# LAUNCHPAD

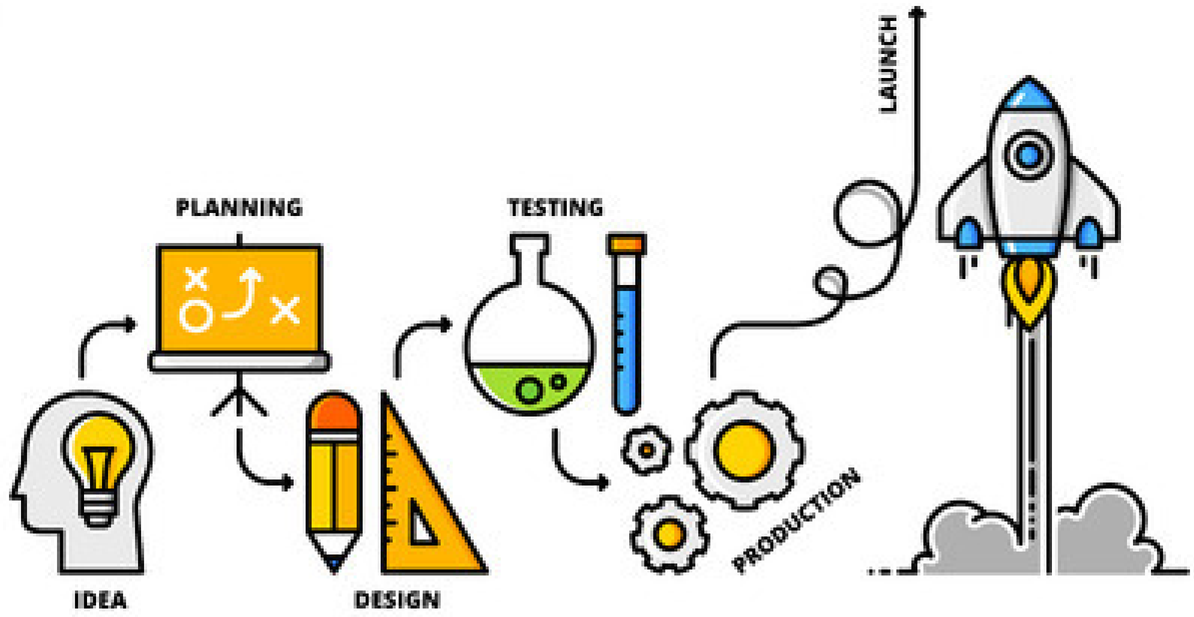
## A PLATFORM FOR APPLICATION DEVELOPMENT

**James Bennett**
Enterprise Platforms
Indiana University

This presentation is best viewed online at https://indiana-university.github.io/internet2-techex-2022-launchpad/

## Enterprise Platforms

- Part of central IT at IU: UITS
- Combination of several former groups:
    - Systems Integration/Middleware
    - Linux Administration
    - Database Administration
    - Cloud Support


PLANNING
TESTING
LAUNCH
IDEA
DESIGN
PRODUCTION

## A relatable story…

- Mid-May 2021 IU HR decides on a new remote work arrangement policy
- Need a system to track these arrangements
- Requirements lead to the need for a custom application
- Opening to all staff and faculty July 1st (~six weeks later)

