Improving developer experience with a developer platform

DevOps Enterprise Summit Las Vegas 2023

Ian Eslick | Chief Technology Officer
Antonio Beyah | Senior Principal Engineer; Developer Platforms
Levi Geinert | Head of Engineering Advocacy and Innovation

U.S. Bank facts

Headquartered
in Minneapolis, Minnesota

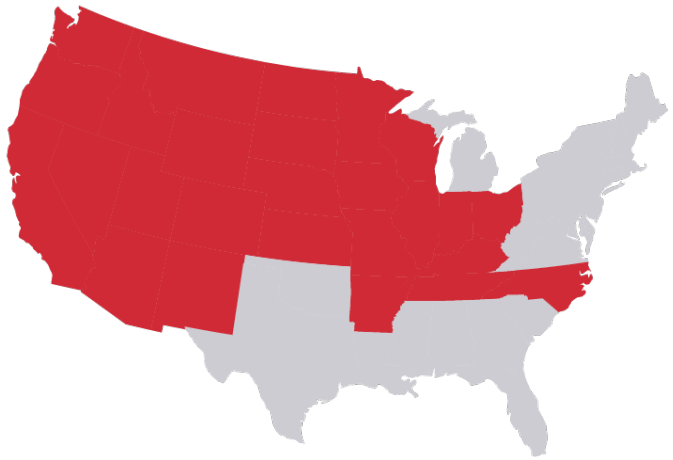
More than 70,000
team members

Fifth largest
commercial bank in U.S.

Where we are

Regional

Consumer & Business Banking and Wealth Management



National

Corporate & Commercial Banking and Wealth Management & Investment Services

International

Payment Services and Investment Services

Our technology accolades

We deliver industry-leading, **innovative** and **resilient technology** to over 40M customers nationally.

Best in Class Mobile App

- Mobile App rated 5 out of 5 stars by more than 1 million users

Best in Industry
Digital Mortgage Tools
Kiplinger

#1
Mobile Check Deposit
CORNERSTONE ADVISORS

#1
Customer Service Features
BUSINESS INSIDER

Ample DIY and DIT Services

DIY

DIT

- Smart Assistant has answered more than 1 million questions since launched in summer 2020

#1
Overall Superregional Bank
FORTUNE

Partnerships and acquisitions

UnionBank
CenPOS
APEX Fintech Solutions™
Azure
talech
Opayo By Elavon
bento for business
TravelBank

2022 Awards

Awarded 2 Model Bank Awards for innovative banking technology

2022 Impact Award for Operational Efficiency

2022 WORLD'S MOST ETHICAL COMPANIES® ™
WWW.ETHISPHERE.COM

History

- 2018 - Digital Transformation and Experience Studios
- 2020 – Covid 19 kicks off a broad Technology Transformation effort
- 2022 – Announced Cloud partnerships and transformation acceleration
- 2023 - Started deploying applications onto our new developer platform

Pillars of our Transformation

Product Model

Align behind customer needs and establish clear, unified ownership of outcomes across business and technology.

Process Re-engineering

Can’t overhaul technology without rethinking our processes and procedures: standardize, consolidate, automate, and instrument.

Platform Strategy

Provide an extensible foundations for standardized patterns to empower our software development teams. Reduce cognitive load and address regulatory concerns “by design”

Team Operating Model

Invest in training, coaching, experiential learning, documentation, and rich measurement to help teams learn to operate

Developer platforms and platform engineering

The Developer Platform

Run Your Platform like a Business within a Business

Rosalind Radcliffe, Charles Betz, Betty Junod, Ravi Maduposu, Luke Rettig, Mark Imbriaco, Levi Geinert

Free

Get

Share:

The Role of a Platform

Enabling Full-Stack Software Engineers to Their Fullest Potential

Satya Addagarla, Josh Atwell, Mik Kersten, Thomas A. Limoncelli, Sara Mazer, Steve Pereira, Jeff Tyree, and Christina Yakomin

Free

Get

Share:

DevOps is dead, long live Platform Engineering!

