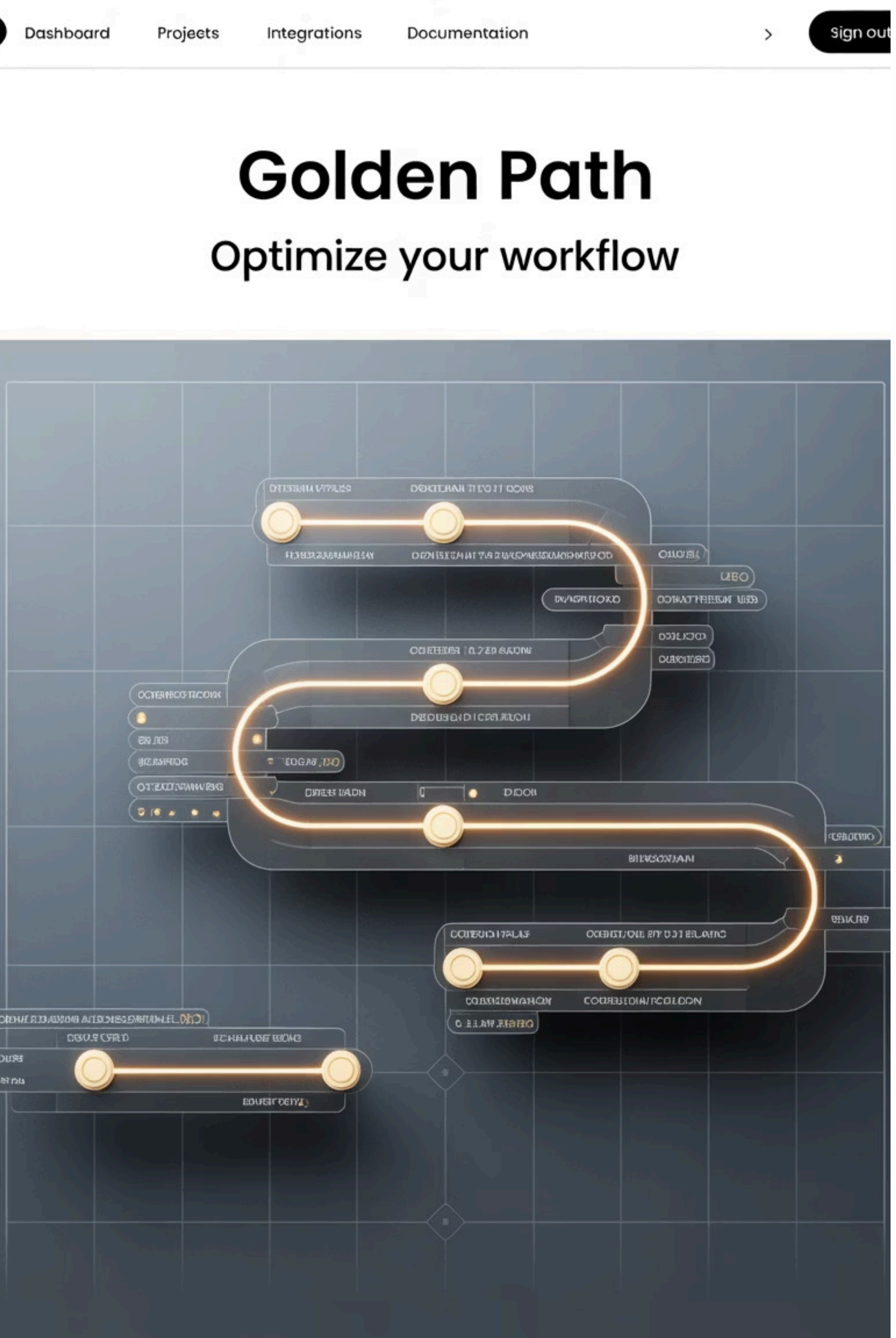


Golden Paths to Developer Success

Designing and Implementing Developer Experience (DevEx) in Internal Developer Platforms

Agenda

Golden Paths

What they are, why they matter, and how to design them effectively

Service Scaffolding

Implementing service scaffolding in Internal Developer Platforms with Backstage Software Templates

AI-Assisted Features

Enhancing developer experience with AI-assisted template scaffolding, recommendations, and ChatOps provisioning

Backstage Ecosystem

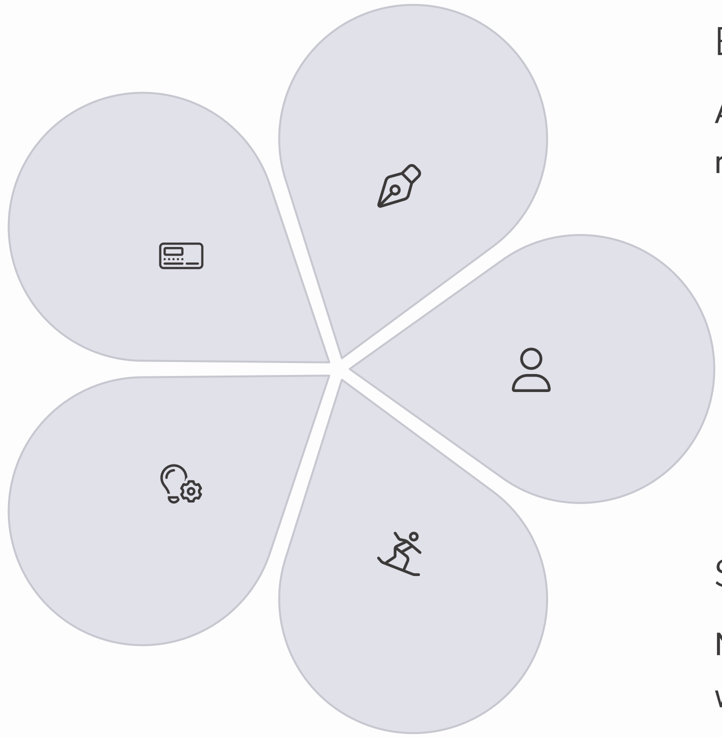
Overview of the Backstage IDP plugin ecosystem and future extension opportunities

What Are Golden Paths?

