   Answers: What to tackle first? What to turn off first?
   Answers: What do claimants + staff need most?

2. **Encasement of legacy database at the heart of the effort**
   Allows modernization to happen without compromising existing functionality

3. **Throughout, cultural change to adapt to new ways of working**



Modularity

Encasement of legacy database

Cultural change to adapt to new ways of working

How to get started

- Connect a modern API to DB2 to bridge connectivity to the existing data or the future system
- Define and prioritize high-value components
- Design and build a new, streamlined data model to support development of these components
- Document mainframe system, starting with monetaries, to create a map of functions and data interactions to be transitioned
- Use front end work to test encasement while delivering value
- Change management and product leadership (state employees) to support the above

Modernization is hard, but it doesn’t have to be painful

- Gather information to make confident, informed choices
- Deliver early, incrementally, and safely — use feedback to learn quickly how well things work and re-adjust when they don’t
- With early results, build enthusiasm with stakeholders and staff
- As the work solves old problems, find new problems to solve

From what we’ve seen, **DWD is ready** to make this a success

Caution: Modernizing system is <u>not</u> a silver bullet

- Design and build for achieving your program's intended outcomes, not technology

- Don't rely fully on other states' decisions as a blueprint for WI: A recent DOL evaluation of 20 states found few systems actually improved program performance after modernization efforts

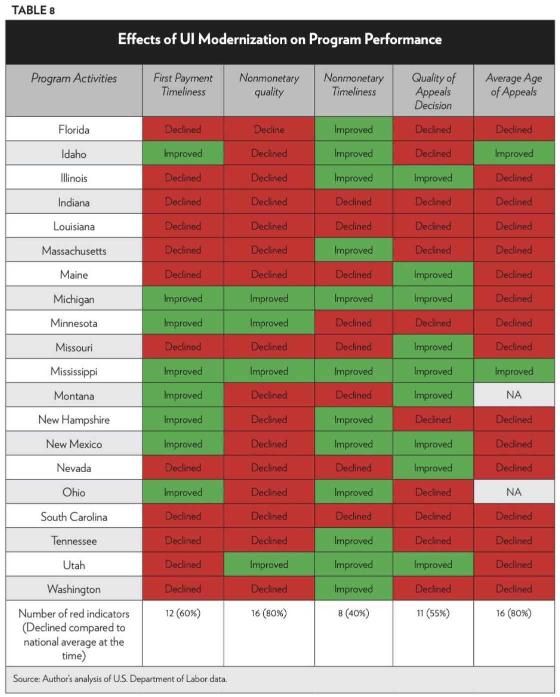


TABLE 8

Effects of UI Modernization on Program Performance

Program Activities	*First Payment Timeliness*	*Nonmonetary quality*	*Nonmonetary Timeliness*	*Quality of Appeals Decision*	*Average Age of Appeals*
Florida	Declined	Decline	Improved	Declined	Declined
Idaho	Improved	Declined	Improved	Declined	Improved
Illinois	Declined	Declined	Improved	Improved	Declined
Indiana	Declined	Declined	Declined	Declined	Declined
Louisiana	Declined	Declined	Declined	Declined	Declined
Massachusetts	Declined	Declined	Improved	Declined	Declined
Maine	Declined	Declined	Declined	Improved	Declined
Michigan	Improved	Improved	Improved	Improved	Declined
Minnesota	Improved	Improved	Declined	Declined	Declined
Missouri	Declined	Declined	Declined	Improved	Declined
Mississippi	Improved	Improved	Improved	Improved	Improved
Montana	Improved	Declined	Declined	Improved	NA
New Hampshire	Improved	Declined	Improved	Declined	Declined
New Mexico	Improved	Declined	Improved	Improved	Declined
Nevada	Declined	Declined	Declined	Improved	Declined
Ohio	Improved	Declined	Improved	Declined	NA
South Carolina	Declined	Declined	Declined	Declined	Declined
Tennessee	Declined	Declined	Improved	Declined	Declined
Utah	Declined	Improved	Improved	Improved	Declined
Washington	Declined	Declined	Improved	Declined	Declined
Number of red indicators (Declined compared to national average at the time)	12 (60%)	16 (80%)	8 (40%)	11 (55%)	16 (80%)

Source: Author's analysis of U.S. Department of Labor data.

https://www.nelp.org/publication/centering-workers-how-to-modernize-unemployment-insurance-technology/

Opportunities & findings

1. Invest time in **documenting the mainframe system**
2. Build a **bridge API, streamline data model for a high priority area of functionality**
3. Make some **strategic, small UX changes to existing interfaces** for quick wins/outsized impact on users
4. Prioritize **staff needs alongside** claimant and legislative priorities
5. Understand and **automate manual workflows in ASP**
6. Invest in **foundational modern practices** in security, DevOps, and product ownership

General operational context

1. Internal problem solvers are **adaptive and responsive to rapidly changing** needs of program. Phone-based contact is widely reported as effective and helpful. Staff is ready to modernize.

2. **Multiple competing, urgent priorities** can de-motivate and overburden staff.

3. Staffing challenges (retirement) and **dramatic increases in volume have made the last 12 months extremely challenging**. Burnout is a concern, and time to train up new staff is a major challenge.

Consider impact of “triage mode” for lengthy periods of time

“[DWD staff] love that we’re moving to new tech, they’re embracing moving to the cloud.”

OPPORTUNITY 1

Invest time in documenting the mainframe system

As an initial encasement step, **document** the mainframe system **starting with monetaries** in order to:

- Sustainably understand data and execution flows, responsibilities, and representations for specific UI program needs
- Uncover dependencies for individual components in the codebase
- Identify migration opportunities

OPPORTUNITY 1

Invest time in documenting the mainframe system

Potential actions

- Engage recent retirees to document the structure of the data, the code, and how they interact.
- Hire content experts to facilitate the drafting of documentation.
- Free up in-house or hire COBOL programmers for documentation sprints.

SUPPORTING FINDING 1

Staff are primary keepers of information, and they have little capacity to help or train others. Keepers of information are retiring, or already retired.

- It is difficult and time-consuming to learn the codebase or program rules and operations.
- Learning one area of the code does not help learning other areas, so knowledge remains dispersed.

SUPPORTING FINDING 1

High-level documentation efforts have been successful in smaller contexts

- In order to understand data flows, TSS has documented components of the mainframe.
- Using a discovery sprint, USDR was able to document the process flow of claims through monetaries.
- Lookup tables for the portals encode knowledge about DB2.

“What we have is called biological documentation.”

“We could have had the retirees we brought back work as SMEs…to provide coding specs.”

OPPORTUNITY 1

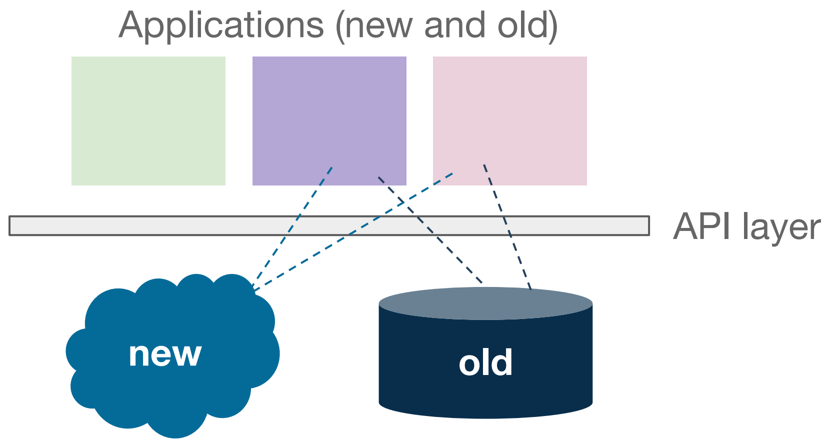
Impact on DWD

+ Improve transparency and situational awareness, allowing more confident modernization decisions
+ Decrease time required to find answers and train staff
+ More rapidly onboard new staff to portfolio of information
+ Increase accuracy of information—single source of truth, single point of update
+ Fundamentally needed to decrease mainframe complexity

OPPORTUNITY 2

Build a bridge layer, streamline data model

As a secondary encasement step, start building an API and the beginning of a new, streamlined data model as a basis to **support a piece of new or existing functionality** while still allowing existing systems to function.

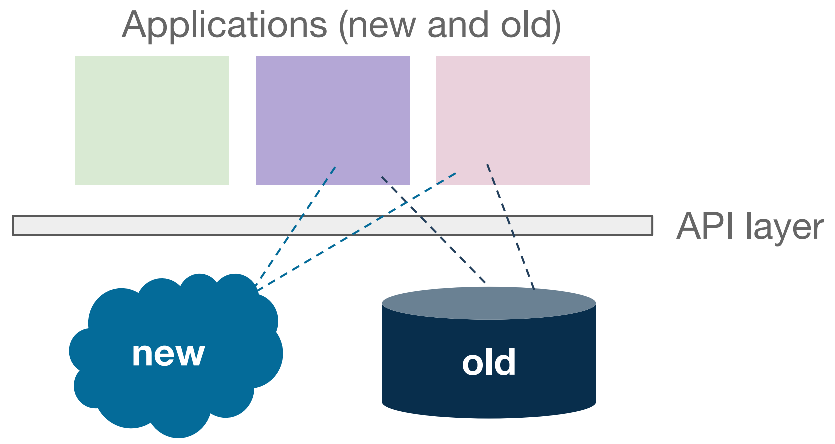


OPPORTUNITY 2

Build a bridge layer, streamline data model

For systems with significant issues with code complexity or data quality, a controlled transition is the key to modernizing without critical outages or data loss.

Encasement provides the framework for that controlled transition.



OPPORTUNITY 2

Build a bridge layer, streamline data model

Potential actions

- Use staff knowledge to draft enough of a data schema* to begin building a bridge API
- Build encasement layer to transform data from the cryptic old format → clean new format
- Use encasement layer to provide the API, using read-only connections to DB2 until the mainframe is further documented
- Match existing data structure to understand execution of reconciling databases

*** this may already exist from internal work on previous modernization projects**

SUPPORTING FINDING 2

Data model is unsustainable and should not be retained for future work. Data must be retained, but its quality is dubious.

- **Data is not in a relational structure** and table structure is not documented.
- **Data structure requires special knowledge to** understand. No single human possesses all this knowledge.
- **Quality and consistency of existing data** is a major concern (data validation: address format, ssn format), and makes reporting and BI very difficult.