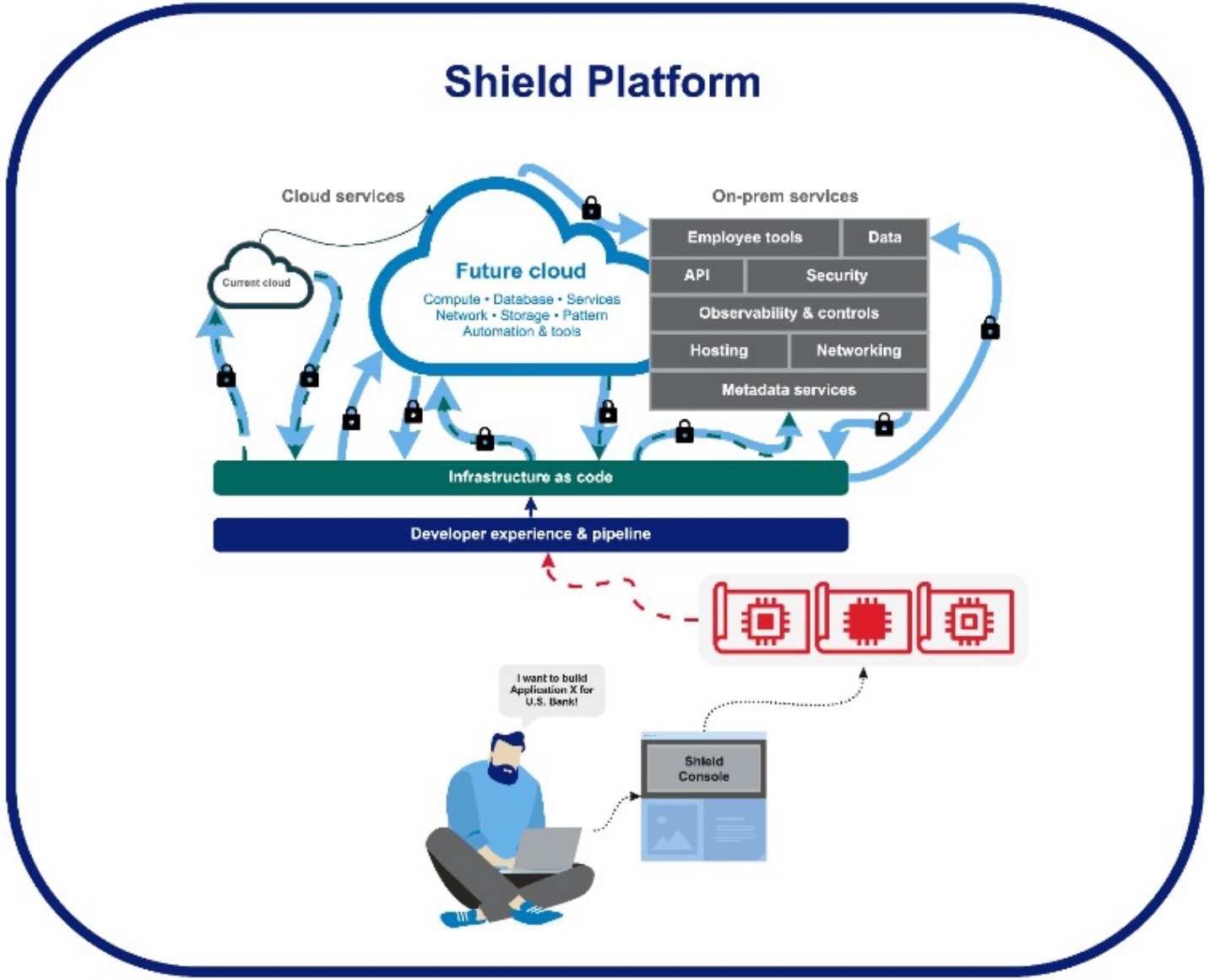
True DevOps - “you build it, you run it” - has been the guiding methodology behind many development teams for years. However, the harsh reality is that most developers don’t like dealing with infrastructure. It isn’t just a source of significant cognitive load, but it also takes time away from their core job of writing, debugging, and running applications.

Shield Platform

Our https://internaldeveloperplatform.org/

Shield Platform

Driving business value through more efficient software engineering



The case for building a platform

Platform capabilities are critical to our ability to scale to our entire company

1

Self-Service

Every common platform should be designed to be **easy to use and well-documented**. Most users should use the platform without ever needing to talk to the platform team

2

Extensible

Platforms enable developers to naturally handle corner cases by providing **extension points** that allow developers to extend the behavior without having to modify the core system.

3

Inner Source

If a common platform lacks a feature you need and extensions are not enough, you should have the freedom to **improve the platform** by submitting a code review request to the core team.

"Good architectural design enables you to free developers while improving the way that we meet our regulatory obligations. It's easy to prioritize one over the other - a good developer platform, particularly for a bank, does both." – Ian Eslick

Developer benefits

Improved experience

Reduce time from weeks to minutes for a developer to spin up new instances, build a new app, and deploy it to Dev

Improved security and resiliency

Increase standardized adherence to operational and risk standards

Improved reliability

Eliminate manual steps and reviews needed to assure compliance with risk, security and operations requirements

What does Shield Platform encompass?

Shield Console

Shield Console will be the main hub for USB engineers to create and manage projects, monitor status of applications, view logged issues, see deployment status, and access documentation

Platform Services

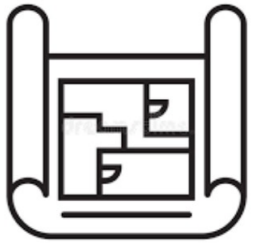
Align all infrastructure services consumed by application engineers. Data as a service, containers, logging, alerting, CI/CD, certificate management, and much, much more…

Declarative Config

Automate configuration of Shield and cloud provider services via declarative configurations that are executed by a shared pipeline. Compliance evidence and enforcement are baked in.