Golden Paths are **opinionated and supported paths** that make it easier for developers to build, test, and deploy software while meeting organizational requirements.

- Pre-approved architectures, technologies, and workflows
- Balance between standardization and innovation
- Reduce cognitive load for developers
- Accelerate time to production
- Bake in best practices and compliance requirements

The Value Proposition of Golden Paths



Accelerated Development

Reduce time spent on infrastructure decisions and setup by 30-50%

Built-in Compliance

Automatically incorporate security, legal, and regulatory requirements

Consistency

Standardized approaches reduce operational complexity and maintenance costs

Focused Innovation

Developers can focus creative energy on business problems, not infrastructure

Simplified Onboarding

New team members can contribute more quickly with clear pathways

The Golden Path Paradox

Too Restrictive

- Stifles innovation
- Frustrates experienced developers
- May not fit all use cases
- Creates "shadow IT" workarounds
- Becomes technical debt over time

Too Flexible

- Proliferation of technologies
- Inconsistent practices
- Higher operational costs
- Compliance risks
- Longer onboarding for new developers

The key challenge: Finding the right balance between standardization and flexibility

The Paved Road Approach

Netflix pioneered the "Paved Road" concept — a middle ground between rigid standardization and complete freedom:

Easy Default Path

Make the golden path the easiest option with excellent tooling and documentation

Clear Exit Ramps

Allow for deviation when justified, with documented processes for exceptions

Support All Roads

Provide baseline support for non-standard approaches, but prioritize the golden path

Designing Effective Golden Paths

Understand Your Developers

Study existing workflows, pain points, and needs through user research

Identify Common Patterns

Look for repetitive tasks and decisions that could be standardized

Start Small

Begin with one technology or service type; iterate based on feedback

Measure Impact

Define clear metrics to evaluate developer productivity and satisfaction

Continuously Evolve

Regularly update paths based on technology changes and developer feedback

Golden Path Implementation

Implementing golden paths in an organization requires a combination of:

Tooling

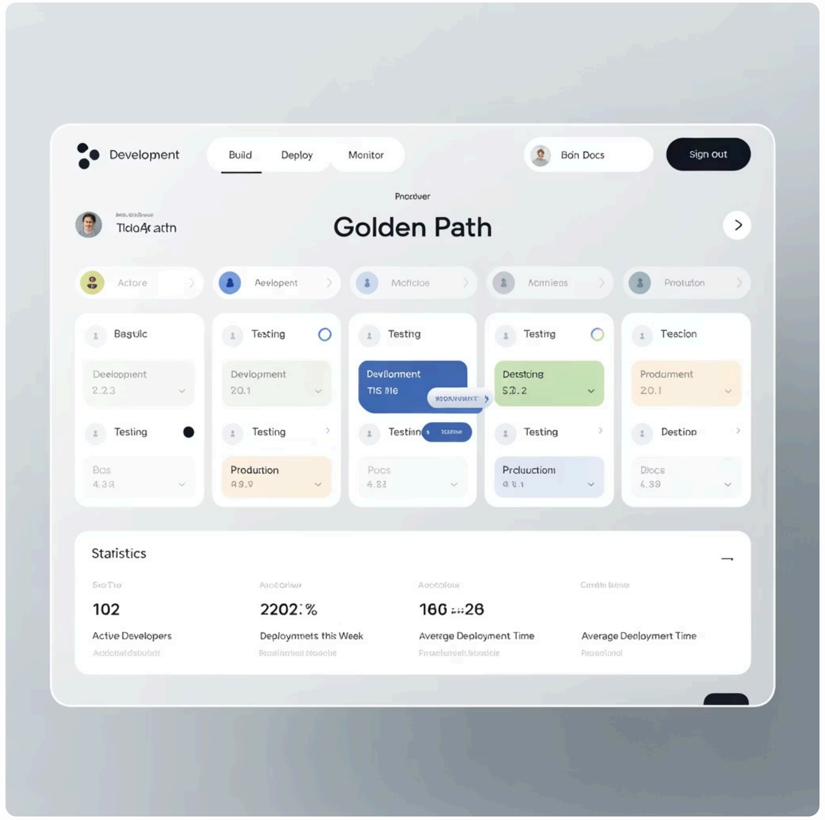
Internal Developer Platforms (IDPs) with templates and automation

Documentation

Clear, accessible guides and examples for developers

Culture

Developer-centric approach with feedback loops and continuous improvement



Internal Developer Platforms (IDPs)

IDPs serve as the technical foundation for implementing golden paths, providing a central interface for developers to interact with your organization's tech stack.

Self-Service

Developers can provision resources, create new services, and deploy applications without dependencies on other teams