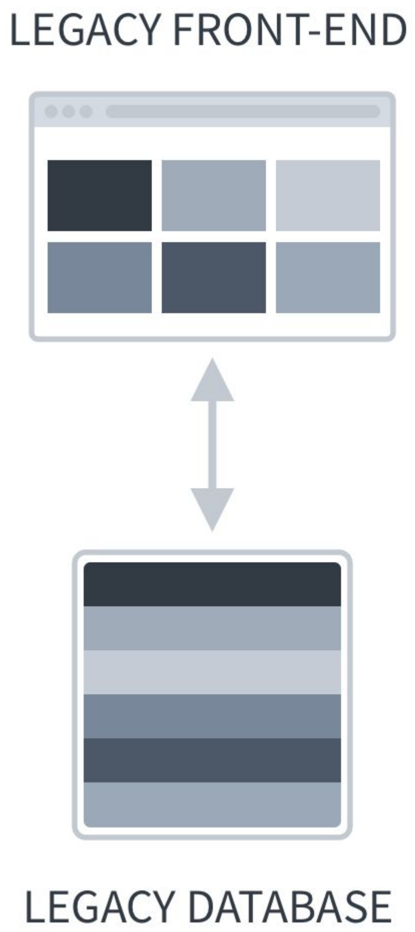
SUPPORTING FINDING 2

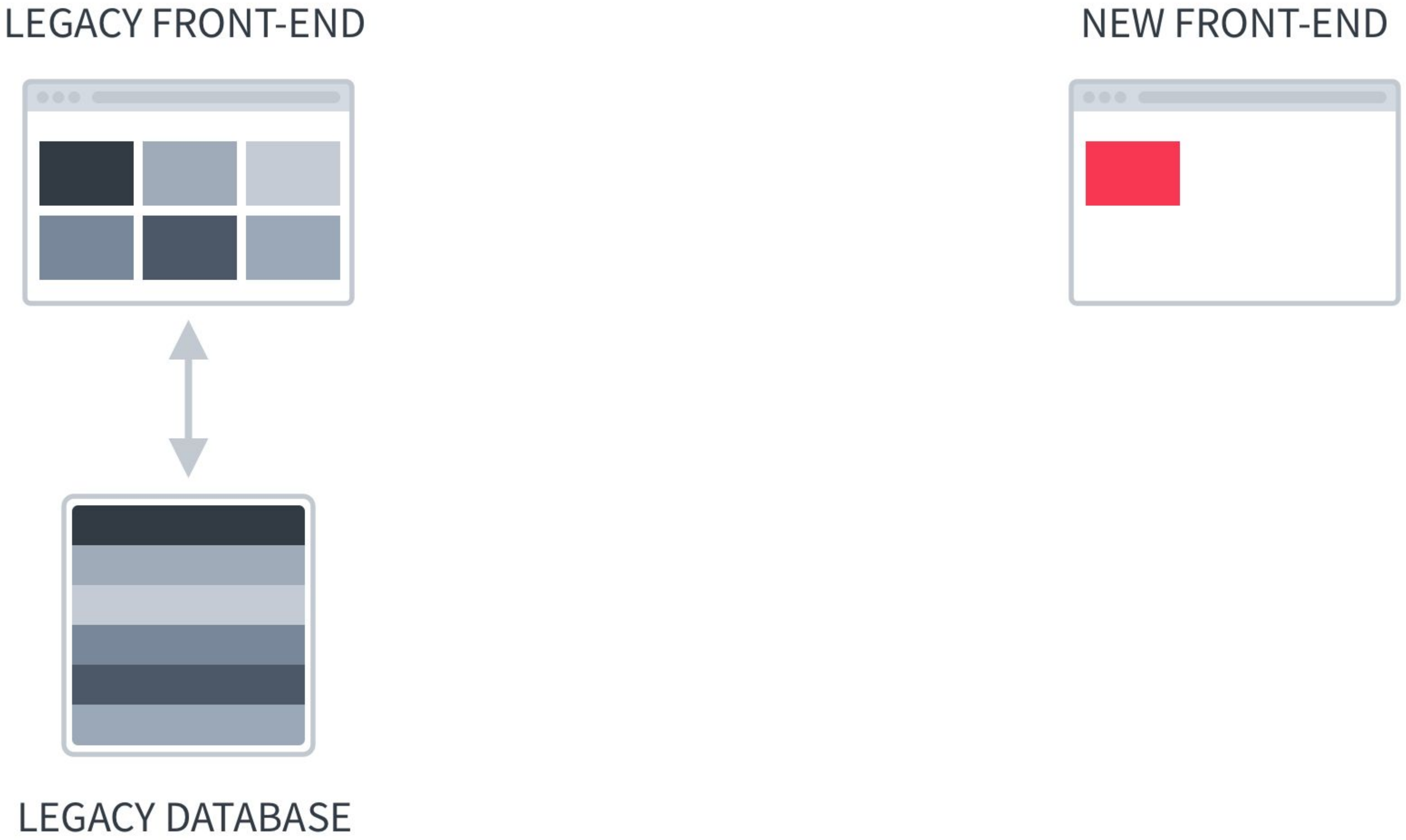
Small, proto-encasement efforts have been the go-to strategy for making use of the legacy data

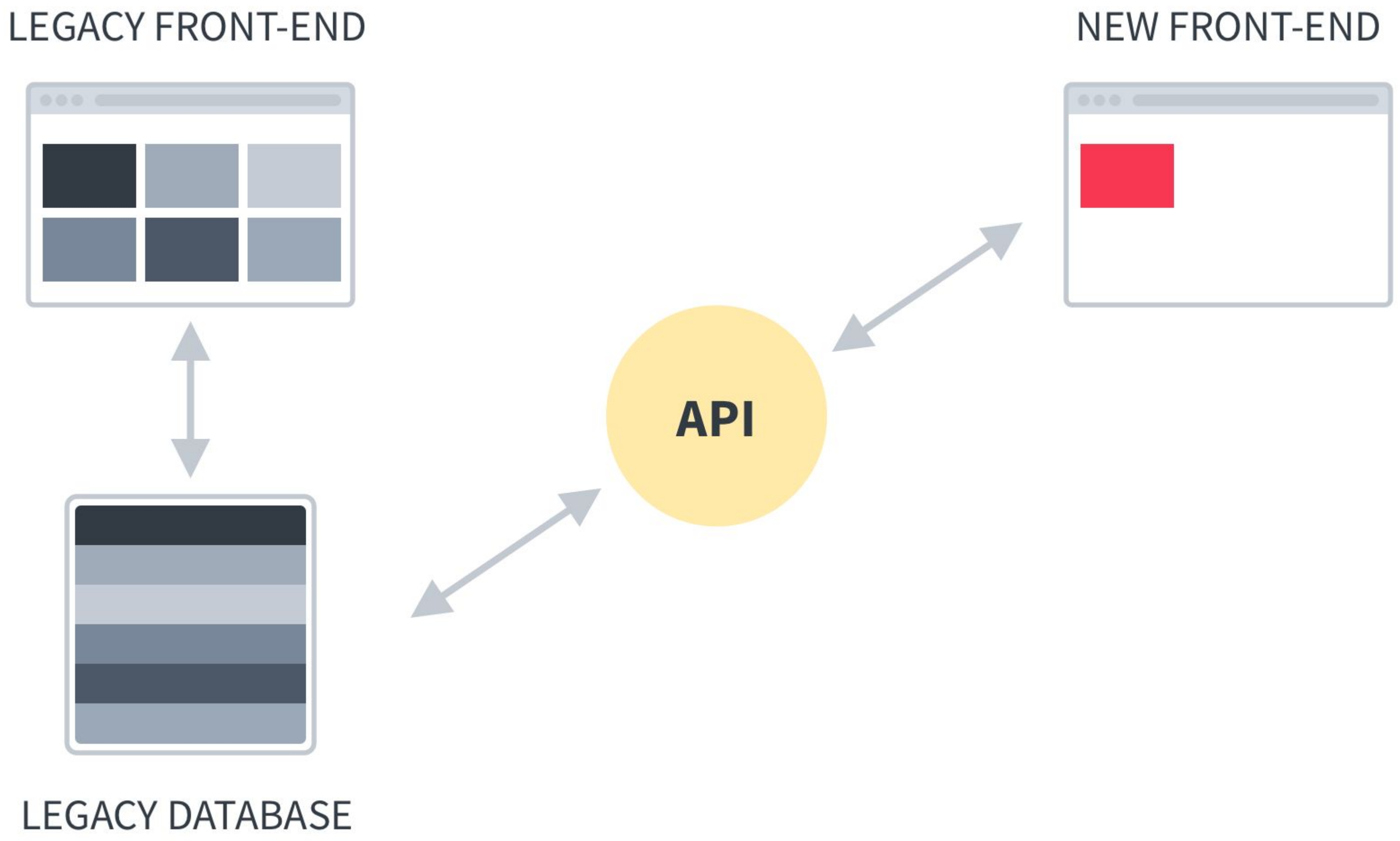
- **The portals are built around extracted data** with lookup tables helping translate cryptic data into more user-friendly language.
- **BI reporting is based on transformed data.** While it is challenging work, TSS has been transforming replicated data in order to do data analysis and prepare reports.

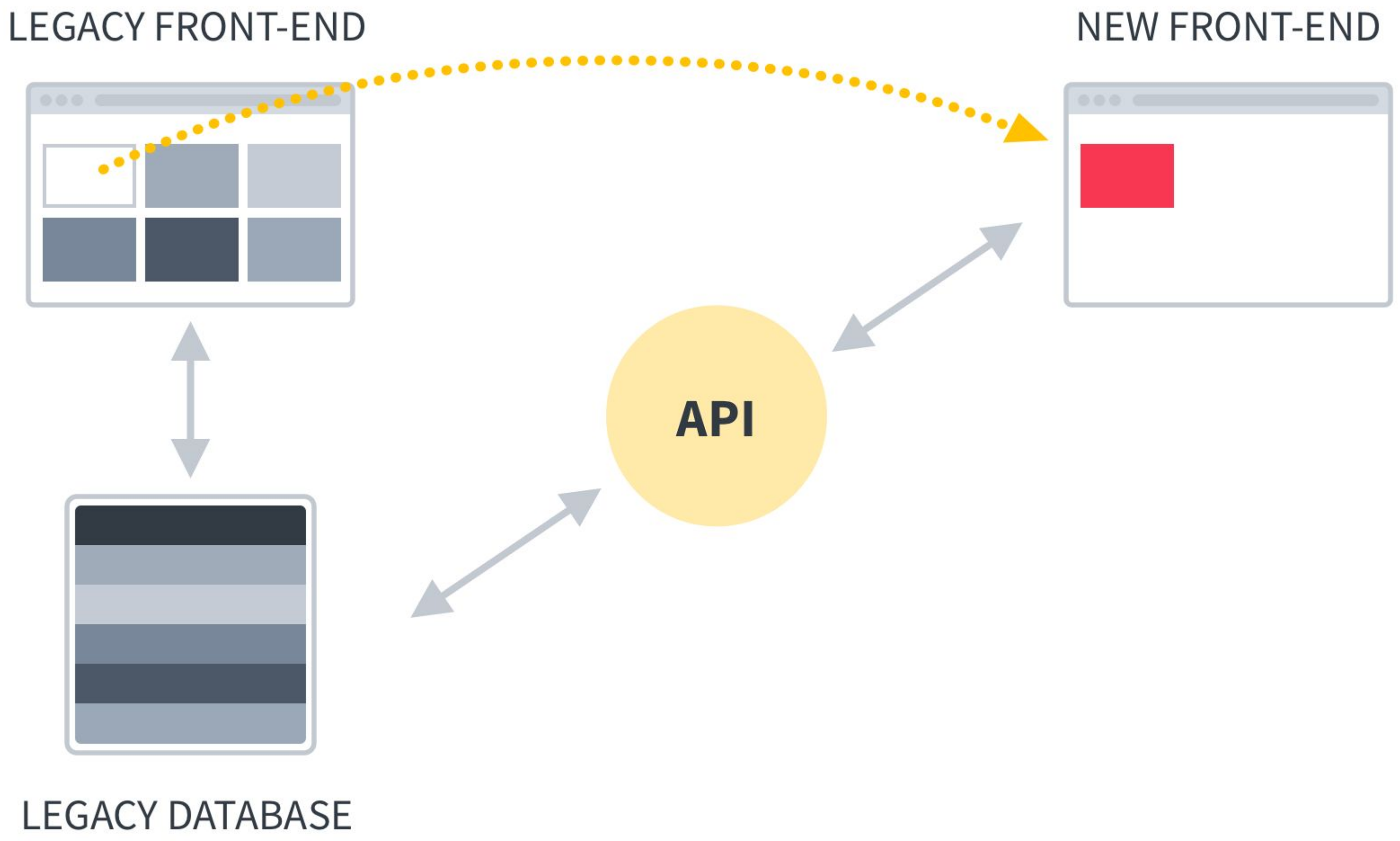
These existing efforts could also take advantage of the deeper encasement push, bringing currently-separate teams closer together.