Blueprints

A formal wrapper for our platform services and/or infrastructure as code which formalizes how controls are validated and inherited by applications to ensure standardized use of our common controls



Platform Pillars

Foundational Elements

- Authentication & Authorization
- Client Libraries & App Framework
- Secrets Management
- Monitoring & Metrics
- Asset Management

Continuous Integration (CI)

- Console Output & Formatting
- Portability
- Artifact Promotion
- Consistent and Repeatable Artifact Creation
- Merge/Pull Request Support

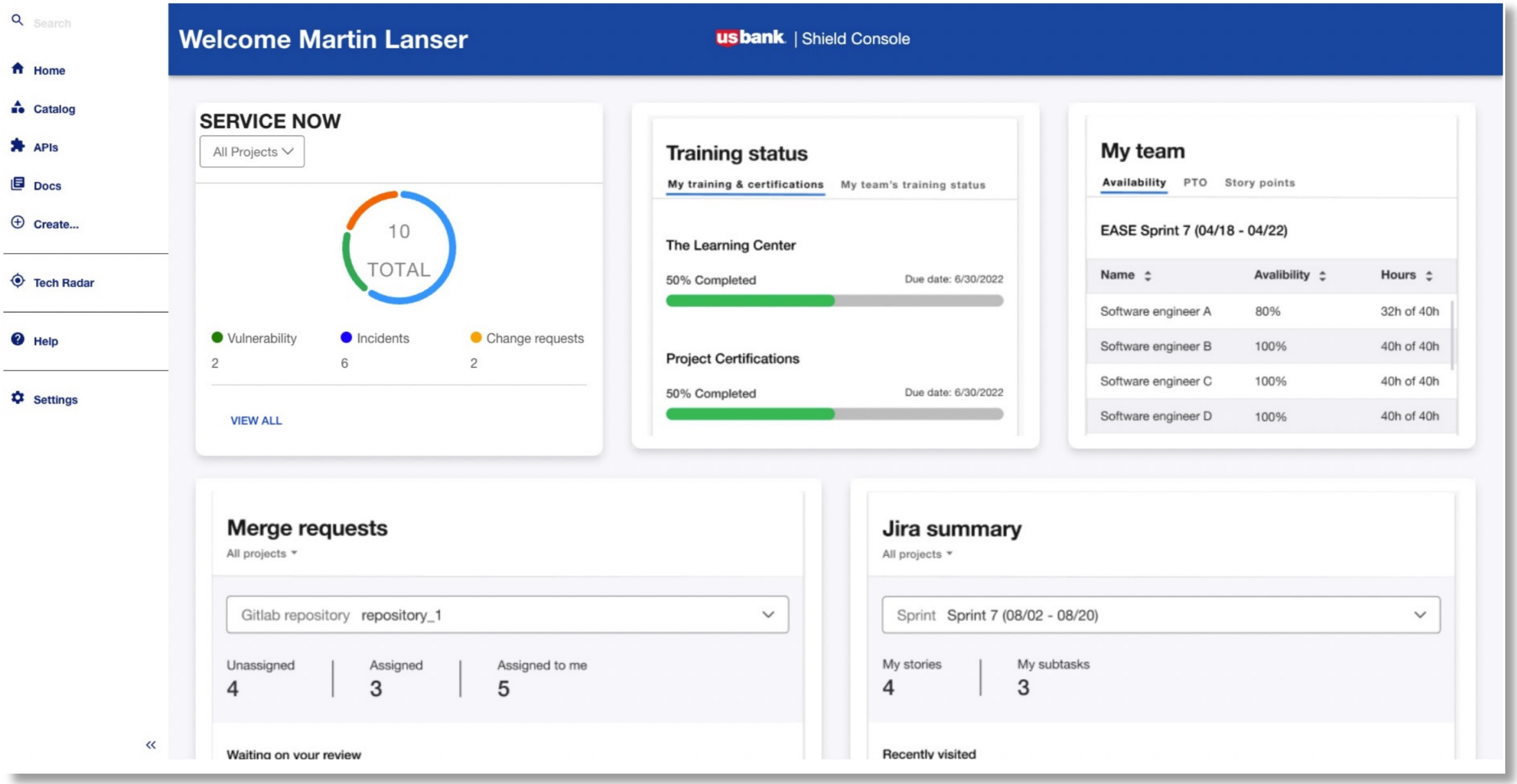
Continuous Delivery (CD)

- Repeatable & Reversible Deployments
- API Driven
- Self-Service Operations
- Preview Environments
- Runtime Configuration

Operations

- Software/API Catalog
- Software Lifecycle
- Vulnerability Management
- Infrastructure
- Compliance