Abstraction

Hide infrastructure complexity behind simple, consistent interfaces

Automation

Reduce manual steps through workflow automation and CI/CD integration

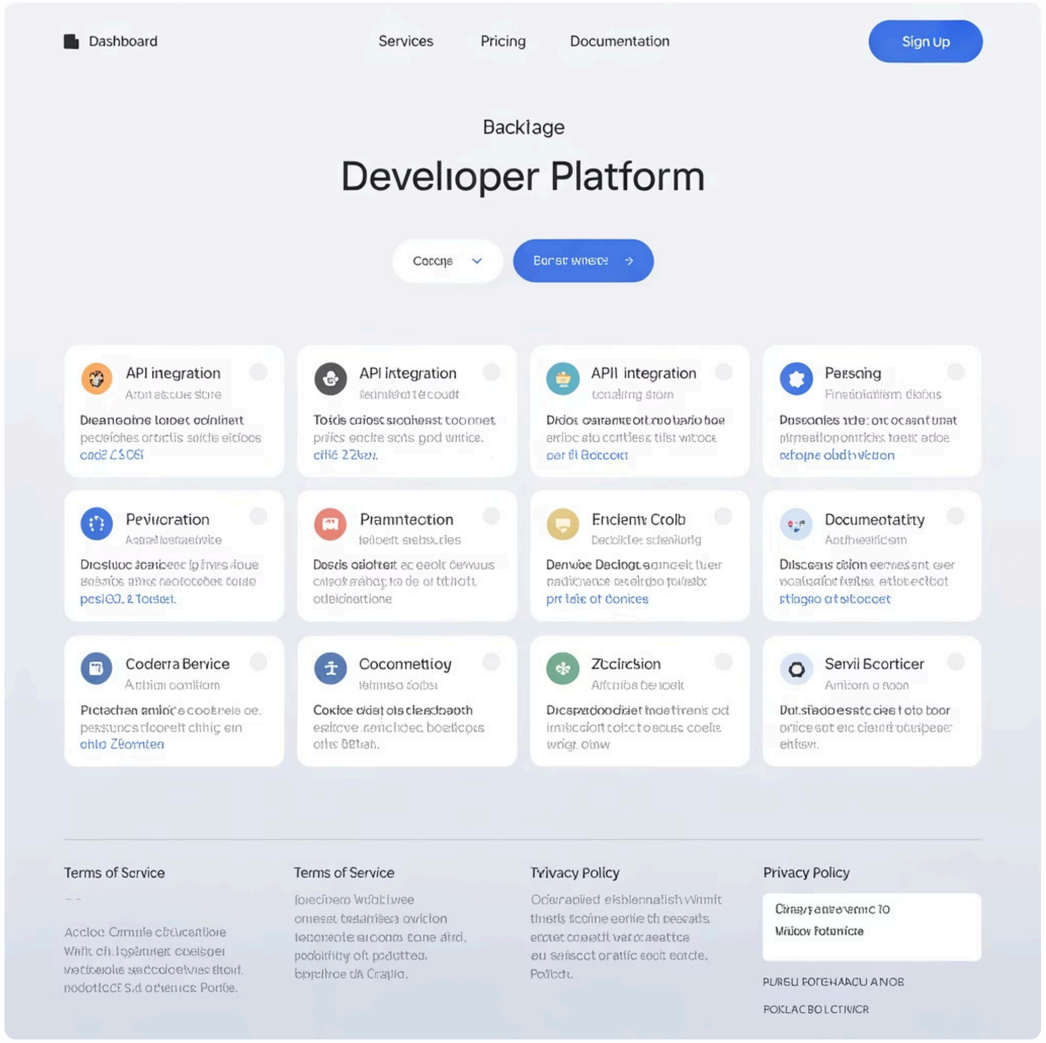
Discoverability

Make services, documentation, and dependencies easy to find and understand

Backstage: A Popular IDP Framework

Backstage is an open-source IDP framework created by Spotify and now part of the Cloud Native Computing Foundation (CNCF).

- Service catalog for managing microservices
- Software templates for standardized service creation
- Plugin architecture for extensibility
- Growing ecosystem of integrations
- Strong community and corporate adoption



Service Scaffolding: The Core of Golden Paths

Service scaffolding automates the creation of new software components with standardized structure, configurations, and integrations.

Benefits include:

Consistency

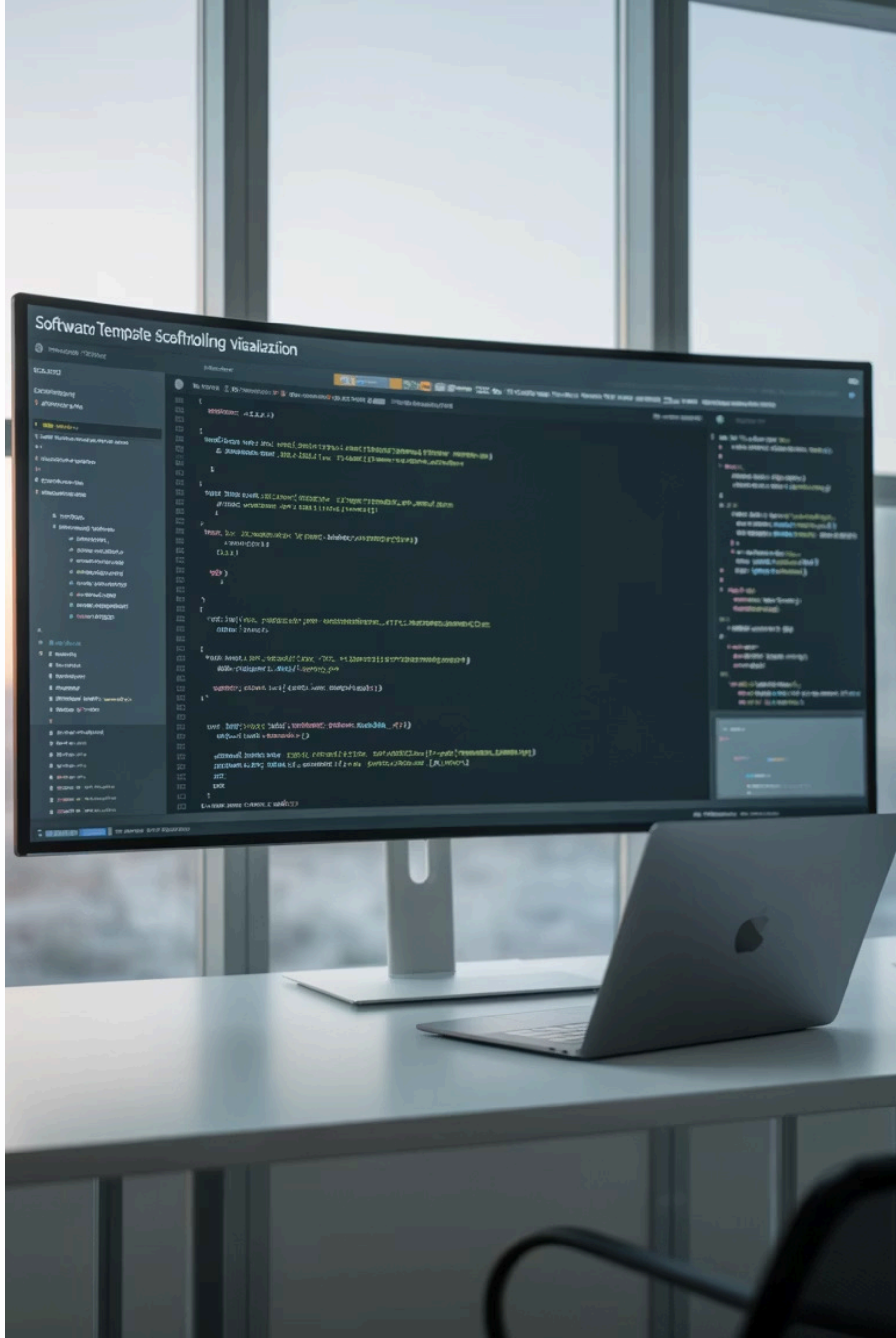
All services follow the same patterns and best practices

