THE FUTURE OF SOFTWARE DEVELOPMENT

NAVIGATING INTERNAL DEVELOPER PLATFORMS

WWW.RIVERSAFE.CO.UK

RIVERSAFE

CONTENTS

Section 1 - Introduction **3**

Section 2 - Why IDPs matter **4**

Section 3 - Main features of an IDP **5**

Section 4 - The role of different teams **7**
Platform Team 7
Architects 7
Developers 8

Section 5 - Building or Extending an IDP **9**
Building an IDP from Scratch 9
Extending an Existing Platform 9

Section 6 - Case Studies **11**

Section 7 - High level IDP Architecture **13**

Section 8 - Challenges in implementing an IDP **14**

Section 9 - How RiverSafe can help **15**

Section 10 - Conclusion **16**

1. INTRODUCTION

Internal Developer Platforms (IDPs) are quickly becoming the norm when it comes to modern software delivery. But here's the thing, many organisations are already running their own version of an IDP, even if they don't necessarily call it that. Whether it's a self-service portal, an automated pipeline, or a collection of scripts and tools put together by your engineering team, these are all constituent parts of what could be considered an IDP.

At their core, IDPs are about creating a developer friendly system that abstracts away complexity while enabling autonomy. Instead of developers spending hours figuring out how to deploy code, configure environments, or navigate compliance requirements, they can focus on building features and delivering value to customers.

For organisations, this means more than just happy developers. A well designed IDP improves speed, consistency, and reliability across the board. It's the glue that holds together modern practices like DevOps, Platform Engineering, and Continuous Delivery, while ensuring they scale as your teams and systems grow.

At RiverSafe, we've seen firsthand how the right approach to IDPs can transform engineering teams and unlock their full potential. This report explores what makes an IDP truly effective, the critical roles involved in building and maintaining one, and the choices organisations have for creating or extending their own platform.

2. WHY IDPS MATTER

I think we can all agree the complexity of modern software development has grown significantly. With the rise of cloud-native applications, microservices, and multi-environment deployments, teams are juggling a vast array of tools, processes, and technologies as highlighted in the recent 2024 State of Developer Experience Report. This complexity often results in inefficiencies, bottlenecks, and friction between developers and operations teams. Internal Developer Platforms (IDPs) are designed to address these challenges head-on, providing a unified solution that drives productivity and alignment across teams.

HERE ARE THE KEY REASONS WHY IDPS ARE SO IMPORTANT:

Reducing cognitive load for developers
Developers shouldn't need to become experts in infrastructure or deployment pipelines to deliver great software. IDPs abstract away the complexity of managing environments, provisioning resources, and configuring tools, allowing developers to focus on writing code. This not only boosts productivity but also improves job satisfaction, as developers can spend more time on meaningful work.

Accelerating Software Delivery
Manual handoffs between teams and inconsistent processes can significantly slow down delivery timelines. IDPs enable self-service workflows, where developers can provision environments, deploy applications, or run tests without waiting on another team. This speed and autonomy lead to faster releases and quicker iteration cycles

Enforcing Governance and Compliance
As organisations scale, ensuring that every team adheres to security, compliance, and operational best practices can become a daunting challenge. IDPs embed these requirements into the platform itself, ensuring that deployments follow organisational policies by default. This minimises risks and helps organisations maintain compliance without adding extra overhead to development teams.

Breaking Down Silos
Traditional development models often create silos between developers, operations, and other teams. An IDP serves as a common ground where all teams can collaborate effectively. By standardising tools and workflows, it promotes alignment and reduces miscommunication.

Driving Consistency Across Teams
When each team operates with its own set of tools and processes, it leads to inefficiency and duplication of effort. IDPs provide a standardised framework for software delivery, ensuring consistency across teams while still allowing flexibility where needed.

Empowering Innovation
By eliminating repetitive tasks and manual processes, IDPs free up developers to focus on what they do best: innovating and solving customer problems. This creates a competitive advantage by allowing organisations to deliver value to their customers faster and more reliably.

IDPs matter because they align the goals of developers, operations, and business stakeholders. They're not just about improving developer experience - they're about enabling organisations to deliver software with speed, safety, and scalability. Whether you're looking to reduce costs, accelerate delivery, or improve collaboration, an IDP is a powerful tool to help achieve those goals.