Developer Console



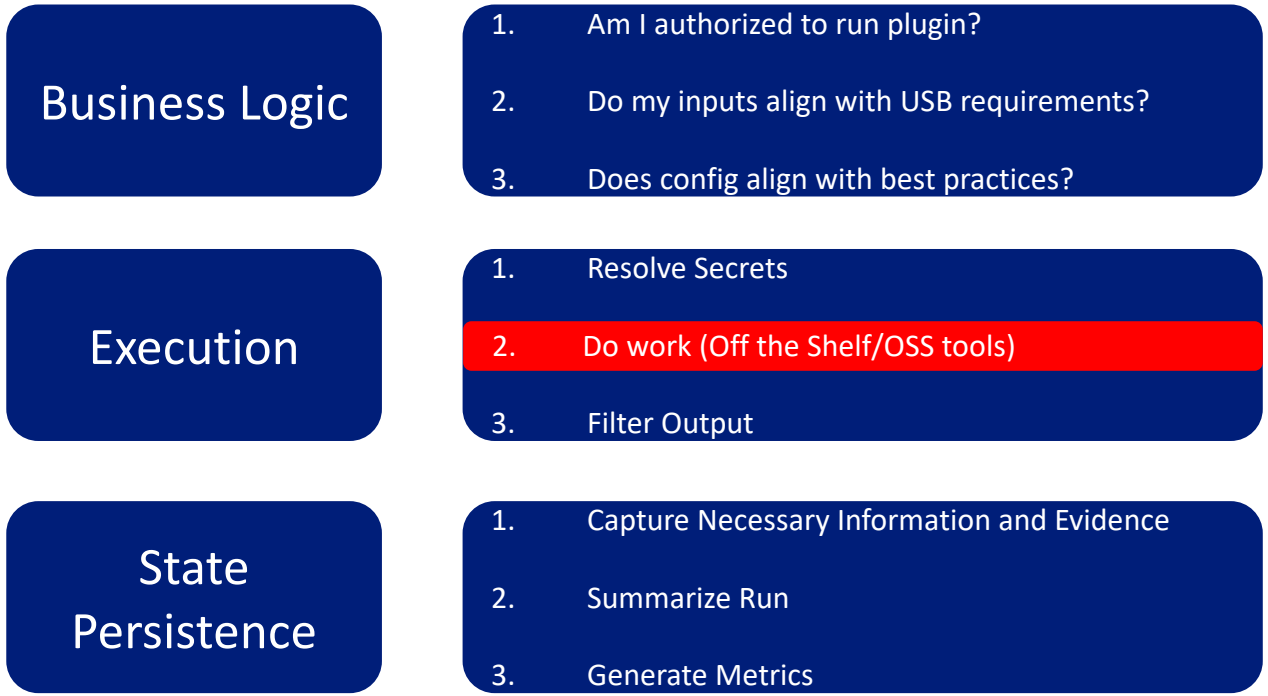
PipelineCLI

The build logic for core primitives of the platform (build, scan, artifact storage) in a **portable** manner and **enabling** local development.

Externalizing our business logic while increasing observability, and enabling control usage through module abstractions.

Pipeline Module standards:

- Configuration via Environment Variables
- Input Validation
- Evidence integration
- Secrets Integration
- Developer Access Control authorization
- Unit Testing
- Local and CI Functional Testing
- Documentation Usage & Expected Output



PipelineCLI Usage

Using the templates

To use a template, you must first `include` it into your `.gitlab-ci.yml`:

```
include:
  - project: 'engineering/pipelinecli'
    file: '/templates/plugins/plugin-steps.yml'
    ref: rc
```

Then extend the job defined in the template and set required inputs as `variables`:

```
build-image:
  extends: .kaniko-build-with-artifactory-template
  stage: kaniko
  variables:
    KANIKO_IMAGE_NAME: "project-name"
    SHIELD_TEAM: "project-namespace"
    KANIKO_IMAGE_VERSION: "1.0.0"
```

Metrics

All plugins output metrics related to the success of the plugin execution and how long of those executions took. The below chart shows the metrics being generated and information about them.