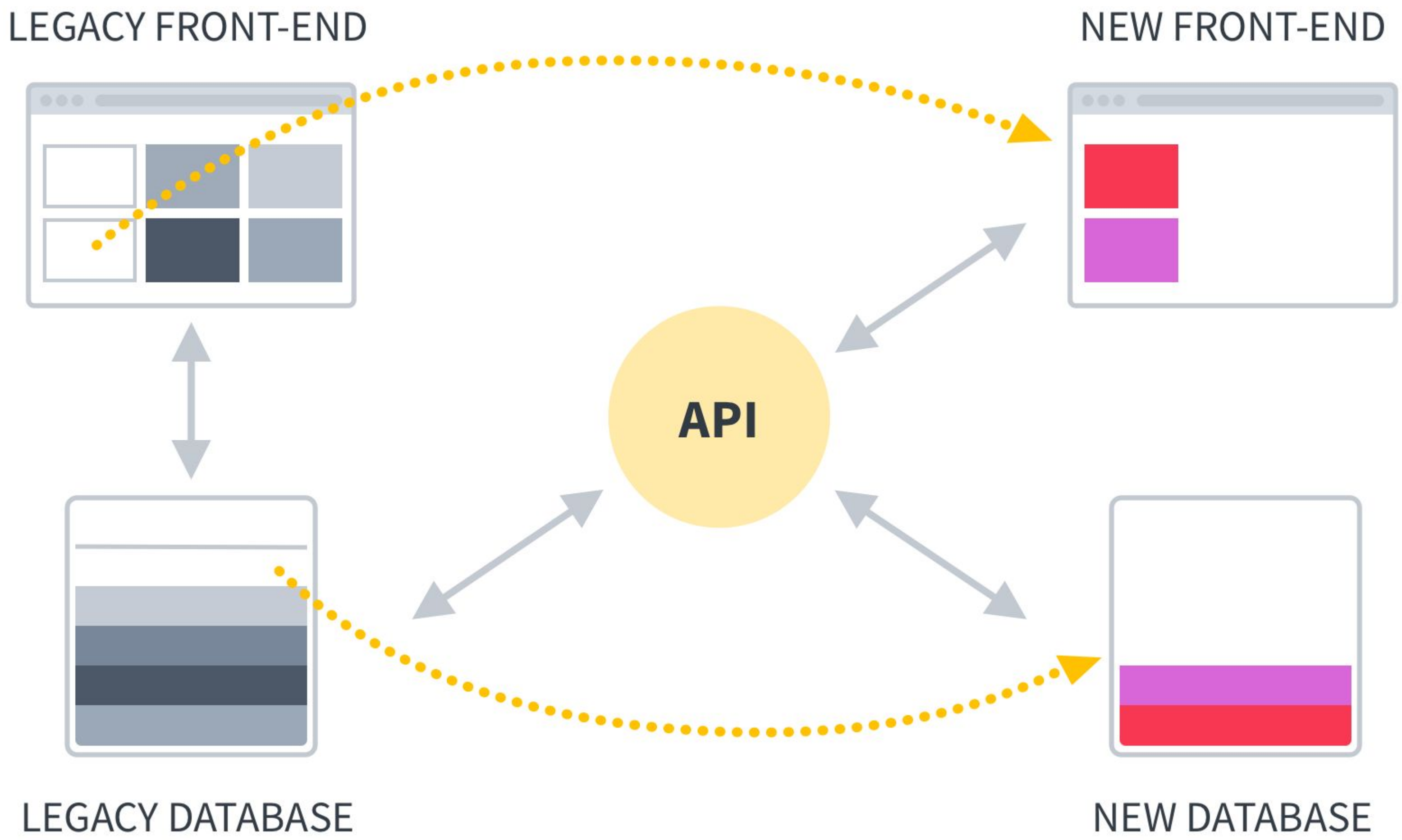
“Trying to count classes, wouldn’t be surprised to find 5,000 database tables. What are the repercussions of making this one change? It takes time to learn.”