3. MAIN FEATURES OF AN IDP

A well-designed Internal Developer Platform (IDP) is more than a collection of tools - it's a way to make life easier for developers while keeping things efficient and organised for platform teams. A great IDP gives developers the freedom to get things done without roadblocks, while also ensuring that standards and best practices are followed. Here's what makes an IDP work well:

FEATURE	WHAT IT DOES	WHY IT MATTERS	EXAMPLE
Self-Service Capabilities	Enables developers to perform tasks like provisioning environments and deploying apps without relying on operations teams.	Reduces bottlenecks and empowers teams to move faster while adhering to best practices.	A developer can spin up a staging environment with a single click through the IDP's portal.
Workflow Automation	Automates repetitive processes like CI/CD pipelines, infrastructure provisioning, and monitoring.	Saves time, reduces errors, and ensures consistency across deployments.	Pipelines automatically run security scans and deploy only when all tests have passed.
Tool & Process Standardisation	Provides a unified way to interact with tools like Kubernetes, Terraform, and Jenkins.	Reduces complexity for developers and ensures interoperability between teams and systems.	A single command or API call to deploy to any environment, regardless of the infrastructure.
Integrated Observability	Embeds logging, metrics, and tracing into the platform for visibility into application performance.	Helps teams identify and resolve issues quickly, improving reliability.	Developers can access pre-configured dashboards to monitor apps without manually setting up tools.
Governance & Compliance	Embeds security, compliance, and operational policies into workflows automatically.	Reduces risk and ensures consistency across deployments without slowing down development.	Enforcing encryption and access control policies in infrastructure provisioning.

FEATURE	WHAT IT DOES	WHY IT MATTERS	EXAMPLE
Customisability	Allows teams to tailor the platform to specific work-flows, tools, and needs.	Balances standardisation with flexibility, ensuring the platform supports diverse team requirements.	Custom templates for provisioning environments that meet team-specific needs.
Developer-Centric Design	Prioritises ease of use with intuitive interfaces, clear documentation, and support.	Encourages adoption and makes the platform an enabler rather than a blocker.	An easy-to-navigate web UI abstracts complexity while offering advanced options for power users.
Integration with Tools	Seamlessly connects with cloud platforms, CI/CD tools, and version control systems.	Ensures teams leverage existing investments while enhancing capabilities.	The IDP integrates with GitOps workflows, triggering deployments automatically when changes are merged.
Developer Onboarding	Simplifies onboarding with pre-configured environments, clear guidelines, and workflows.	Reduces the time for new developers to b ecome productive.	A new developer can start deploying to staging on their first day using pre-defined pipelines and templates.

4. THE ROLE OF DIFFERENT TEAMS

A successful IDP doesn't exist in isolation - it relies on collaboration between several key teams. Each group plays a distinct role in shaping, building, and maintaining the platform, ensuring it meets the needs of the organisation while delivering maximum value. Here's how the responsibilities break down:

PLATFORM TEAMS

Primary Role: The builders and maintainers of the IDP.

Responsibilities:

- Design, build, and evolve the platform to align with the organisation's needs.
- Integrate tools, automate workflows, and ensure the platform is scalable and reliable.
- Act as the bridge between infrastructure and development, translating operational complexity into simple, developer-friendly interfaces.

Key Objective: Enable developers to focus on coding while ensuring compliance, scalability, and operational excellence.

Real-World Example: A platform team might create a self-service tool that allows developers to deploy applications securely without needing to understand the underlying Kubernetes setup.

ARCHITECTS

Primary Role: The strategists who ensure the IDP aligns with the organisation's broader technical and business goals.

Responsibilities:

- Define the technical guidelines and principles for the platform.
- Ensure the platform integrates seamlessly into the organisation's overall architecture.
- Balance innovation with long-term maintainability and scalability.

Key Objective: Ensure the IDP supports the strategic direction of the business and technical landscape.

Real-World Example: An architect might design how the IDP integrates with a multi-cloud strategy, ensuring flexibility and future proofing.

DEVELOPERS

Primary Role: The primary users of the IDP.

Responsibilities:

- Use the platform to deploy code, manage environments, and perform day-to-day tasks.
- Provide feedback to the platform team to ensure it meets real-world needs.
- Follow the workflows and standards embedded in the platform to ensure consistency.

Key Objective: Deliver value to customers faster and more efficiently, with fewer distractions from infrastructure or operational tasks.

Real-World Example: A developer might use a pre-configured pipeline from the IDP to deploy a new feature to production with minimal manual effort.

For an IDP to succeed, these teams must work together:

- Platform Teams ensure the IDP is functional and developer friendly.
- Architects ensure it aligns with the bigger picture.
- Developers ensure it's used effectively and evolves to meet practical needs.

When these groups collaborate effectively, the IDP becomes a powerful enabler for the organisation, driving efficiency, consistency, and innovation across the board.

5. BUILDING OR EXTENDING AN IDP

Organisations at different stages of their platform journey often face a common question: should we build an IDP from scratch or extend what we already have? The answer depends on factors like existing infrastructure, team expertise, and long-term goals. Here's a look at both approaches to help you decide which might be right for your organisation.