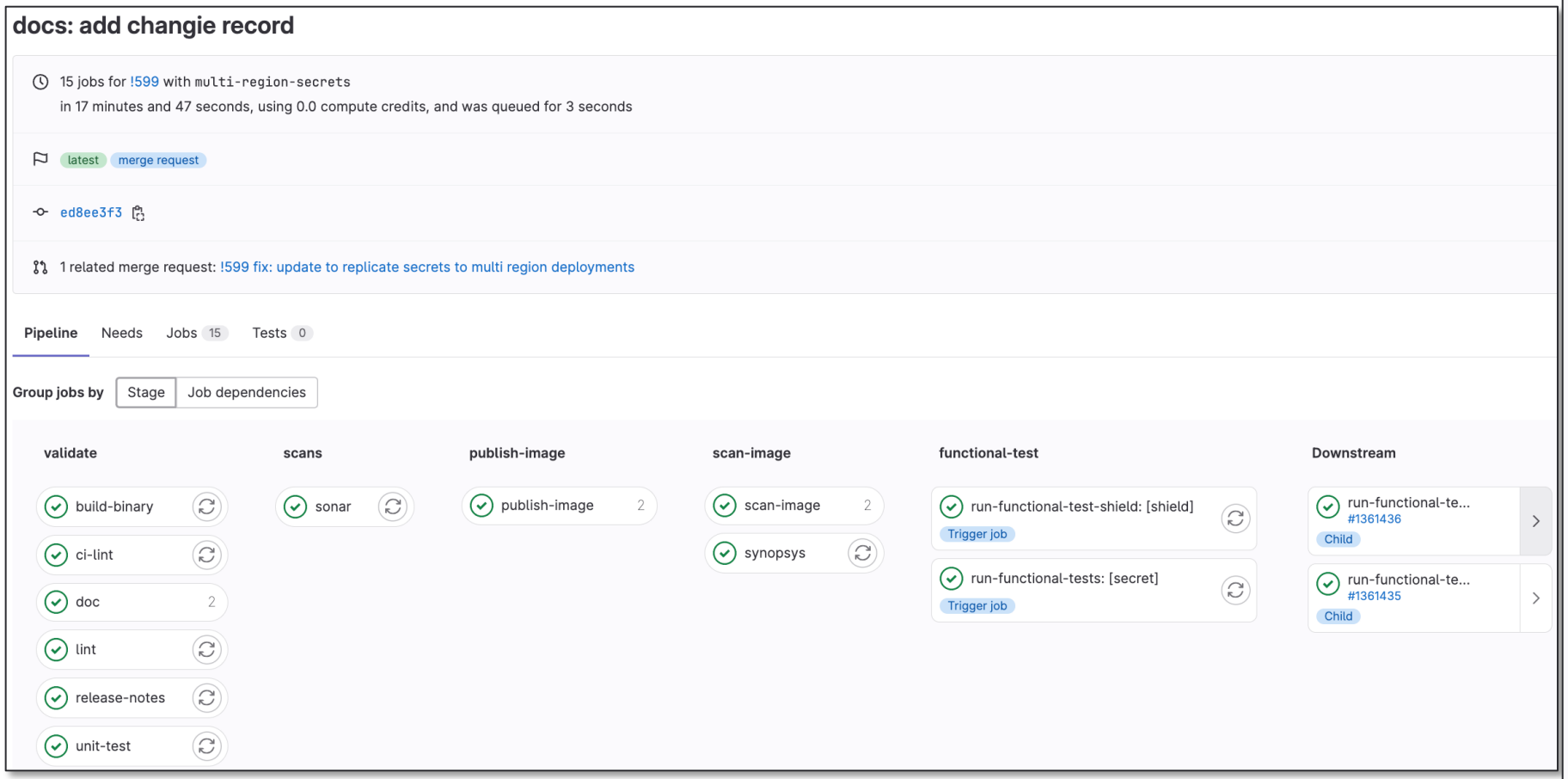
name	type	attributes	description
pipeline.plugin.execution	histogram	service, car.id, build.pipeline.type, pipeline.trigger, repo.path, job.url, pipeline.url, merge.request.build, plugin, action, success	Execution Timings of plugin
pipeline.plugin.step.execution	histogram	service, car.id, build.pipeline.type, pipeline.trigger, repo.path, job.url, pipeline.url, merge.request.build plugin, action, success, step	Execution Timings of plugin step