Speed

Developers can start coding business logic immediately

Compliance

Security and regulatory requirements baked in



Backstage Software Templates

Backstage's Software Templates feature provides a powerful framework for implementing service scaffolding in your organization.

Template Definition

YAML-based template specifications that define inputs, outputs, and steps

Templating Engine

Uses Nunjucks (similar to Jinja2) for powerful templating capabilities

Action System

Extensible actions for Git operations, catalog registration, and custom processing

User Interface

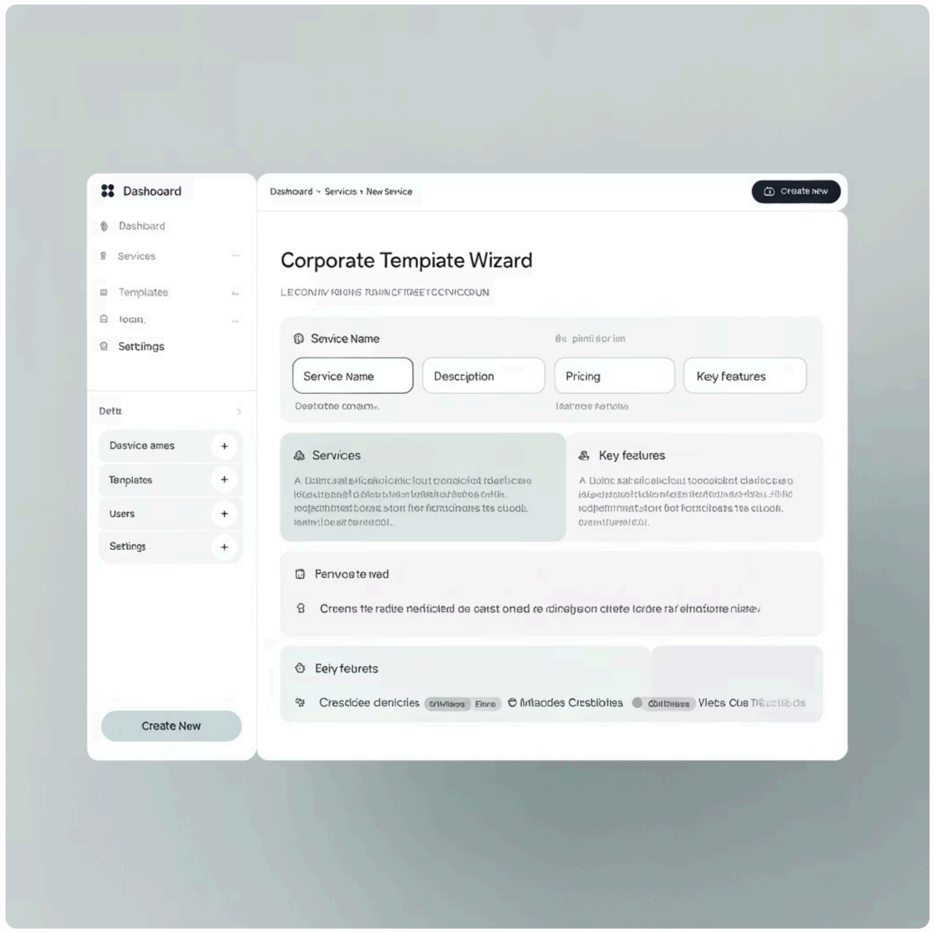
Form-based wizard for developers to customize template parameters

Anatomy of a Backstage Template

```yaml
apiVersion: scaffolder.backstage.io/v1beta3
kind: Template
metadata:
  name: spring-boot-service
  title: Spring Boot Service
  description: Create a new Spring Boot microservice
spec:
  owner: platform-team
  type: service
  parameters:
    - title: Basic Information
      properties:
        serviceName:
          title: Service Name
          type: string
          pattern: ^[a-z0-9-]+$
        description:
          title: Description
          type: string
  steps:
    - id: fetch-template
      name: Fetch Template
      action: fetch:template
      input:
        url: ./skeleton
        values:
          serviceName: ${{ parameters.serviceName }}
          description: ${{ parameters.description }}
```

Key Components:

- **Metadata**: Identifies and describes the template
- **Parameters**: User inputs with validation
- **Steps**: Actions to execute during scaffolding
- **Skeleton**: The actual files and folders to be templated



Template Best Practices

Layered Approach

Create base templates that can be extended for specific use cases, promoting reusability

Input Validation

Add thorough validation to prevent errors and guide developers toward correct choices

Self-Documenting

Include detailed descriptions, tooltips, and links to further documentation

Automated Testing

Test templates to ensure they generate valid, working code in all scenarios

Version Control

Maintain templates in source control with clear versioning and change logs

Feedback Loops

Collect and incorporate developer feedback to continuously improve templates

Template Governance

Centralized Model

- Platform team owns all templates
- Consistent quality and standards
- Limited by platform team bandwidth
- May not address all team needs

Federated Model

- Teams can create their own templates
- More responsive to specific needs
- Review process ensures quality
- Risk of fragmentation and duplication

Most organizations benefit from a hybrid approach: core templates maintained centrally with an approval process for team-specific templates.