BUILDING AN IDP FROM SCRATCH

Building an IDP from scratch allows you to design a platform tailored to your organisation's unique needs. However, it requires significant effort, expertise, and time.

When to Consider This Approach:

- You're starting fresh and don't have existing platform components to build upon.
- Your organisation has specific requirements that off-the-shelf tools or extensions can't meet.
- You have a skilled platform engineering team capable of handling the complexity of building and maintaining a custom solution.

Steps to Build:

- **Understand Developer Needs:** Interview developers to identify pain points and workflows that need improvement.
- **Select Core Technologies:** Choose foundational tools like Kubernetes, Terraform, or CI/CD systems.
- **Focus on Self-Service:** Create workflows and automation that enable developers to work independently.
- **Iterate and Improve:** Start small, test with pilot teams, and refine the platform based on feedback.

Challenges:

- Significant upfront investment in time and resources.
- Risk of over-engineering or building features that don't get adopted.
- Long-term maintenance and scalability considerations.

EXTENDING AN EXISTING PLATFORM

Most organisations already have some form of platform in place, even if it's not formalised as an IDP. Extending these existing tools and workflows can be a faster, more cost-effective way to deliver value.

When to Consider This Approach:

- You have existing tools or systems that work well but need better integration or user experience.
- Your goal is to deliver incremental improvements rather than an entirely new solution.
- Resources or expertise to build from scratch are limited.

Steps to Extend:

- **Audit Your Existing Setup:** Identify the tools, scripts, and workflows currently in use and their pain points.
- **Prioritise Integrations:** Focus on connecting systems to create a more seamless experience.
- **Introduce Standardisation**: Ensure consistent practices across teams by embedding them into the platform.
- **Add Self-Service Layers:** Simplify processes like environment provisioning or deployment with user-friendly interfaces.

Advantages:

- Faster time to value since you're building on what already works.
- Lower upfront costs compared to building from scratch.
- Less risk as it focuses on enhancing existing processes rather than reinventing them.

Challenges:

- Potential limitations of legacy tools or systems.
- Complexity in integrating disparate tools and workflows.
- Risk of accumulating technical debt if extensions are not thoughtfully designed.

MAKING THE CHOICE

- **Build from Scratch** if you need a tailored solution, have the resources, and are prepared for a longer implementation timeline.
- **Extend Existing Platforms** if you need faster results, want to optimise existing investments, and have a clear understanding of the gaps to address.

Both approaches benefit from an iterative mindset. Whether building or extending, start small, involve your developers early, and continually refine the platform based on feedback. At RiverSafe, we've guided organisations through both paths, helping them achieve efficient, scalable platforms tailored to their unique needs.

6. CASE STUDIES

At RiverSafe, we partnered with a large oil and gas client to design and implement an Internal Developer Platform (IDP) from scratch, tailored to simplify application deployment and promote best practices across their organisation. The goal was to create a seamless process for managing resources, deploying applications, and ensuring governance in multi-environment setups.

THE CHALLENGE

The client needed a unified platform that could:

- Automate application deployments across multiple environments (development, staging, production).
- Standardise infrastructure and application management while maintaining flexibility for team-specific workflows.
- Incorporate robust secrets management, approval workflows, and rollback mechanisms.
- Enable visibility into deployment progress and resource utilisation.

THE SOLUTION

The solution leveraged several cutting-edge tools and technologies integrated into Kubernetes clusters to provide a comprehensive platform.

Key components included:

- **Resource Management:** A control plane was implemented to manage cloud resources natively through Kubernetes using a declarative approach.
- **Secrets Synchronisation:** A synchronisation layer ensured that secrets from an external secrets store were securely injected into application pods as environment variables.
- **Certificate Management:** Automated lifecycle management for TLS certificates, ensuring secure application exposure via ingress.
- **Observability and Monitoring**: A centralised monitoring system collected and visualised application metrics, including traces and logs.
- **Policy Enforcement:** A Kubernetes-native policy engine enforced best practices, access controls, and dynamic pod configurations.
- **Dynamic Application Deployment:** Templates and configurations were dynamically rendered based on environment-specific and application-specific requirements, ensuring consistency while allowing for custom overrides.

KEY FEATURES DELIVERED:

Automated Deployment Pipelines:

- Applications were promoted through environments with optional approval gates for production.
- Continuous deployment was the default, with fallback options for approval-based or manual workflows.

Centralised Secrets Management:

- Secrets were structured hierarchically for project, environment, and application scopes.
- Policies ensured secrets were securely injected into pods without manual intervention.

Infrastructure as Code:

- All cloud resources, such as storage buckets and IAM roles, were managed declaratively, reducing the need for manual configuration.