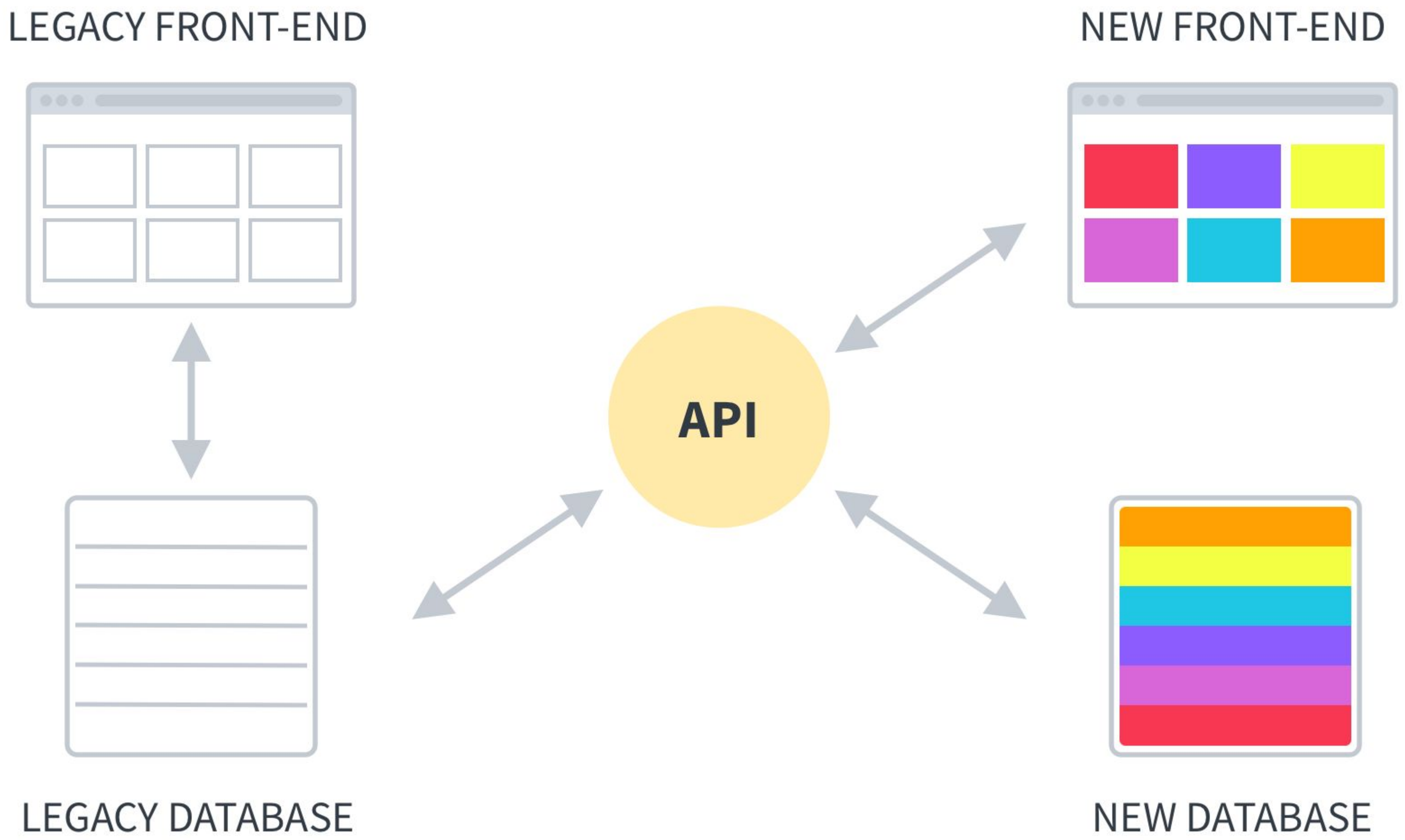

LEGACY FRONT-END
LEGACY DATABASE

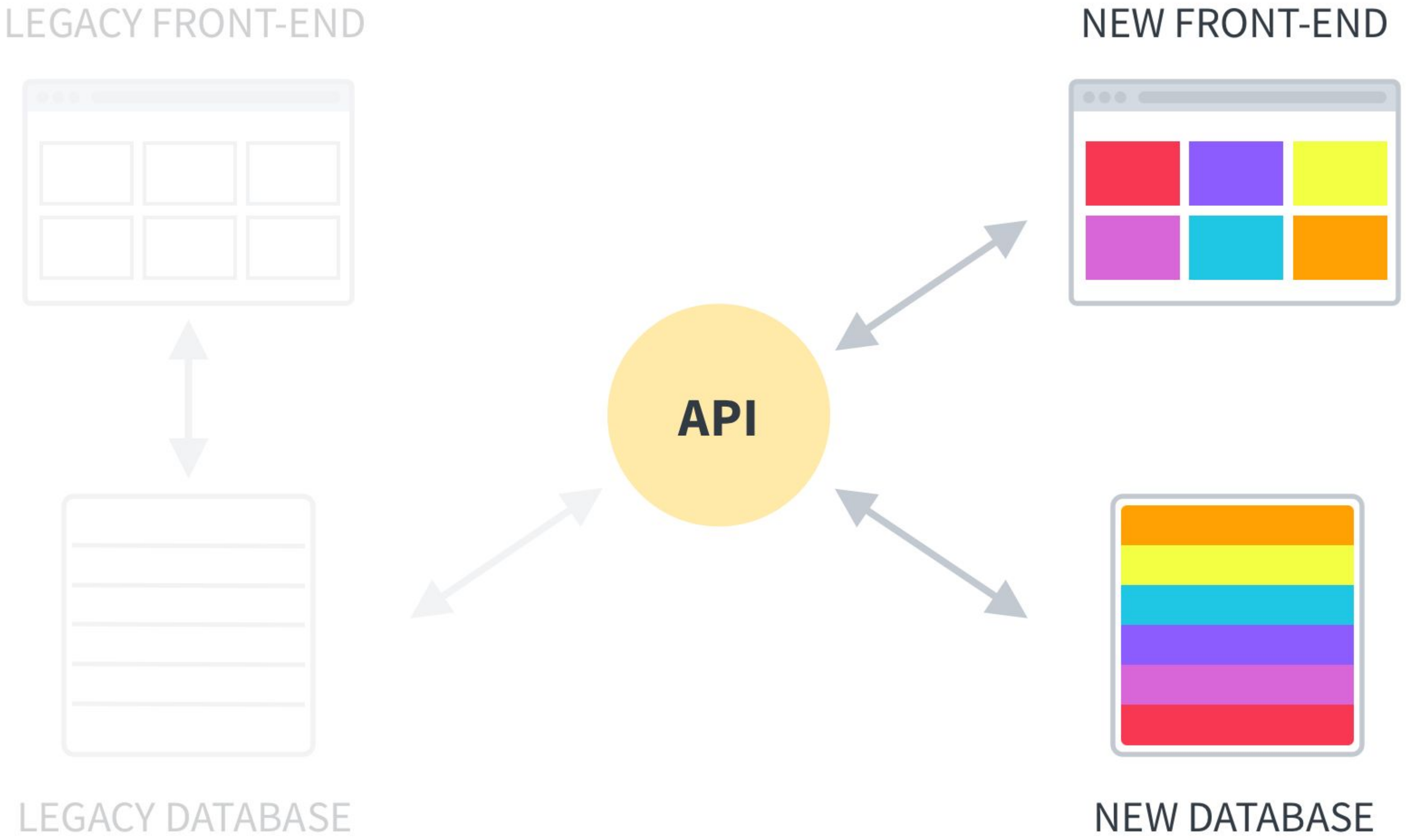

LEGACY FRONT-END
NEW FRONT-END
LEGACY DATABASE

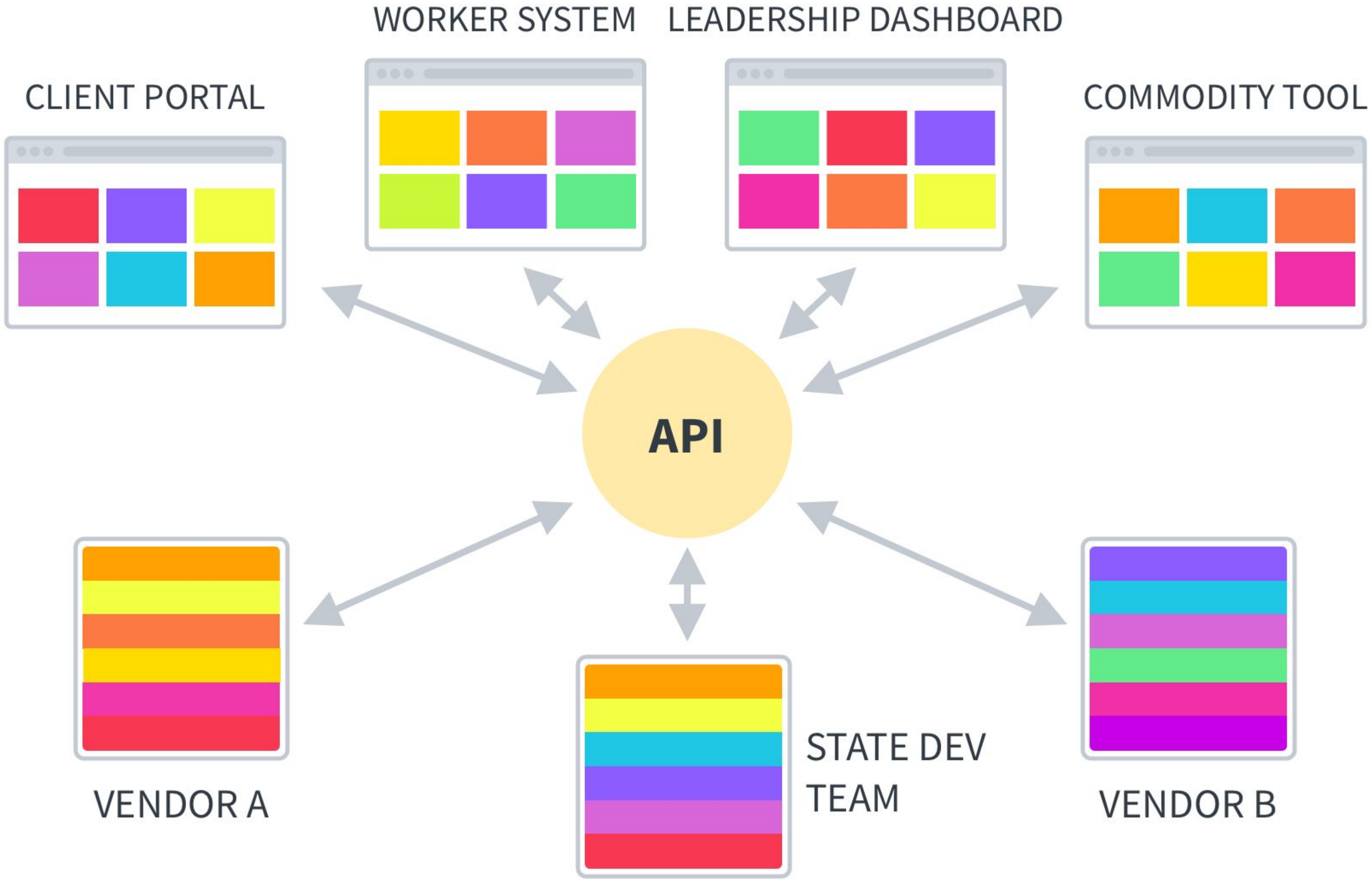

LEGACY FRONT-END
NEW FRONT-END
API
LEGACY DATABASE


LEGACY FRONT-END
NEW FRONT-END
API
LEGACY DATABASE


LEGACY FRONT-END
NEW FRONT-END
API
LEGACY DATABASE
NEW DATABASE


LEGACY FRONT-END
NEW FRONT-END
API
LEGACY DATABASE
NEW DATABASE


LEGACY FRONT-END
NEW FRONT-END
API
LEGACY DATABASE
NEW DATABASE


WORKER SYSTEM
LEADERSHIP DASHBOARD
CLIENT PORTAL
COMMODITY TOOL
API
VENDOR A
STATE DEV TEAM
VENDOR B

OPPORTUNITY 2

Impact of this work

+ Streamlines migration of historical datasets, allowing DWD to **retain data** while making it more usable
+ Creates a vehicle for **controlled, low-risk transition** by keeping mainframe running while building around it
+ Quickly provides data for front-end application development, **allowing that work to begin, get deployed, and be validated earlier** than in a “big bang” approach

OPPORTUNITY 3

Make some strategic, small UX changes to existing interfaces for quick wins/outsized impact on users

Minor content improvements to existing interfaces based on needs identified in user research will go a long way:

- General Notices: Let people know what to expect: synchronize mail & phone notifications with portal
- Claimant portal: announcements and end of application page
- Employer portal: consider allowing upload of .xls

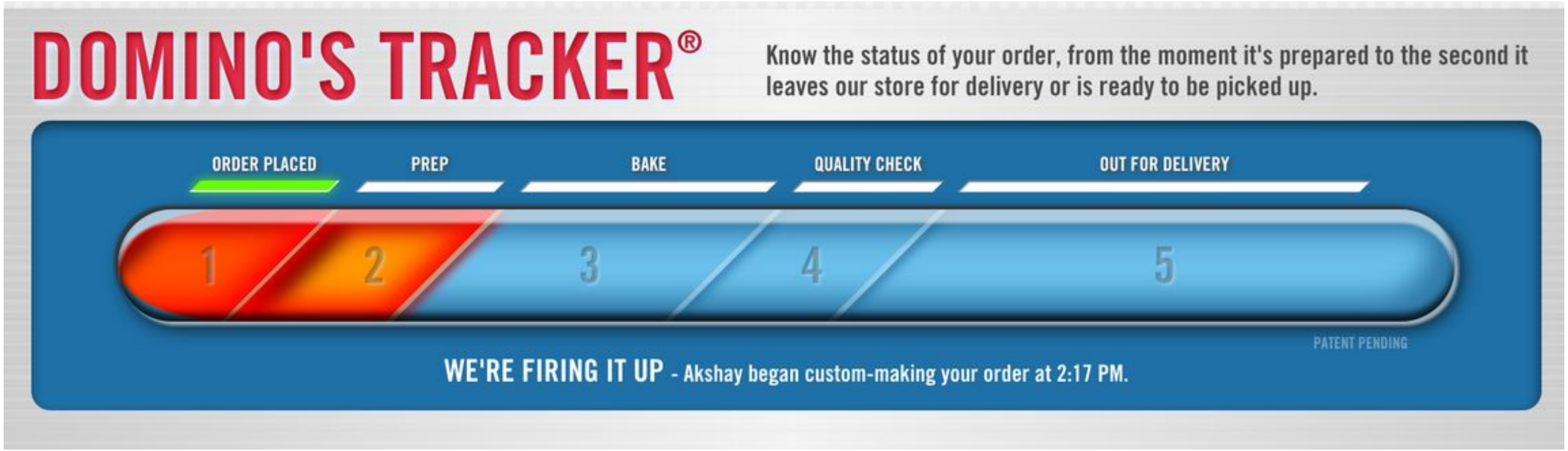
OPPORTUNITY 3

Make some strategic, small UX changes to existing interfaces for quick wins/outsized impact on users

Potential actions

- Phone-based interactions (they're working!), consider scale during high volume periods
- Support in-flight work to improve public-facing information
- **Conduct regular usability testing with real users**
- Test the bridge API by providing targeted real-time data that users frequently need
  - Incorporate this data into existing portals
  - Hire a vendor to build a fresh interface for staff and claimants

Improving customer experience by describing what will happen next


DOMINO'S TRACKER®

Know the status of your order, from the moment it's prepared to the second it leaves our store for delivery or is ready to be picked up.

WE'RE FIRING IT UP - Akshay began custom-making your order at 2:17 PM.

SUPPORTING FINDING 3

Claimant and employer-facing portals work fairly well. Users are noticing improvements DWD is making. Call center interactions are helpful and “marvelous.”

- Surprise phone calls trigger concern: claimants and employers were **surprised by UNKNOWN callers** and perceived interaction as scammers

- Framing, lack of visual hierarchy and some communication of key information [announcements page, weekly reporting rule] is **overwhelming users. In some cases, triggered rescinding an application voluntarily.**

- Call center interactions, once connected, was reported as helpful, kind and “marvelous.”

“Should they be calling me, is this real? I didn't know.”

“I was hesitant to talk to anyone. I didn't get any notices in the mail and I don't know who you are.”

OPPORTUNITY 3

Impact of this work

+ Delivers improved experience to DWD customers
+ Builds user research and user-centered design practices, which helps define later priorities
+ Ties in to bridge layer work in a cohesive way
    - Tests end-to-end delivery to production with low risk, giving early feedback into the viability of this modernization approach
    - Provides an opportunity to build CI/CD pipelines, building modern tooling and best practices for later work to leverage

OPPORTUNITY 4

Prioritize staff needs alongside claimant and legislative priorities

Balance prioritizing efforts (alongside state mandates and claimant needs) to include staff experience. By reducing staff burden, you will avoid burnout and create opportunities for efficiency. Limit number of programs/interfaces staff must engage with to do their work.

(Smarter, not harder)

OPPORTUNITY 4

Prioritize staff needs alongside claimant, employer, and legislative priorities

Potential actions

- Seek out projects that automate manual, slow processes.
- Balance staff needs along with claimant needs in calculating priorities. Alleviating staff burden will alleviate claimant burden.
- Existing operational strategies and priorities have been postponed (agile practices, known fixes prior to pandemic). Make space to return to these goals.

SUPPORTING FINDINGS 4

Staff is at capacity, some pre-pandemic operational goals have been deprioritized.

- Impact of competing priorities: de-prioritization of existing operational goals, agile practices turned into “just do it”.
- Many interfaces that staff need to use in order to do work (we counted 11+), all built on top of the same data.

“We’re used to handling 50–100 claims a week, but are handling 400 without additional people.”

OPPORTUNITY 4

Impact of this work

+ DWD staff have skills and are actively working on the current platforms — supporting these people and maintaining these skills positively impacts sustainability of modernization efforts
+ Will give staff more time to deliver higher-value work, including supporting of modernization efforts or business intelligence/reporting

OPPORTUNITY 5

Understand and automate manual workflows in ASP

Use data and analytics to evaluate the things taking most time and impacting claimants most. Identify highest priority manual processes to automate (or not) depending on lift.

SUPPORTING FINDINGS 5

ASP Staff have a number of manual workarounds because of the gaps in system function and the difficulty of making updates. This is the biggest cause of delays to claimants.

For high-priority functionality, third parties are processing data outside of the system (SpringML) and porting it back in, manually.

“We [ASP] are the end of the rope, we are the ones who handle everything that the system can’t.”

OPPORTUNITY 5

Impact of this work

+ Reduce the delay of benefit payments for claimants
+ Shift the focus of ASP staff to higher value work
+ Alleviate cost of training ramp up time and work management

OPPORTUNITY 6

Invest in foundational modern practices in security, Devops, and product ownership

People are ready for change, including cultural and
organizational change. This is an opportunity to introduce
More of these foundational practices to your organization, long-term.

OPPORTUNITY 6

Invest in foundational modern practices in security, Devops, and product ownership

Potential actions

- Embed user-centered design into requirements development. Re-invest in agile methods and empower product owners.
- Establish a cross-division product team to lead modernization work.
- Identify a state employee lead to drive vendor efforts and prioritize customer centric design
- Involve security earlier, and invest in security resourcing and process documentation and automated tools.
- Build a CI/CD pipeline as part of any new application development, using automated testing to decrease production risks.

SUPPORTING FINDINGS 6

While Agile was introduced in 2008, modern product development practices have been set aside to deal with the huge volume of work

- Requirements are described in terms of output and bug fixes. Desire to move to business-driven, outcomes-focused feature development.
- Security not involved at the start, focusing on user administration and compliance reporting rather than application security.
- Steps that can be automated, such as testing and deployment, are still manual, creating additional work and increasing risks of accidents.

“People checking out code in [the development environment] is causing code merge issues and means we are overwriting each other.”

OPPORTUNITY 6

Impact of this work

+ Customer experience at center of all development efforts, delivering value more quickly to end users
+ Continuous deployment and integration of code
+ Automated testing will relieve manual processes conducted by staff, often new to testing software
+ Allows programmers to work on more value-added work vs. disconnected patches
+ Improves time to delivery by mapping end-to-end path to production
+ Better environment for hiring a modern vendor team

In the next 2 weeks…

Roadmap

Prioritize of intended outcomes

Identifying DWD roles (resource planning, change management strategy, product ownership)

Cost estimation

In the next 2 weeks…

Roadmap

Prioritize of intended outcomes

Identifying DWD roles (resource planning, change management strategy, product ownership)

Cost estimation

Close information gaps

Prototype based on our hypotheses

Service blueprint

More research into other states

System diagramming

In the next 2 weeks…

Roadmap

Prioritize of intended outcomes

Identifying DWD roles (resource planning, change management strategy, product ownership)

Cost estimation

Close information gaps

Prototype based on our hypotheses

Service blueprint

More research into other states

System diagramming

Acquisition

Identify which pieces to build and/or buy

Market research

RFP de-risking workshop

Finishing RFPs

Upcoming milestones

5/10-5/14	May 18 & 19	5/24 – 5/28	5/31 – 6/11	6/15 – 6/30	7/1 – 7/31	7/30 – 8/31
Prioritize UI outcomes, Build Roadmap, draft solicitation	**De-risking Acquisition Workshop**	**Review RFQ & Release draft for comment by 5/28, open for 2 weeks**	**Final review and adjustments to RFP**	**Publish final RFP**	**Receive questions & proposals**	**Interviews & selection**
• Drafted RFQ • Drafted roadmap • Build vs. buy discussion	• Refine Draft RFQ • Alignment across team on roadmap & prioritized components	• 2 weeks of vendor feedback				

Thank you!