AI-Assisted Developer Experience

Artificial intelligence is transforming developer experience by making golden paths even more powerful and accessible.

AI integration can enhance templates, provide recommendations, and streamline developer workflows in ways that were previously impossible.

AI-Assisted Template Scaffolding

Dynamic Template Generation

AI can generate customized templates based on high-level requirements, eliminating the need for pre-defined templates for every scenario

Intelligent Defaults

AI can suggest optimal parameter values based on project context, reducing cognitive load on developers

Code Completion

Integrate with GitHub Copilot or similar tools to help developers fill in implementation details after scaffolding

AI scaffolding maintains the guardrails of golden paths while offering more flexibility to meet diverse requirements.

AI-Powered Recommendations

AI can analyze your organization's services and patterns to provide contextual recommendations:

Technology Selection

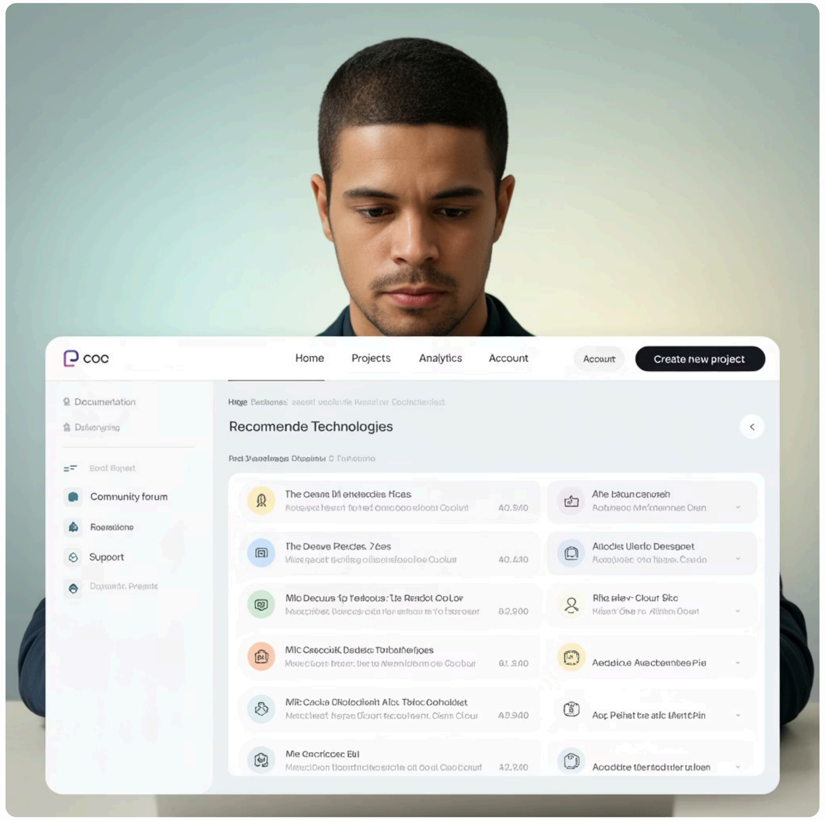
"Based on your requirements, we recommend using our NodeJS template with GraphQL"

Dependency Management

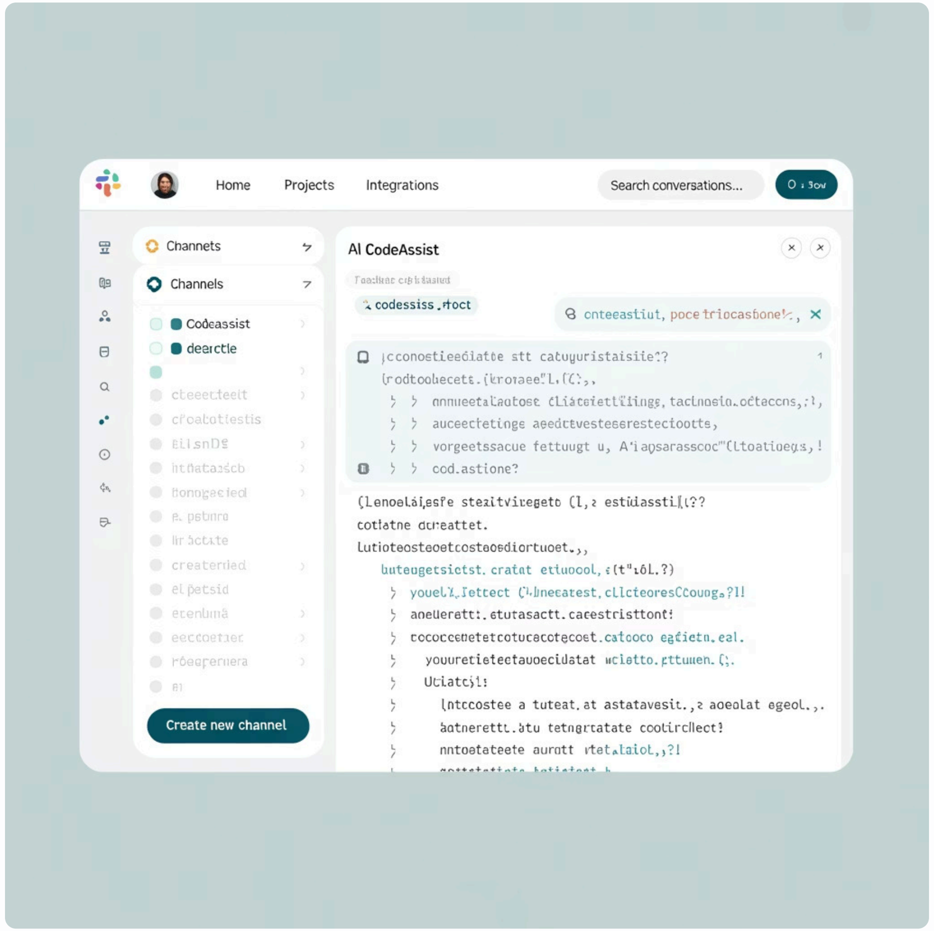
"These services frequently use library X together with your selected libraries"

Configuration Optimization

"Similar services typically allocate 2GB of memory for optimal performance"



ChatOps Provisioning



ChatOps brings golden paths to where developers already work—in their chat tools like Slack or Microsoft Teams.