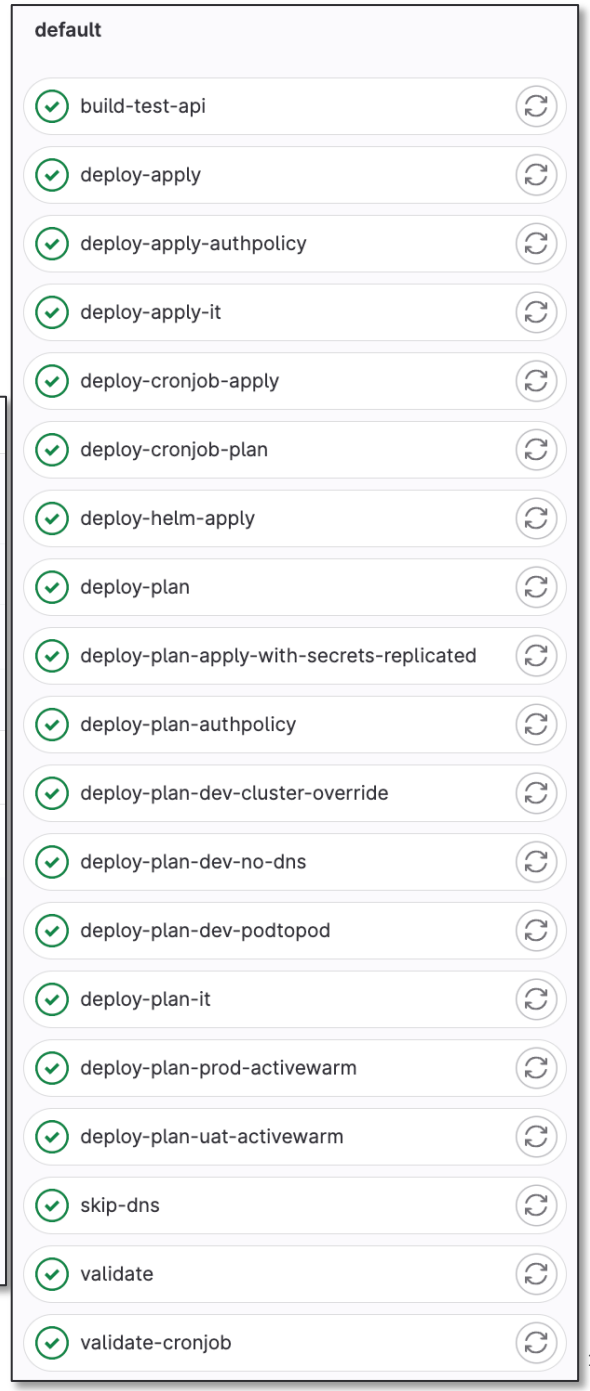
Plugins

Plugin	Description	Technology Support
artifact	The `artifact plugin` is used for managing artifacts. The primary use-case is container image promotion from non-prod to prod registry.	Language Agnostic
dotnet	The `dotnet plugin` is used for building .NET 6+ projects.	dotnet supported languages
kaniko	The `kaniko plugin` is used to build and push container images to the container image registry	Language Agnostic
maven	The `maven plugin` is used to run a mvn command.	maven supported languages
gradle	The `gradle plugin` is used to run a gradle command.	gradle supported languages
nodejs	The `nodejs plugin` is used to install npm packages and execute scripts using NPM and Yarn.	nodejs supported versions
packer	The `packer plugin` is used to build and push Virtual Machine Images.	Language Agnostic
secret	The `secret plugin` is used to replicate secrets from hashicorp to secret providers such as Azure KeyVault.	Language Agnostic; Only has support for container deployments
shield	The `shield plugin` is used to perform deployments to a runtime environment such as Azure/Kubernetes.	Language Agnostic
sonar	The `sonar plugin` is used to perform a SonarQube scan (code quality) on source code.	sonar supported languages
synopsys	The `synopsys plugin` is used to start a Synopsys IO scan on the Synopsys IO Jenkins server. This replaces the Black Duck and Fortify scans in your pipeline.	synopsys supported languages
terraform	The `terraform plugin` is used to run a terraform command.	Language Agnostic; Helps support cloud infrastructure provisioning
twistlock	The `twistlock plugin` is used to scan container images for vulnerabilities.	Language Agnostic
servicenow	The `servicenow plugin` is used to enforce the rules for blocking deployments going to uat/prod to ensure that a proper change request / change task exists	Language Agnostic
helm	The `helm plugin` is used to run test and package helm charts for bring your own helm chart deployments	Language Agnostic

PipelineCLI Module Development

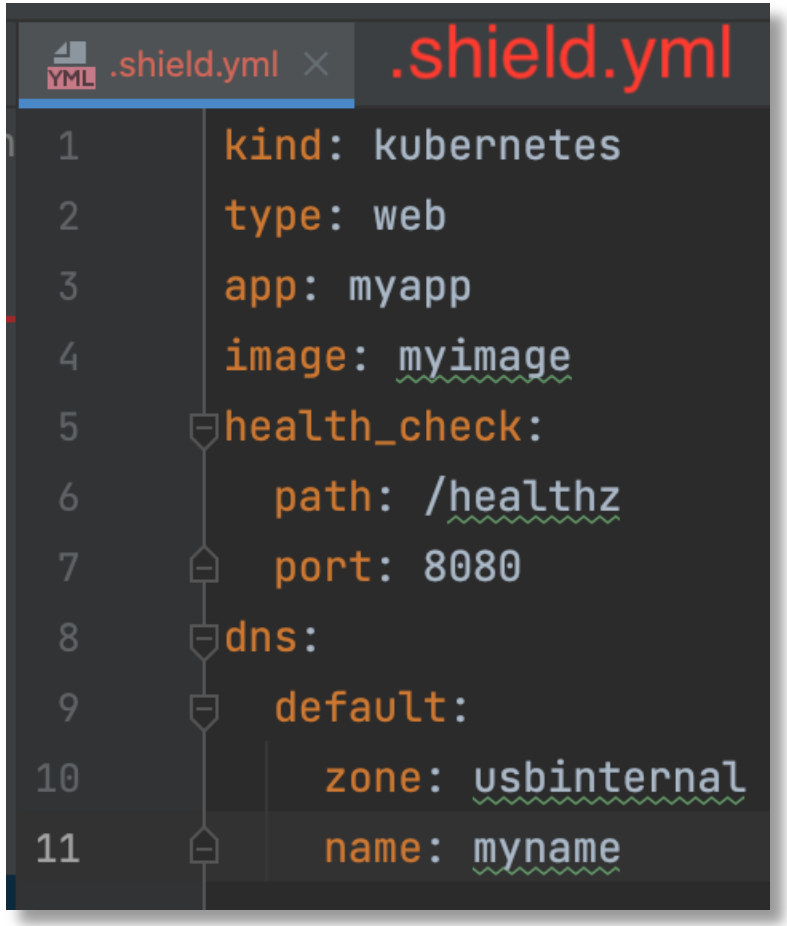
Strong software engineering practices applied to platform tooling development