Appendix

DevSecOps

Adapted from slides by Greg Walker, 18F

DevSecOps

noun

An environment where the tech team on a software project **works together** to facilitate the **continuous delivery** of safe and valuable software.

development (writing code)

security (making sure code is safe)

operations (code working in production)

Culture
Practices
Tools

Not an uncommon pattern for government...

Sysadmin

Security

Developer

Business

An ideal we can work towards, one day/meeting/email/call at a time…

DevSecOps culture — what and why

Open communications and **knowledge sharing** enhance trust within the team

Continuous feedback results in tighter processes that get better and better

Emphasizes the entire **system**, not just one piece, department, or interest

Experimentation lets us learn from successes and failures, constantly

DevSecOps culture — some ideas for how

Collaborative workspaces and chat-based communication tools

Regular meetings and retrospectives with all DevSecOps teammates

Include all DevSecOps stakeholders in meetings, tools, check-ins and communications

Experiment with one DevSecOps practice at a time, prioritized and chosen by the team

Culture

Practices

Tools

4 DevSecOps practices to ask your technology vendors to use

1/ Testing

Does it do what it’s supposed to?
Do the things that worked yesterday still work today?

Testing is about making sure the application does what it’s supposed to and catching bugs as early as possible.

Testing applications

1 **Assert** what the features of the application should do:

```
assert screen-output =
"Hello!"
```

Testing applications

1. **Assert** what the features of the application should do:
```
assert screen-output =
“Hello!”
```

2. **Write** some code to implement those features:
```
puts “Hello!”
```

Testing applications

1 **Assert** what the features of the application should do:

```
assert screen-output =
“Hello!”
```

2 **Write** some code to implement those features:

```
puts “Hello!”
```

3 **Execute** the tests to make sure the application does what we expect:

```
Finished in 0.3s, 1 assertion
0 failures, 0 errors
```

Why is testing good?

Writing tests often helps you **understand** the problem you feature is trying to solve

Good tests make sure fixed bugs stay fixed — **regression testing**

Tests can help future developers understand what a feature does — they're like **documentation**

Some bad development habits make testing difficult — tests encourage **good habits**

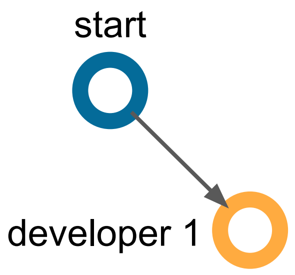
2/ Continuous Integration

Making sure the different parts go together, each step of the way

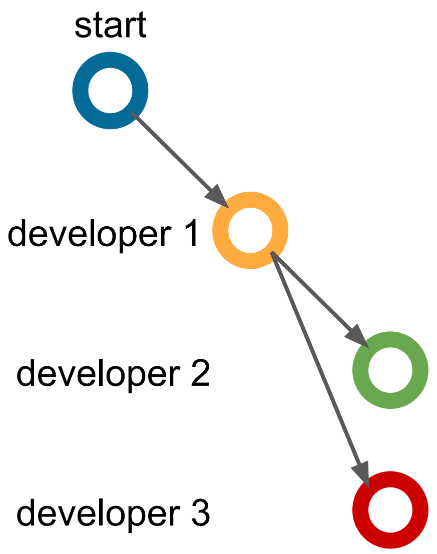
Continuous integration is taking each change, as it happens, and checking whether it breaks the product. This way the team knows quickly, as soon as a change is made, rather than days or weeks later.

From many, one

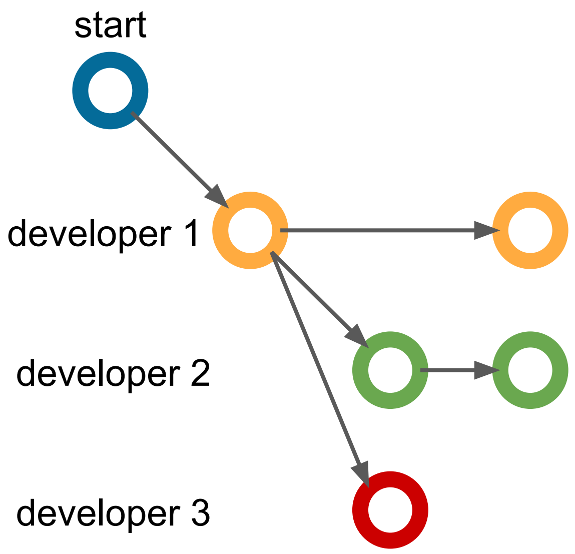
From many, one



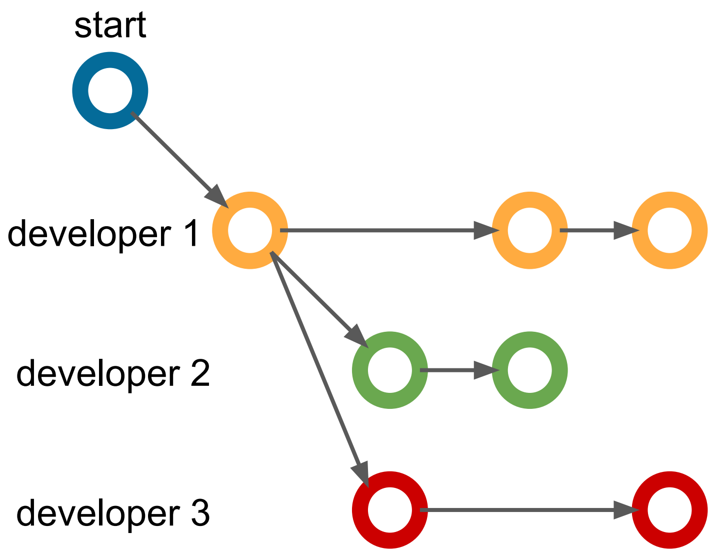
From many, one



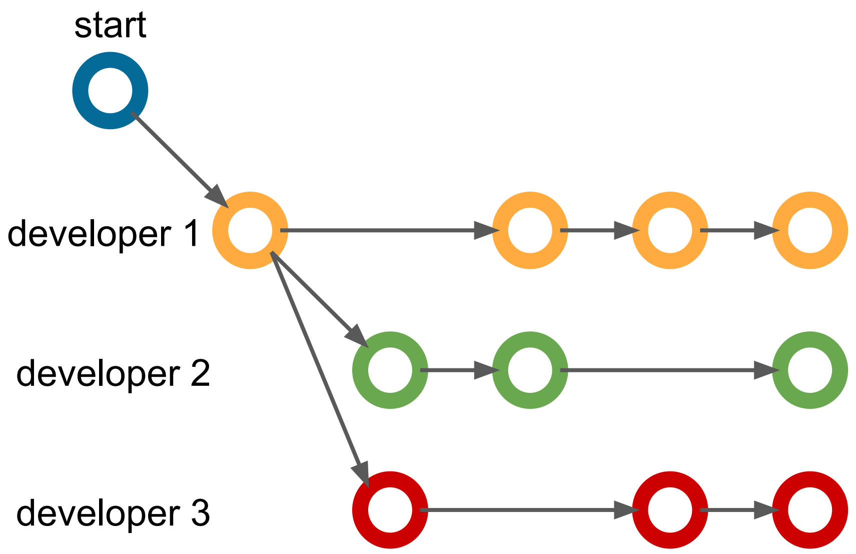
From many, one



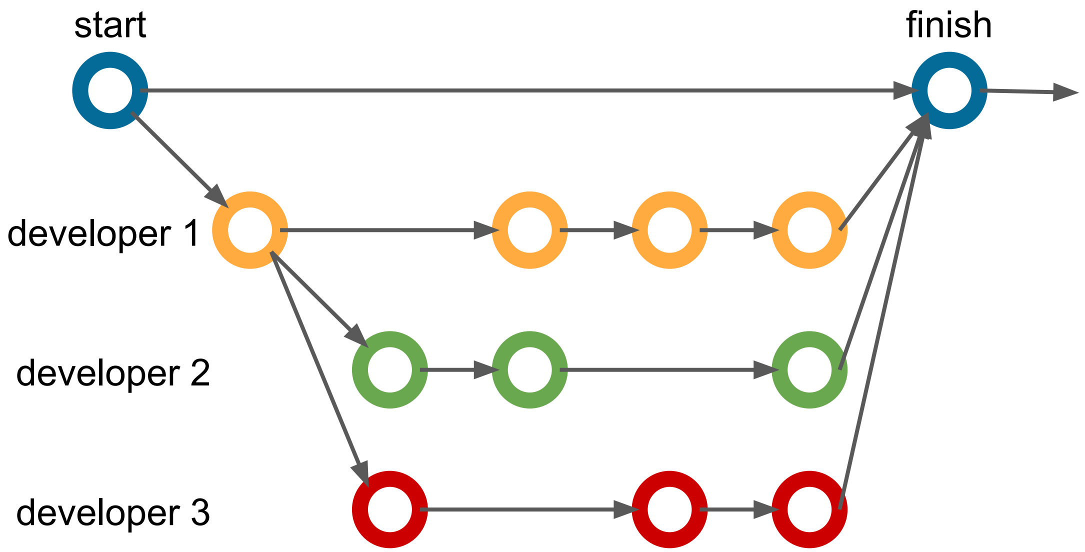
From many, one



From many, one

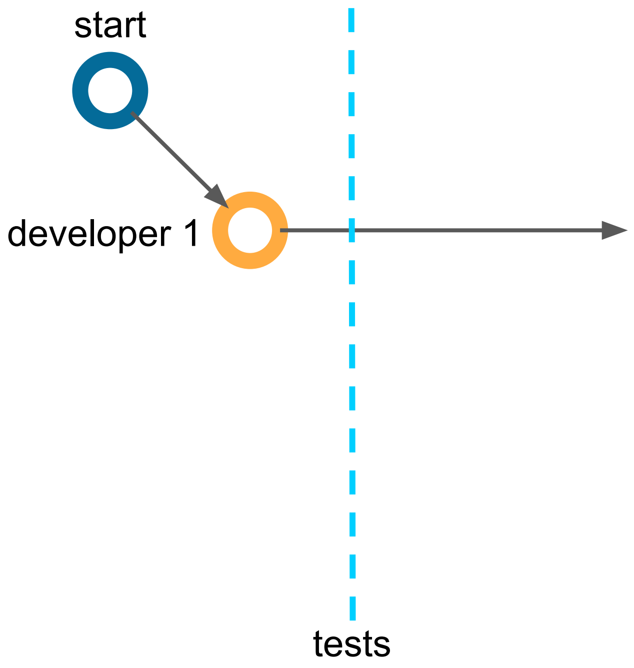


From many, one

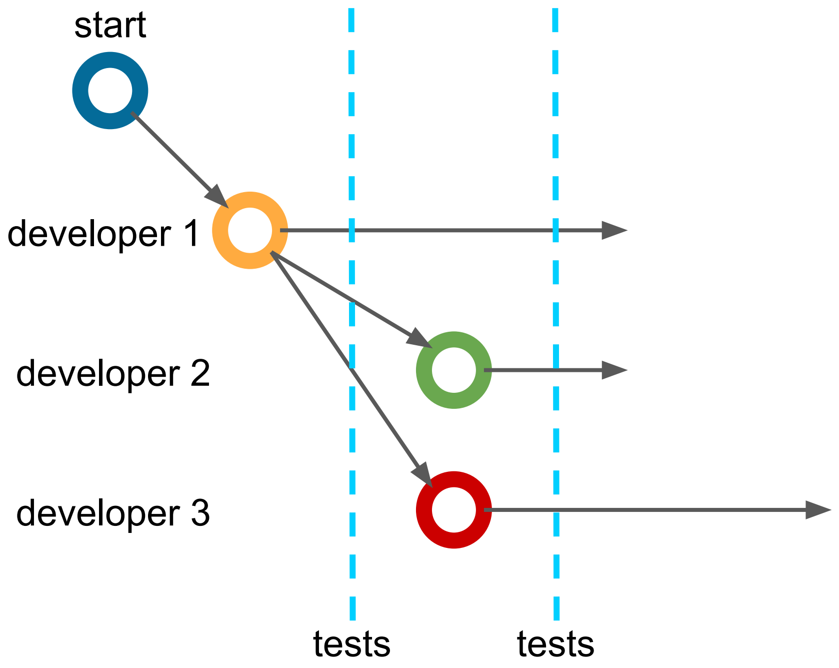


That looks like a disaster about to happen...