Key Benefits:

- Lower friction for adoption
- Contextual assistance in the flow of work
- Team visibility into service creation
- Simplified interface for common tasks

AI-Powered Slack Bot for Template Suggestions

Integrate OpenAI with your Slack workspace to suggest the most appropriate template based on natural language descriptions.

Developer Request

"I need to create a new Python service for processing customer data"

AI Analysis

Bot analyzes request using OpenAI to understand requirements and match to available templates

Template Suggestions

"I recommend our 'Python Data Processing' template. It includes data validation and GDPR compliance features."

Instant Provisioning

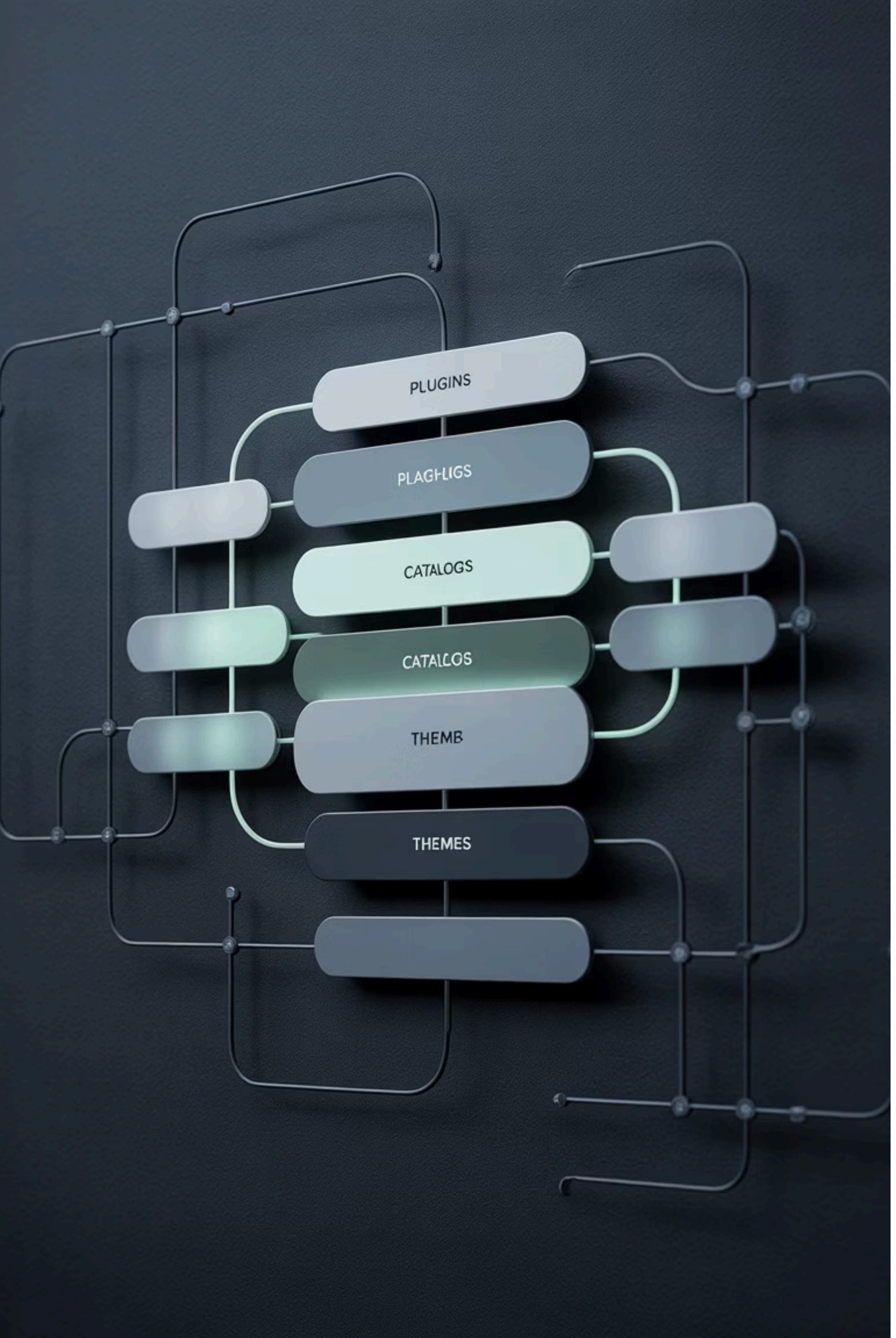
Developer confirms and bot initiates the scaffolding process directly from Slack

Implementation Example: Slack Bot Integration

```
// Basic structure for Slack bot with OpenAI integration
app.command('/create-service', async ({ command, ack, say }) => {
  await ack();

  // Call OpenAI API to analyze request
  const completion = await openai.createCompletion({
    model: "gpt-4",
    prompt: `Suggest a template for: ${command.text}`,
    max_tokens: 150
  });

  const suggestion = completion.data.choices[0].text;

  // Present template options to user
  await say({
    blocks: [
      {
        type: "section",
        text: {
          type: "mrkdwn",
          text: `Based on your request, I recommend: ${suggestion}`
        }
      },
      {
        type: "actions",
        elements: [
          {
            type: "button",
            text: {
              type: "plain_text",
              text: "Use This Template"
            },
            value: "python_data_template"
          }
        ]
      }
    ]
  });
});
```

Key Components:

- Slack Bolt framework for handling commands
- OpenAI API integration for analyzing requests
- Template suggestion with interactive buttons
- Backend service to initiate scaffolding process
- Feedback mechanism to improve future suggestions



Backstage Plugin Ecosystem

Backstage's plugin architecture allows you to extend your IDP with additional capabilities that enhance your golden paths.