## Six short weeks

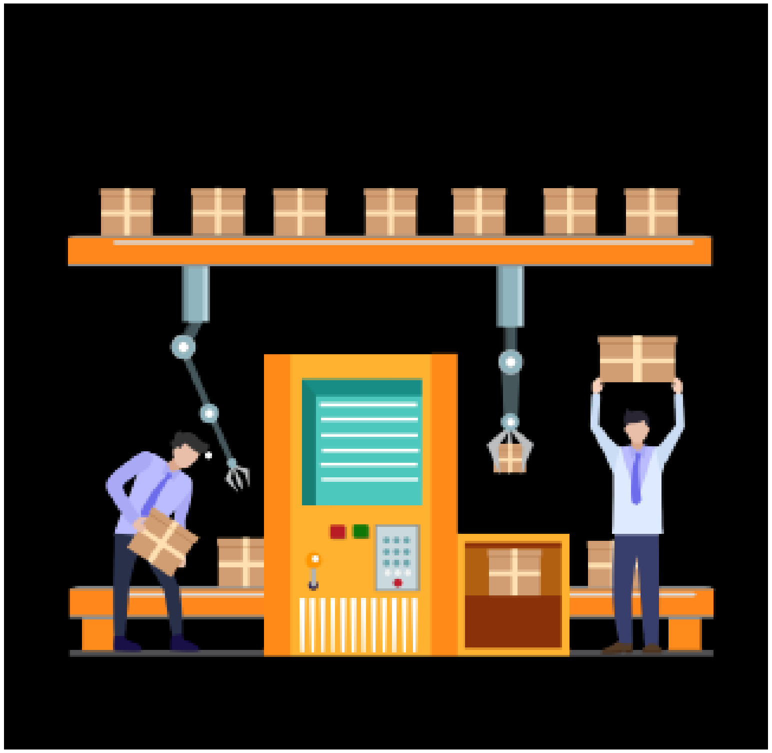
- Solution based on **platform engineering**
- Guided by Cloud principles
- Focused on developer experience
- Opens up a path for migrating central IT workloads to the cloud

# Platform Introduction

> *“A **digital platform** is a foundation of self-service APIs, tools, services, knowledge and support which are arranged as a compelling internal product. Autonomous delivery teams can make use of the platform to deliver product features at a higher pace, with reduced co-ordination.”*
>
> — ***Evan Bottcher***

## Why a platform?

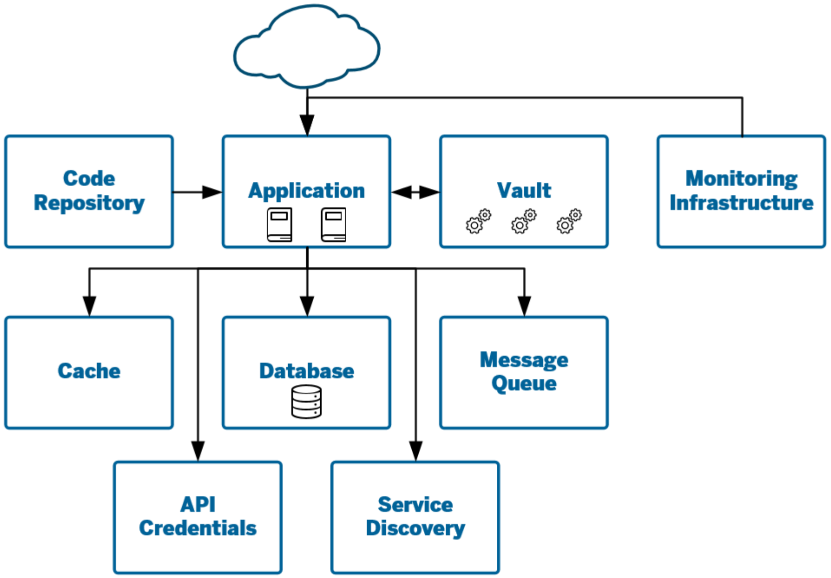
- Empower development teams
- Reduce **cognitive load**
- Tie everything together
- Amplify best practices
- Take full advantage of our infrastructure



# What we're building

## Thinnest viable platform

- Single stop for all services
- Related documentation pulled together for easy consumption
- Support forms built with low-code system (Fireform)
- Connected to workflows to automate (**Argo**)



Code Repository

Application

Vault

Monitoring Infrastructure

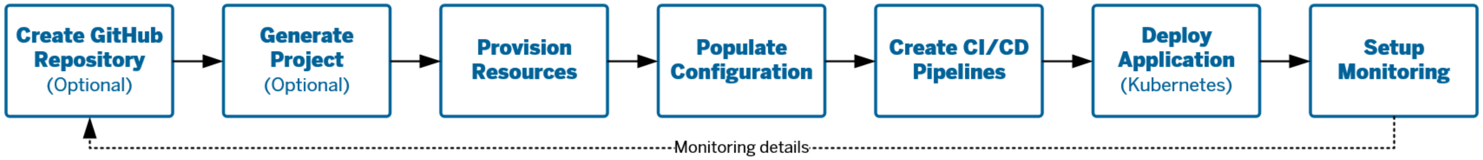
Cache

Database

Message Queue

API Credentials

Service Discovery



Create GitHub Repository (Optional)