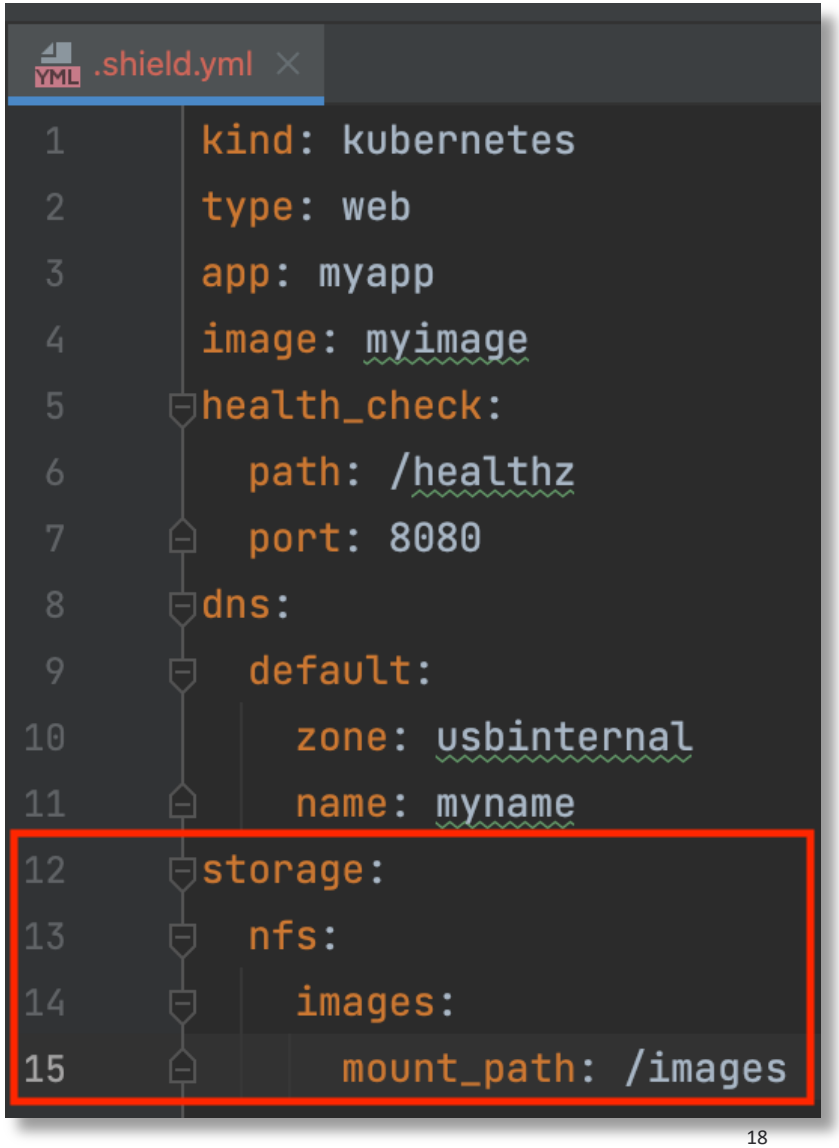




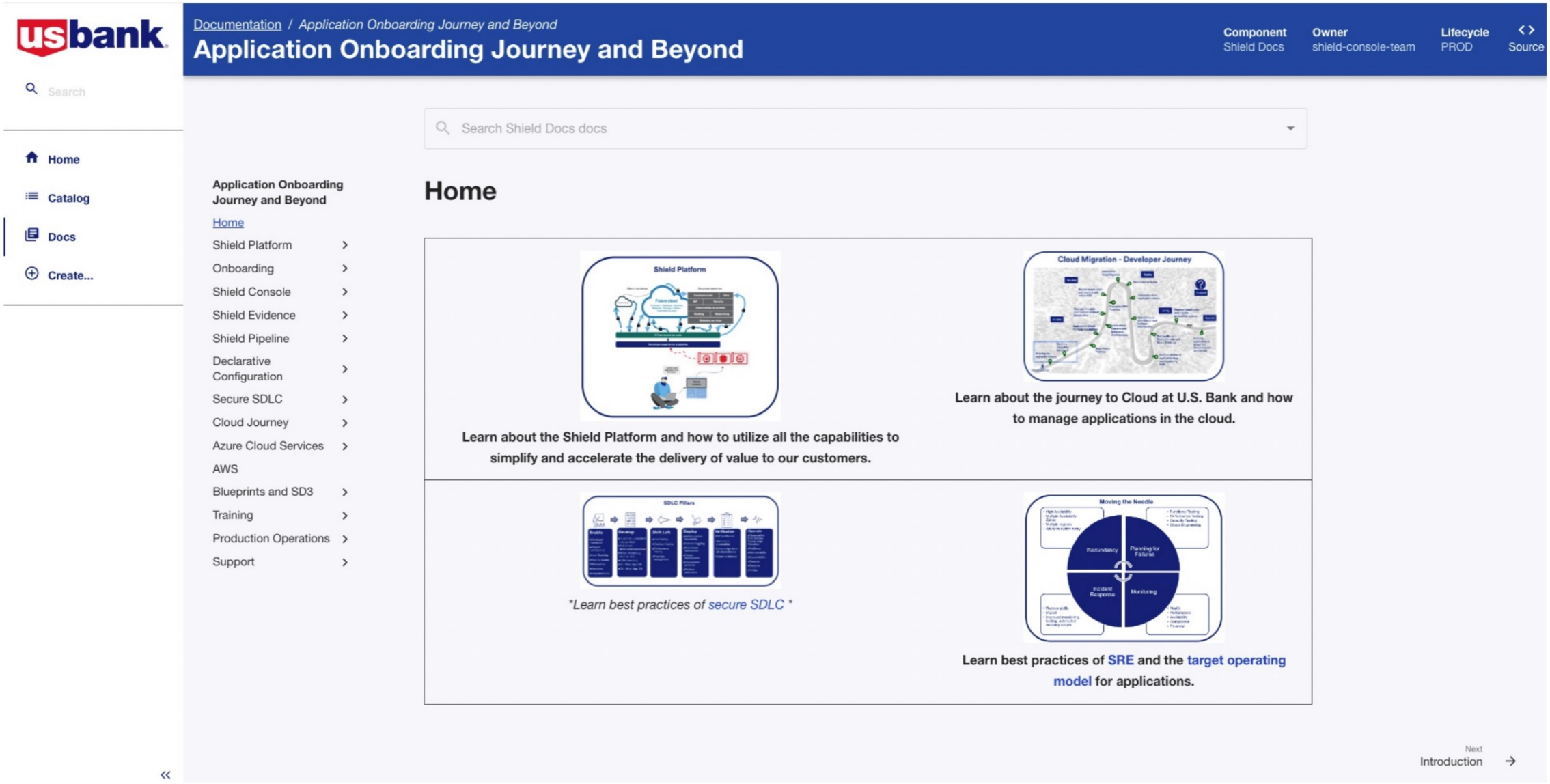
Declarative Configuration

A tooling-agnostic desired state declarative configuration format where application teams provide only the information they need to get up and running.





Automated Documentation in Backstage



Current Progress

- Continuous Integration Plugins for building / scanning / publishing of artifacts
- Declarative Configuration supports automated deployments
- Robust secrets management capabilities enable centralized management, streamlined operations, and automated rotations
- Regulatory evidence is collected by default

Customer Feedback

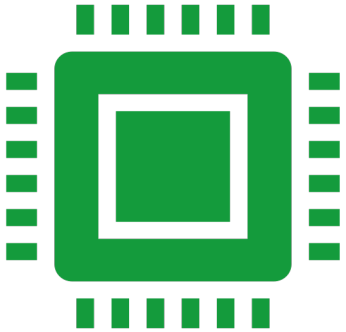
SD3 equals massive efficiency gains over the legacy SD2 process. It greatly reduced the documentation and review burden on all parties involved.