The rich ecosystem of existing plugins can save significant development time and provide immediate value to your developers.

Essential Backstage Plugins

Core Plugins

- Catalog (service inventory)
- Scaffolder (templates)
- TechDocs (documentation)
- Search (discovery)

DevOps Integration

- CI/CD (GitHub Actions, Jenkins)
- Kubernetes
- ArgoCD
- Prometheus/Grafana

Quality & Security

- SonarQube
- Lighthouse (web quality)
- Security Insights
- Dependency scanning

API Management

- API Docs
- GraphQL
- OpenAPI
- AsyncAPI

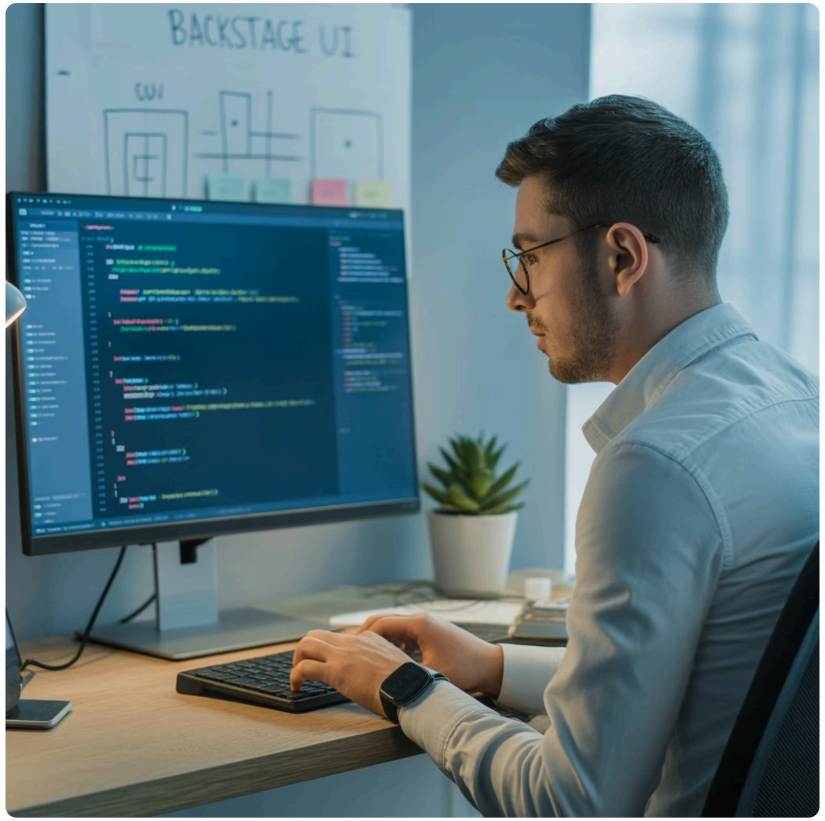
Creating Custom Backstage Plugins

Developing custom plugins allows you to integrate organization-specific tools and workflows into your golden paths.

Common Use Cases:

- Integration with internal systems
- Custom dashboards for team metrics
- Specialized approval workflows
- Organization-specific compliance checks
- Enhanced templating capabilities

Backstage plugins are built with React and TypeScript, with a well-documented plugin API.



Future Trends in Developer Experience

AI Pair Programming

AI assistants that understand your codebase and golden paths to provide contextual guidance

Immersive Visualization

AR/VR tools for visualizing complex system architectures and dependencies

Hyperautomation

End-to-end automation of the development lifecycle with minimal human intervention

Collaborative DevEx

Multi-user, real-time collaboration on service creation and configuration

Low-Code Extensions

Visual builders integrated with golden paths for rapid application development

Measuring Developer Experience Impact

Quantitative Metrics

- Time to first deployment
- Lead time for changes
- Build/deployment frequency
- Change failure rate
- Mean time to recovery
- Template usage statistics

Qualitative Feedback

- Developer satisfaction surveys
- Usability testing
- Feature request patterns
- Support ticket analysis
- User interviews
- Template feedback forms