Generate Project (Optional)

Provision Resources

Populate Configuration

Create CI/CD Pipelines

Deploy Application (Kubernetes)

Setup Monitoring

Monitoring details

## 🎵 Woah, we're halfway there

- Delivered a project skeleton and infrastructure components iteratively
- Everything delivered shortly after Memorial Day
- Developers focused on development
    - Worked with user experience group on design
    - Accessibility review
    - Security review
    - Automated testing (after the release)

# Benefits

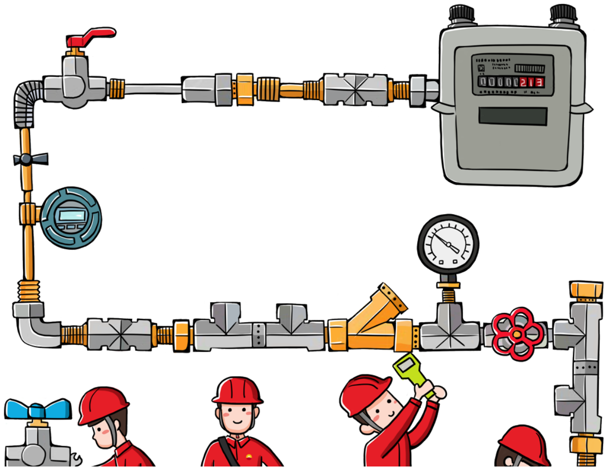
## Reduce toil

- Reduce operational complexity
- Automated, timely feedback loops
- Streamline cross-team operations

> *“If a human operator needs to touch your system during normal operations, you have a bug. The definition of normal changes as your systems grow.”*

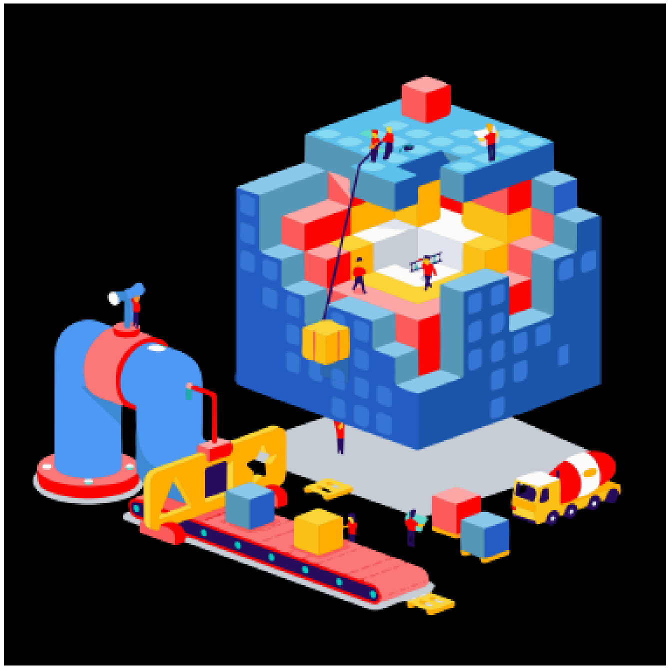
# Maintainability

- Everything is code and versioned together
- Reduced configuration
- Apply upgrades more often



## Deployability

- One click application deployments
- Encourage automated testing & peer review
- Advanced deployment capabilities
- Ephemeral environments in the future