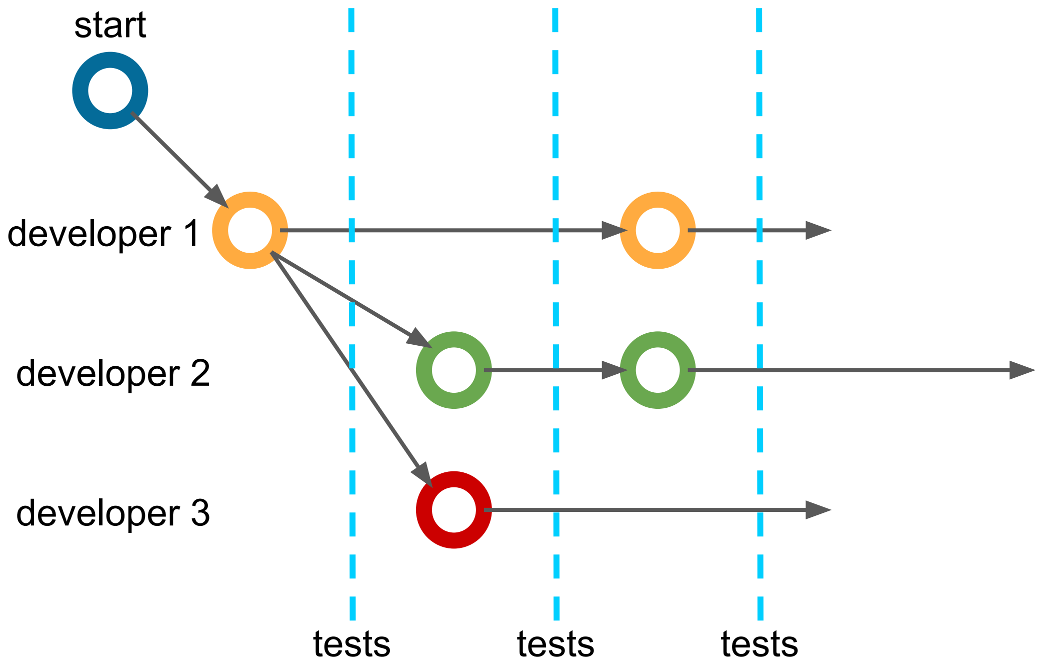
Everything is okay...



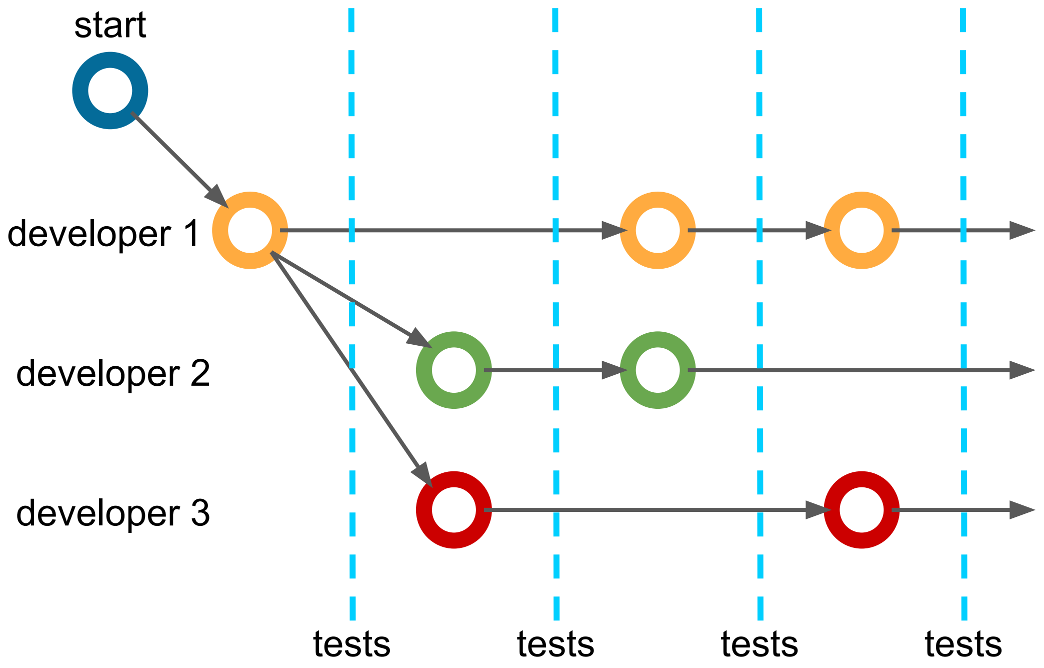
...and it’s still okay...



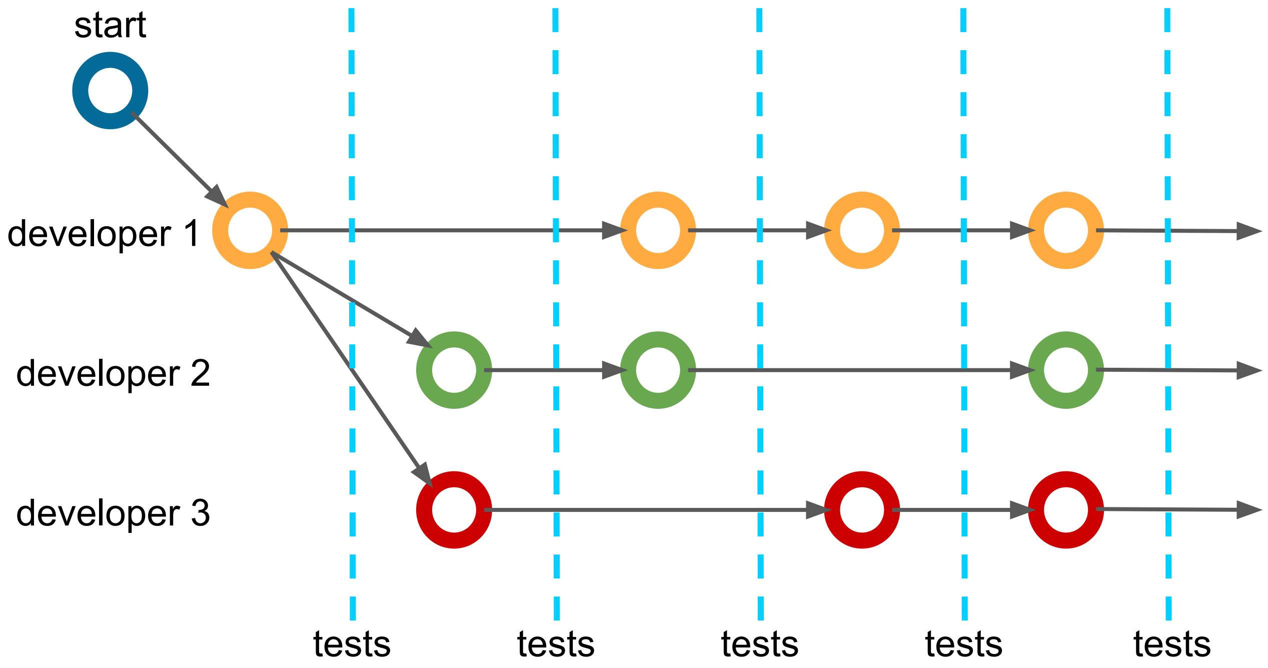
...and it's still okay...



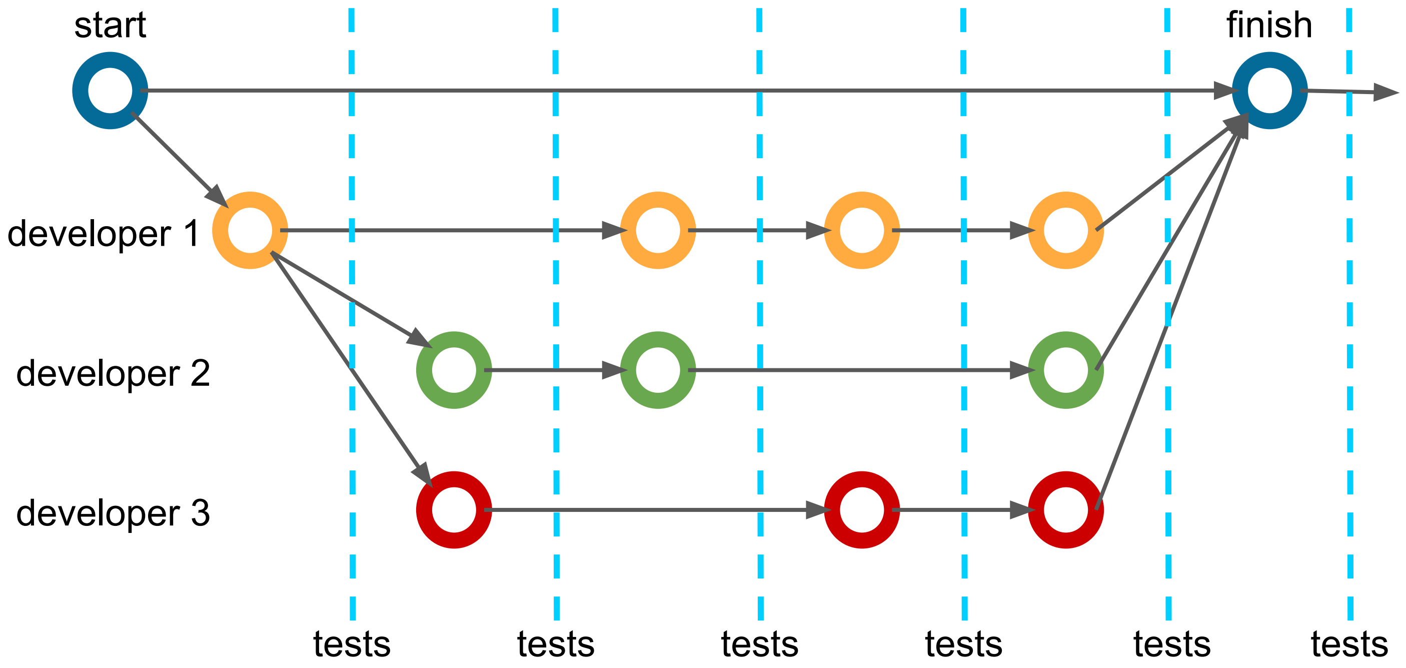
...and it's still okay...



...and it's still okay...



...and everything is good!



Why is continuous integration good?

As changes are made, we can have assurances that everything still works

Developers can work on multiple features simultaneously

Tests aren't the only thing we can CI — we can also checks for security, compliance, accessibility, code style, documentation...

CI results are usually open to the whole team — everyone can see what's going on

3/ Continuous Delivery

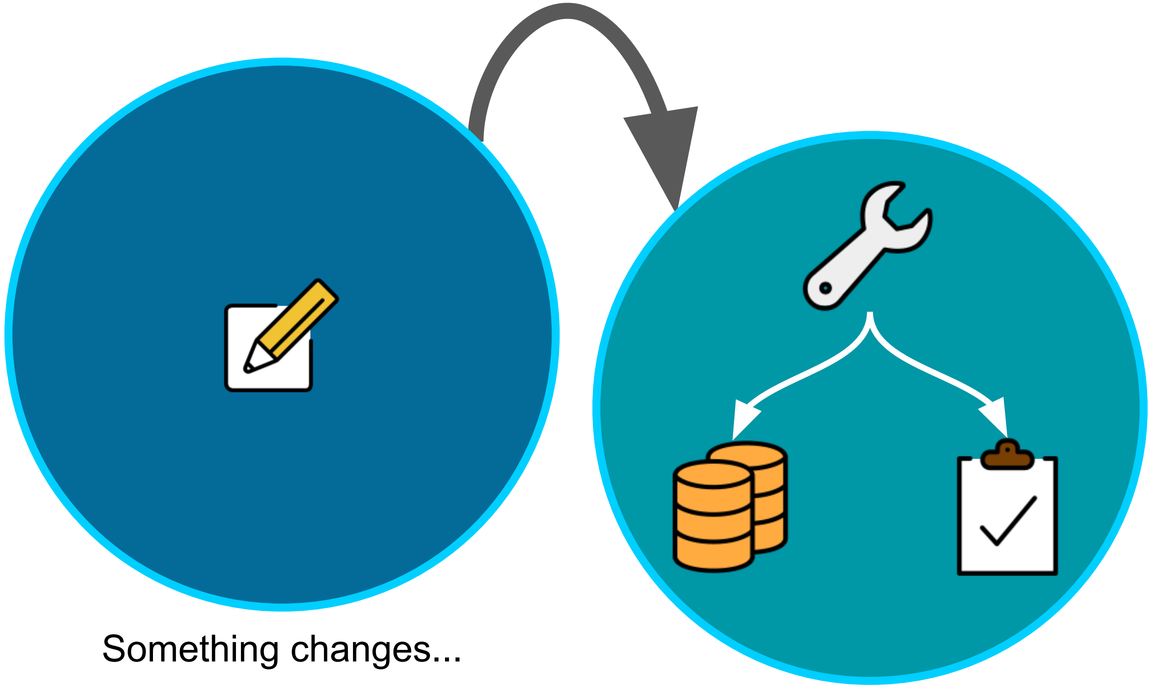
Making the new, updated app available as often as possible.