I found the Shield documentation and tools are so useful and provides lot of examples that I had to interact with team very rarely. Documentation was very much sufficient to complete our POC and expand our footprint in cloud.

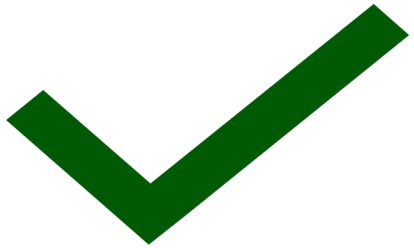
Elimination of certification updates and change requests is a huge time savings for our team with Shield Platform as well as the 5-10x performance gains with the migration to Azure.

Some key accomplishments to date…

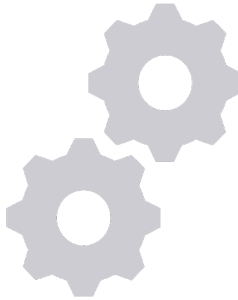
Support for container-based and VM deployments and Data Services



Robust infrastructure



Resiliency built in from day one



Risk and Compliance baked in

Challenges Faced, and Remaining

- Building a platform team
- Building while operating
- Upskilling our organization
- Enabling adoption at scale

Start discovering how you'll thrive

Apply at **careers.usbank.com**

An equal opportunity employer/disability/veteran.


us bank