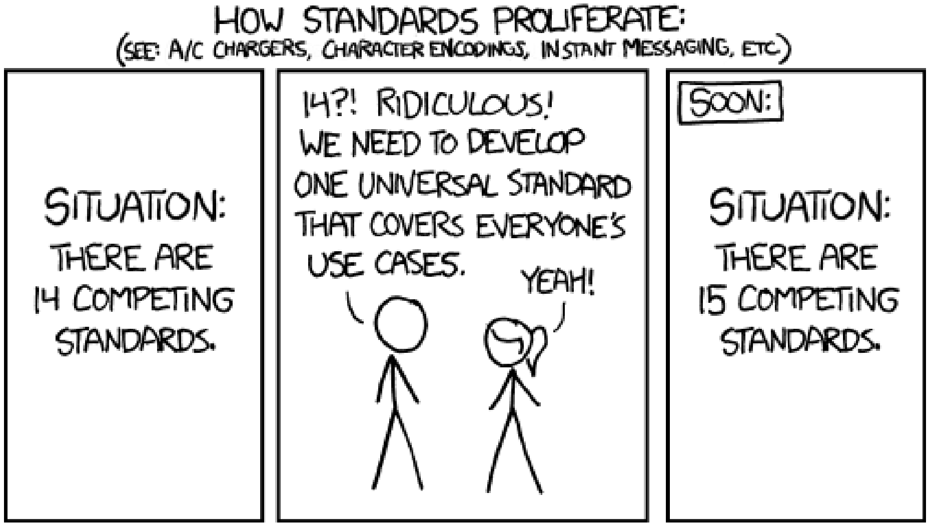


## Security

- Role-based security model based on Grouper and eventually midPoint
- Provision access to all components
- Platform managed secrets
- Hardened delivery chain
- Documentation handled as much as possible

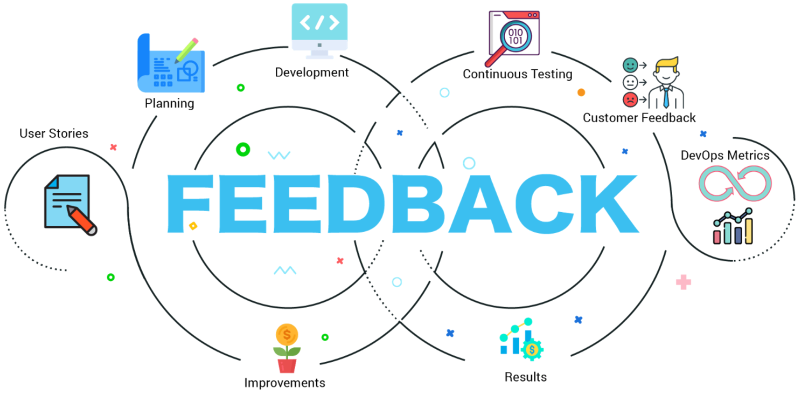
# Standards

- Macro-level standards (e.g. 12-factor apps)
- Make the right thing the easiest thing
- Maturity models


HOW STANDARDS PROLIFERATE:
(SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC)
SITUATION: THERE ARE 14 COMPETING STANDARDS.
14?! RIDICULOUS! WE NEED TO DEVELOP ONE UNIVERSAL STANDARD THAT COVERS EVERYONE'S USE CASES.
YEAH!
SOON:
SITUATION: THERE ARE 15 COMPETING STANDARDS.

# Insight

- Standard logging & metrics
- Automated monitoring & alerting
- Data-driven objectives



## Retention?

- Upskilling current team
- Overwhelmingly positive feedback

## Race to the finish

- App was delivered in production a week early
- No death march
- Most stable applications they maintain
- Three more apps delivered since in this area
- Working to replicate this at scale

# Technology stack

- Built with Kubernetes using reusable templates
- Uses Terraform (CDKTF) for provisioning and Ansible for configuration management
- Inventory management/tracking
- Event-driven architecture based on the operator pattern
- Workflow orchestration using Argo events/workflows
- Composing well-established components together

# What about the Cloud?

- Containerization is our path to the Cloud
- Abstractions/social contracts we have in place helps us prepare
- Already working to tie in Cloud account provisioning
- Upskilling reduces the anxiety about moving
- Supporting this kind of infrastructure in the Cloud is now part of our strategic plan