Every time some changes gets pulled into the app, the app gets deployed.

The delivery

Something changes...

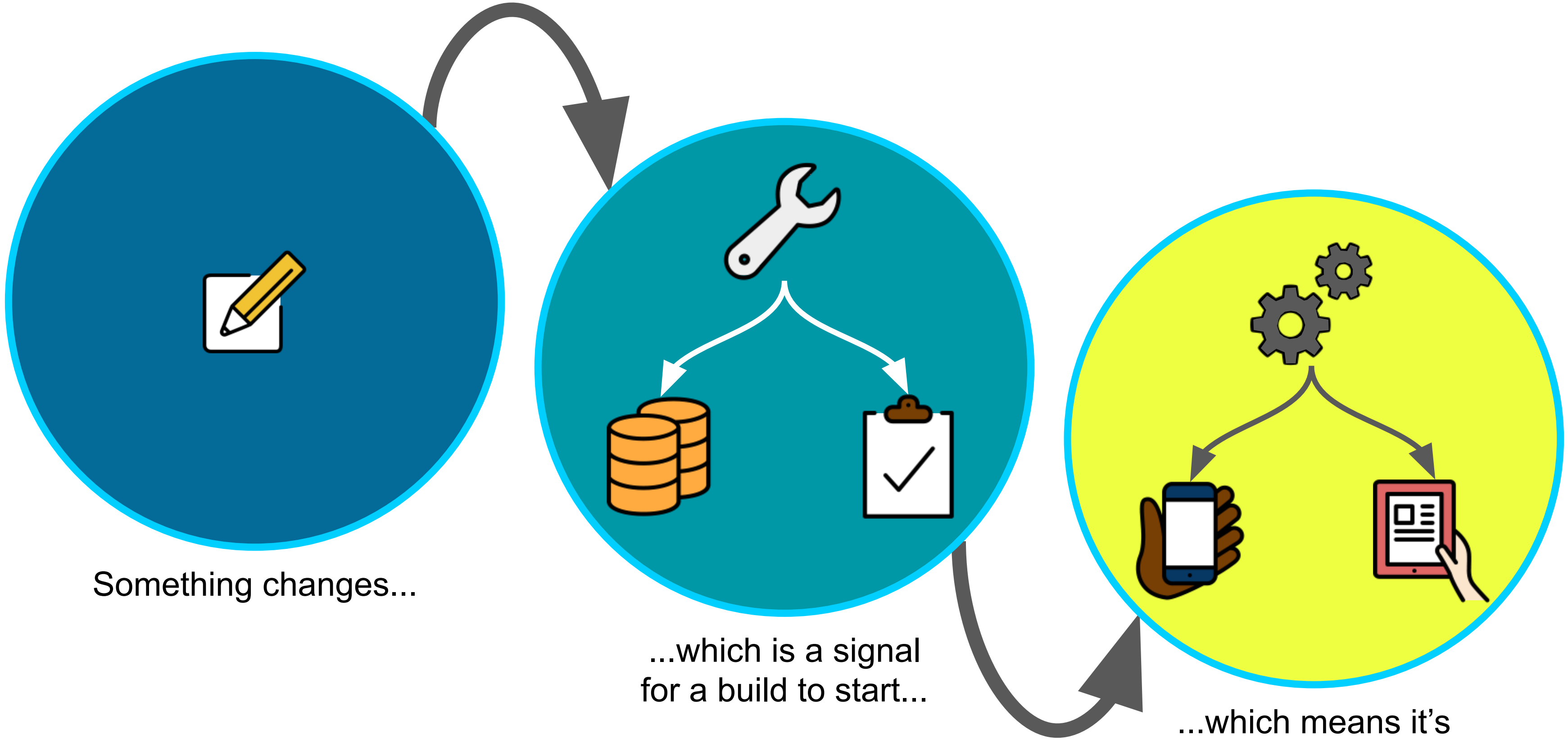
The delivery



Something changes...

...which is a signal
for a build to start...

The delivery


Something changes...
...which is a signal for a build to start...
...which means it's time for deployment

Why is continuous delivery good?

Changes can be quickly made available to users for their testing and feedback

Problems that aren’t covered by tests can be identified in days instead of weeks

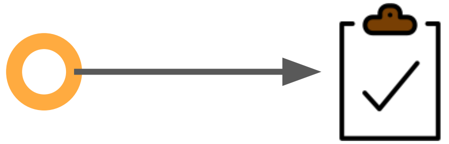
Deliveries become routine and unexciting, which is a nice change of pace for ops

Continuous delivery enables multiple “kinds” of deploys — test, staging, production, etc.

4/ Automate!

An automated CI/CD process

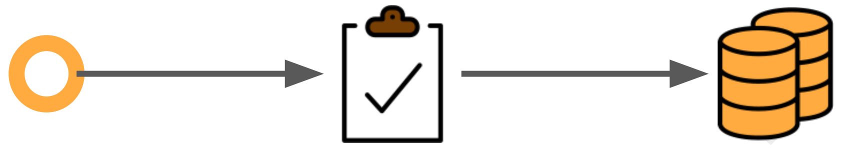
An automated CI/CD process



a developer makes a change

CI process detects change and runs tests

An automated CI/CD process

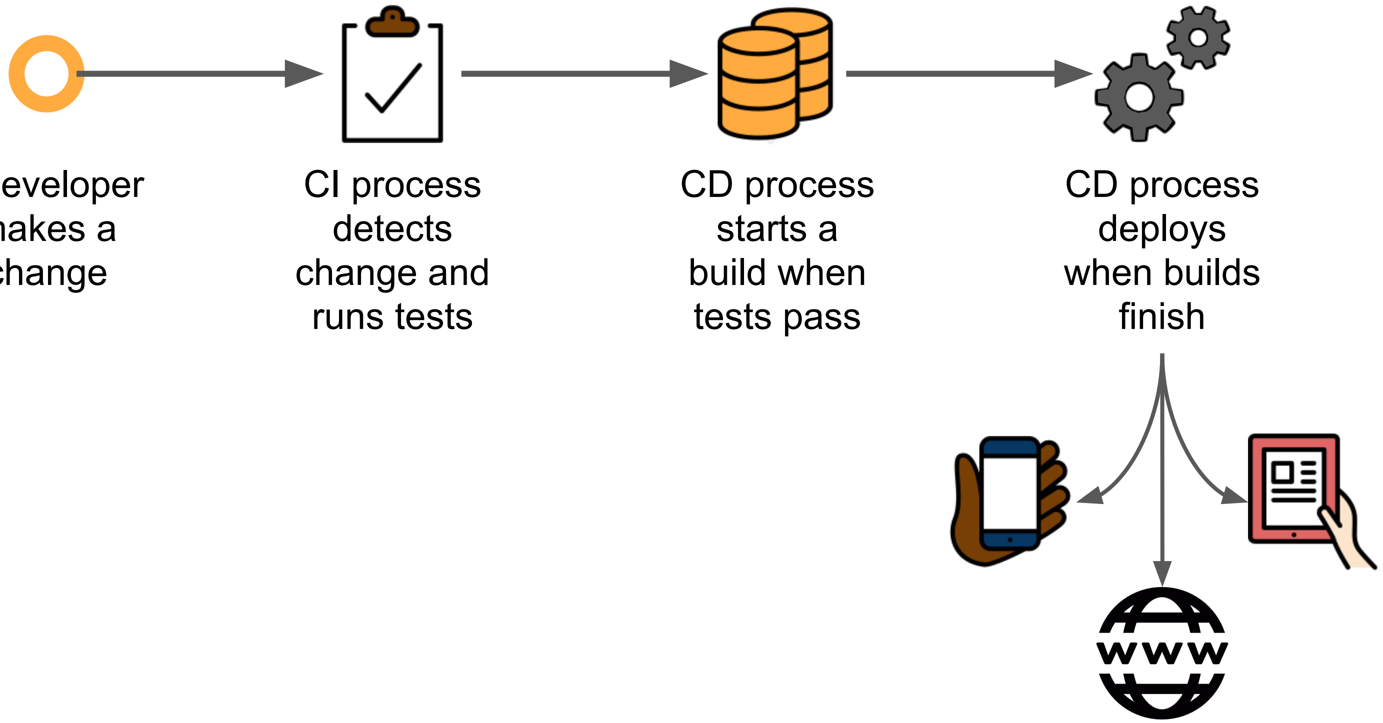


a developer makes a change

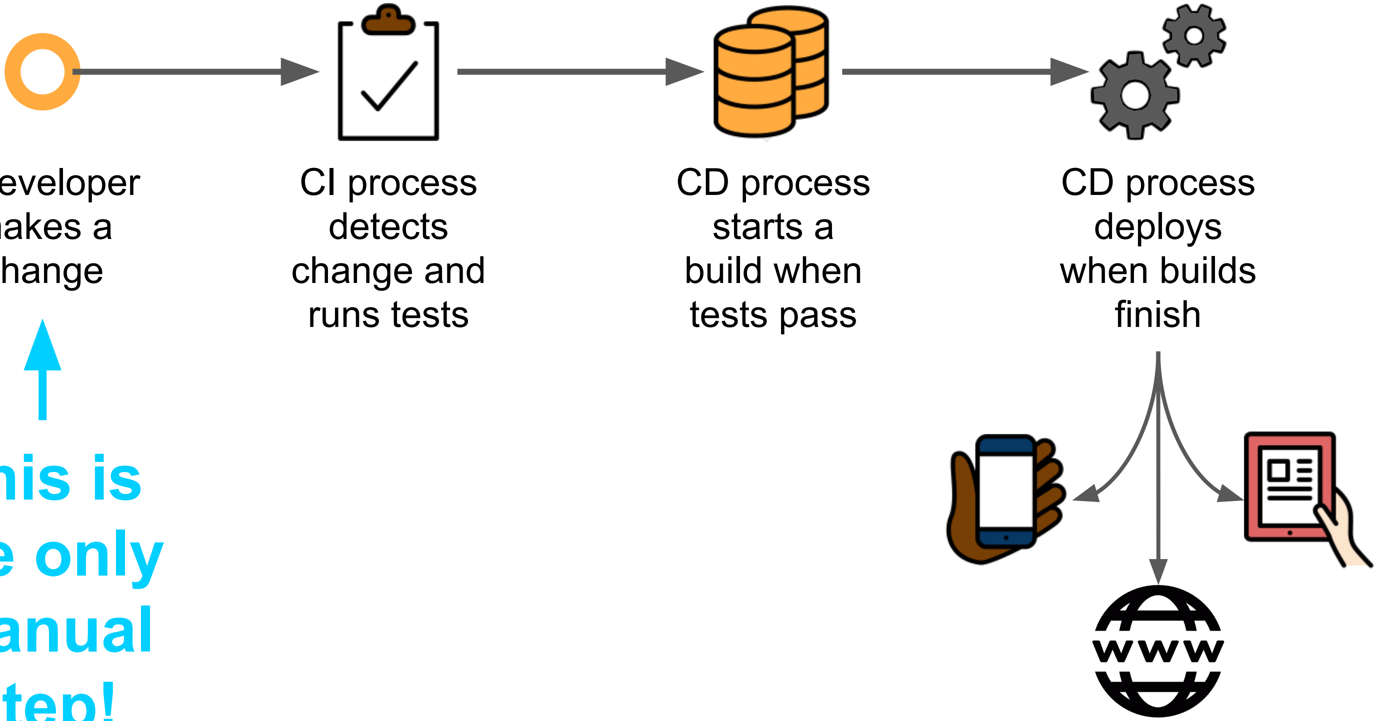
CI process detects change and runs tests