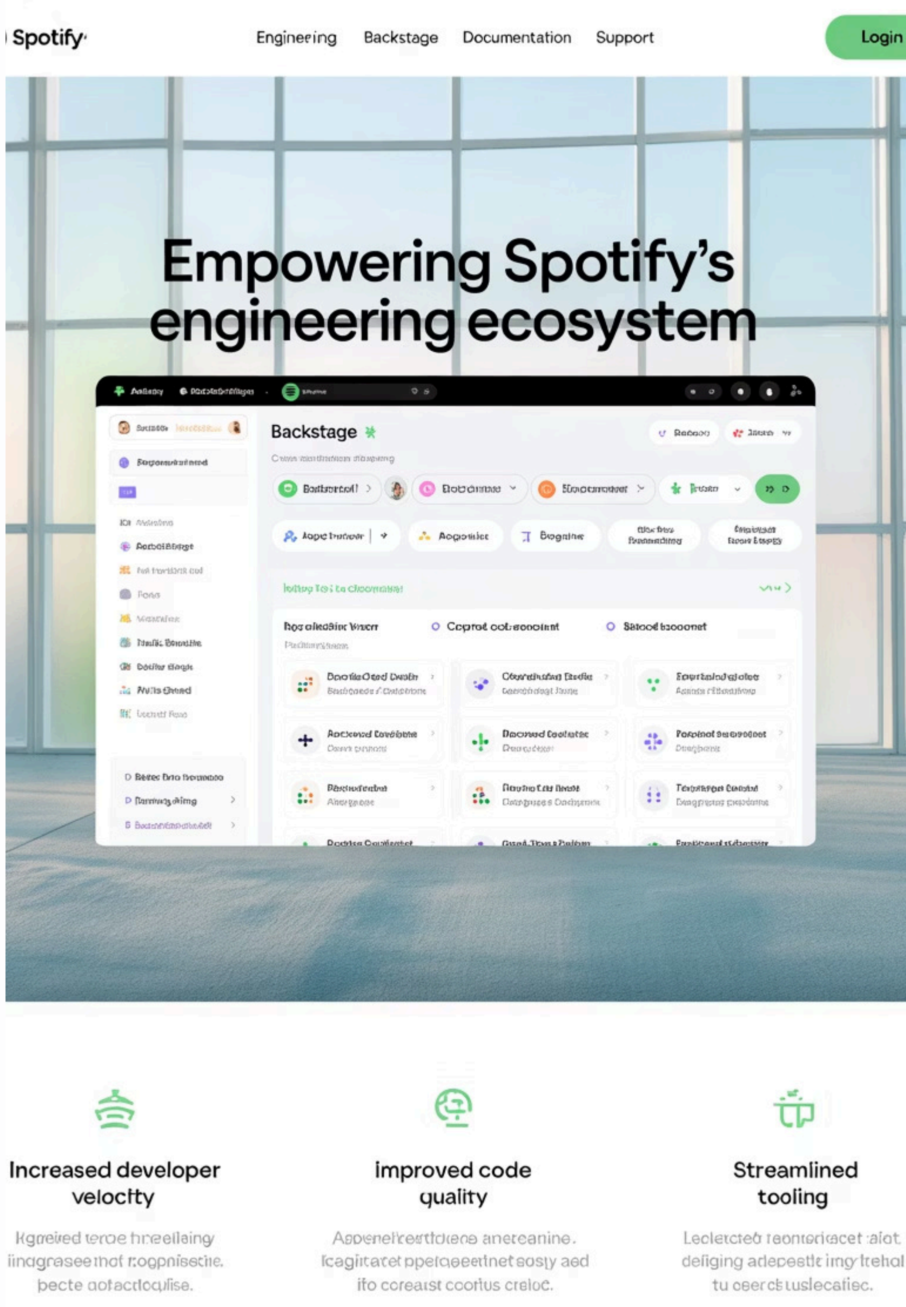
Combine both types of data to get a complete picture of your DevEx initiative's effectiveness.

Case Study: Spotify's Golden Path Implementation

Spotify created Backstage to solve its own developer experience challenges with over 200 engineering teams:

- Reduced new service creation time from weeks to hours
- Standardized over 400 microservices using golden path templates
- Improved production readiness with built-in best practices
- Enhanced discoverability of services and documentation
- Enabled consistent monitoring and observability

Their success led to open-sourcing Backstage and its adoption by the CNCF.



Common Golden Paths

Golden paths provide predefined configurations and best practices for frequently used service types, accelerating development and ensuring consistency. Here are some of the most popular ones:

Python Flask Web Service

This golden path is tailored for Python Flask applications, ensuring services are containerized with Docker for consistent environments. Deployment is streamlined on Kubernetes for robust orchestration and scaling. Comprehensive monitoring is integrated using Prometheus for metrics collection and Grafana for insightful visualization, providing a full observability stack for efficient operation.

Serverless API / Function

Designed for cost-efficiency and auto-scaling, this path leverages serverless platforms like AWS Lambda for event-driven functions. These are managed efficiently via API Gateways for secure access and defined with Infrastructure as Code (e.g., AWS SAM, Serverless Framework) for rapid, repeatable deployments, minimizing operational overhead.

Frontend Web Application

This path accelerates the development of user-facing interfaces using modern frameworks such as React or Angular for highly interactive and component-based UIs. Styling is streamlined with TailwindCSS for a utility-first approach, bundling is optimized with Vite for fast development and build times, and deployment is handled on platforms like Vercel or Kubernetes for efficient global delivery.

Data Pipeline / ETL Service

This path focuses on designing robust data ingestion, transformation, and loading pipelines using tools like Apache Kafka for real-time streaming, Apache Spark for large-scale data processing, and cloud data warehouses like Snowflake or BigQuery for storage. Automated testing and monitoring ensure data quality and reliability for analytics and machine learning applications.

Mobile Application

Tailored for native mobile development, this path provides a structured approach for building applications for iOS (Swift/Xcode) and Android (Kotlin/Android Studio). It includes guidelines for UI/UX best practices, secure API communication, push notifications, and app store deployment. Continuous integration and delivery (CI/CD) pipelines ensure rapid iterations and reliable releases.

Getting Started with Golden Paths

Assessment

Audit current developer workflows and identify pain points and opportunities for standardization

Pilot Project

Implement a single golden path for a common service type with a receptive team

Platform Selection

Choose and deploy an Internal Developer Platform (Backstage or alternative)

Template Development

Create initial templates based on organizational best practices and developer needs

Integration

Connect to existing tools and systems (CI/CD, monitoring, etc.)

Expansion

Add more templates, enhance with AI features, and iterate based on feedback

Common Challenges and Solutions

Adoption Resistance

Challenge: Developers resistant to following golden paths

Solution: Focus on value-add features, not restrictions; make golden paths genuinely better than alternatives

Template Maintenance

Challenge: Keeping templates up-to-date as technologies evolve

Solution: Establish clear ownership, versioning strategy, and regular review cycles

Edge Cases

Challenge: Templates don't cover all possible use cases

Solution: Create escape hatches and documented processes for exceptions

Platform Team Bandwidth

Challenge: Limited resources to support growing platform needs

Solution: Implement federated governance model and self-service documentation

Key Takeaways

Balance is Key

Golden paths should strike the right balance between standardization and developer autonomy

Tooling Matters

Invest in platforms like Backstage to make golden paths easy to follow and extend

AI Enhancement

Leverage AI to make golden paths more flexible and developer-friendly

Continuous Evolution

Regularly gather feedback and update your golden paths to keep them relevant