CD process starts a build when tests pass

An automated CI/CD process

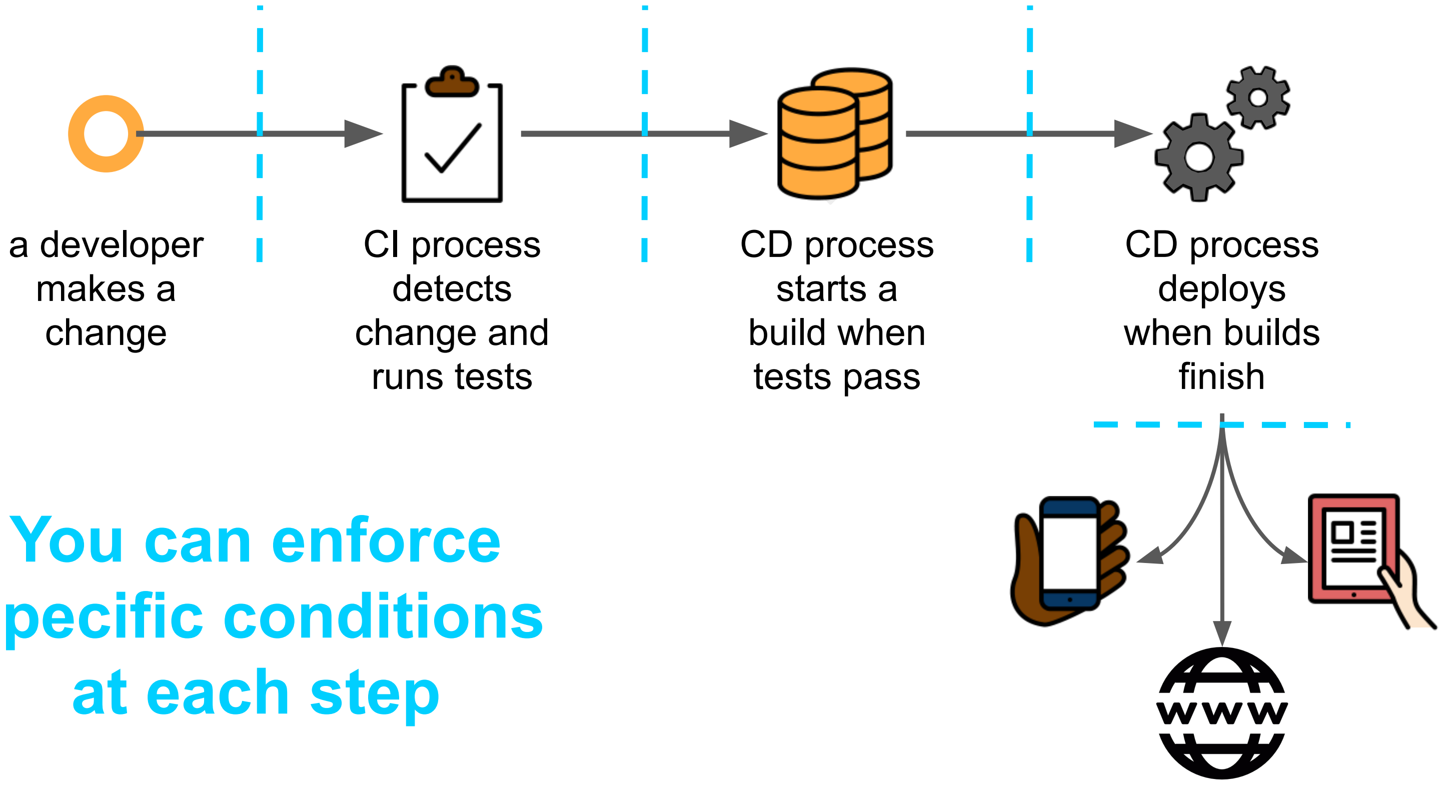

a developer makes a change
CI process detects change and runs tests
CD process starts a build when tests pass
CD process deploys when builds finish
www

An automated CI/CD process


a developer makes a change
CI process detects change and runs tests
CD process starts a build when tests pass
CD process deploys when builds finish
www

This is the only manual step!

An automated CI/CD process


a developer makes a change
CI process detects change and runs tests
CD process starts a build when tests pass
CD process deploys when builds finish
www

You can enforce specific conditions at each step

Why is automation good?

Computers are really good at doing **repeated** tasks the same way each time.

Automation can be audited for **policy and security** compliance.

Automation scripts are treated like code and maintained with **version control**.

Dev, sec, and ops are more **efficient**.

Side effect of insisting on these practices...

Helps attract
best-in-their-field
technology vendors.

18F

Culture
Practices
Tools

Choosing the tools comes last!

There are lots and lots of great tools out there for implementing any/all of these DevSecOps practices.

Which you choose depends on your tech stack, your project needs, and team preferences.

Question.

What about a legacy system with none of these tools/practices currently in place?

Start small, but start somewhere.

1 automated test > 0 automated tests

1 continuous integration pipeline > 0 pipelines

Practice, reap the benefits, and repeat!

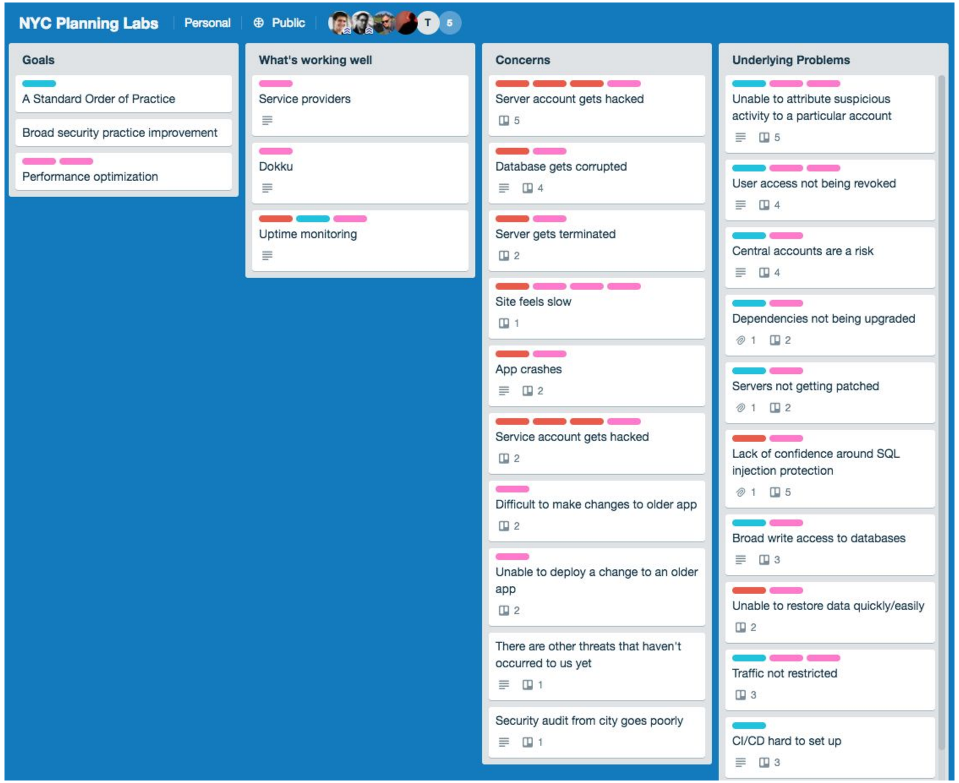
Question.

What kind of things does the security team do in a DevSecOps team?

Identify & prioritize security improvements

Case study:

DevOps and Security on a Small Team by Aidan Feldman, 18F alum



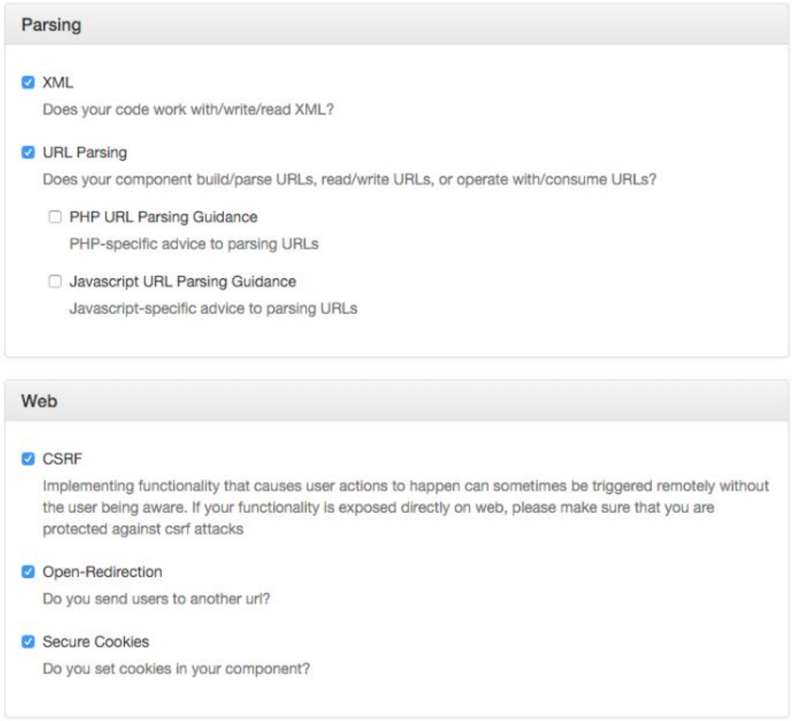
Goals
- A Standard Order of Practice
- Broad security practice improvement
- Performance optimization

What's working well
- Service providers
- Dokku
- Uptime monitoring

Concerns
- Server account gets hacked 5
- Database gets corrupted 4
- Server gets terminated 2
- Site feels slow 1
- App crashes 2
- Service account gets hacked 2
- Difficult to make changes to older app 2
- Unable to deploy a change to an older app 2
- There are other threats that haven't occurred to us yet 1
- Security audit from city goes poorly 1

Underlying Problems
- Unable to attribute suspicious activity to a particular account 5
- User access not being revoked 4
- Central accounts are a risk 4
- Dependencies not being upgraded 1 2
- Servers not getting patched 1 2
- Lack of confidence around SQL injection protection 1 5
- Broad write access to databases 3
- Unable to restore data quickly/easily 2
- Traffic not restricted 3
- CI/CD hard to set up 3

Build checklists & lightweight processes



Parsing

- [x] XML
  Does your code work with/write/read XML?
- [x] URL Parsing
  Does your component build/parse URLs, read/write URLs, or operate with/consume URLs?
  - [ ] PHP URL Parsing Guidance
    PHP-specific advice to parsing URLs
  - [ ] Javascript URL Parsing Guidance
    Javascript-specific advice to parsing URLs

Web

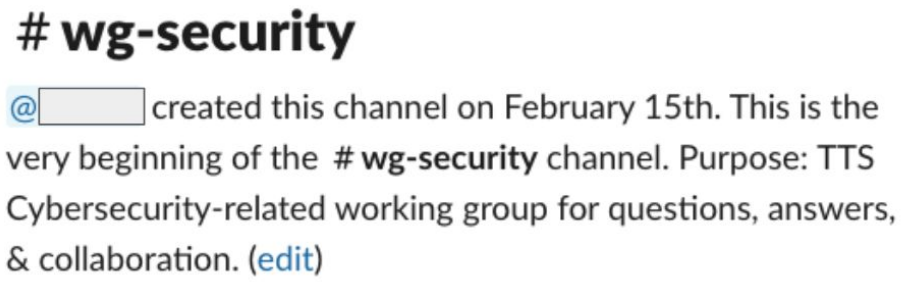
- [x] CSRF
  Implementing functionality that causes user actions to happen can sometimes be triggered remotely without the user being aware. If your functionality is exposed directly on web, please make sure that you are protected against csrf attacks
- [x] Open-Redirection
  Do you send users to another url?
- [x] Secure Cookies
  Do you set cookies in your component?

Case study:

Moving Fast and Securing Things
by Max Feldman, Slack

Collaborate, teach & learn

Example: TTS Slack (18F's parent agency)



wg-security

@[redacted] created this channel on February 15th. This is the very beginning of the # **wg-security** channel. Purpose: TTS Cybersecurity-related working group for questions, answers, & collaboration. (edit)

🔑 Key takeaways

DevSecOps enables continuous delivery of valuable, safe, working software.

DevSecOps is about a team's culture, practices, and tools (in more or less that order).

18F

Questions!

What resonates with you here?

What seems similar to the way you currently deploy code on your teams? What seems different?

What ideas or questions does this spark for you?