Observability at Scale:

- A centralised observability stack provided detailed metrics and tracing capabilities for applications deployed on the platform.

Dynamic Rollback Capabilities:

- Rollbacks could be triggered automatically or manually, with detailed tracking of deployment and rollback states.

TECHNOLOGIES USED

- **Kubernetes-native tools:** Used for provisioning and managing cloud resources directly from Kubernetes, leveraging technologies like Crossplane to declaratively manage infrastructure components such as S3 buckets, IAM roles, and RDS instances.
- **GitOps workflows:** Implemented using tools like ArgoCD to automate infrastructure and application deployments, ensuring consistent and version-controlled updates across environments.
- **Helm charts:** Standardised application configurations with Helm, enabling teams to template and layer environment-specific customisations for streamlined deployments.
- **Integrated policy engines:** Kyverno was used to enforce compliance and governance, dynamically injecting configurations, secrets, and labels into Kubernetes resources to ensure best practices were followed.
- **Centralised logging and tracing systems:** OpenTelemetry Collector gathered logs, traces, and metrics from running applications, which were then visualised in Grafana to provide insights into performance and troubleshooting.

THE OUTCOME

- **The platform empowered the client to:**
- Accelerate application delivery through automated pipelines.
- Reduce human error by embedding governance and standardisation into the platform.
- Provide developers with a self-service interface for deploying and managing their applications.
- Enhance visibility and control across environments, fostering collaboration between development and operations teams.

7. HIGH LEVEL IDP ARCHITECTURE

This illustration shows the high level architecture for the IDP mentioned above, which seamlessly integrates Kubernetes, cloud services, and SDLC tools, accelerating provisioning and reducing cognitive load for developers. By automating pipelines, offering out-of-the-box observability, and centralising IAM, our IDP handles the heavy lifting so teams can focus on building features.

DEPLOYMENT FLOW

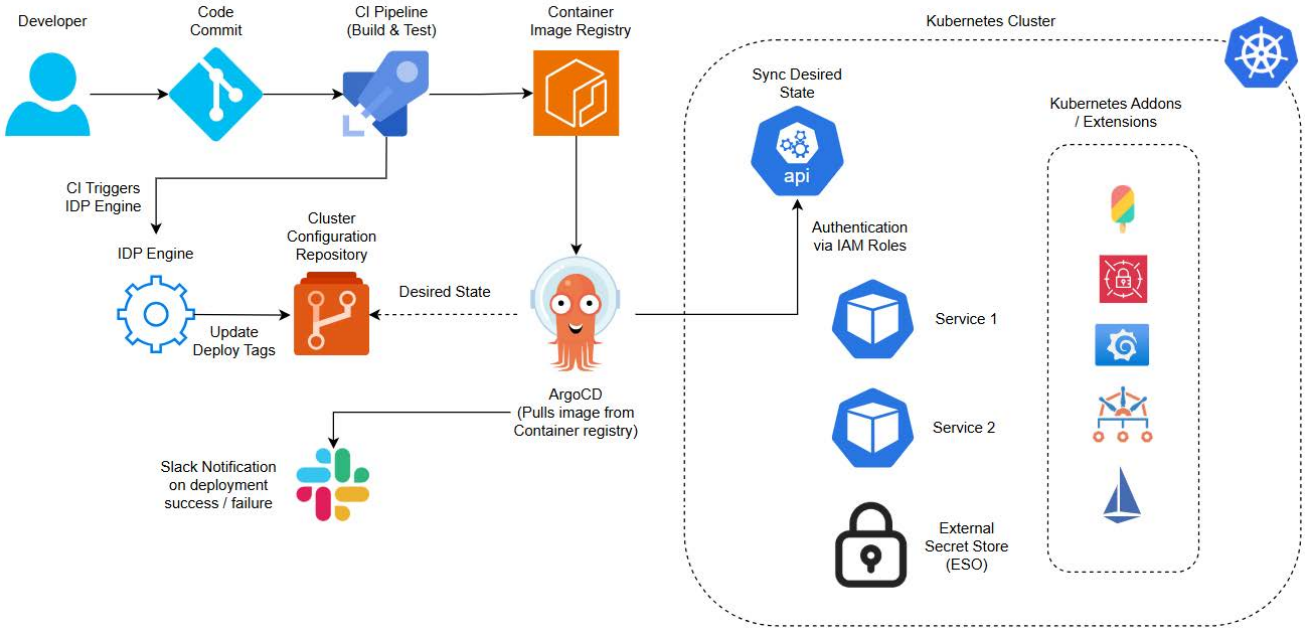
- When developers commit new code, a CI pipeline builds and tests the application, the resulting container image is pushed to a central registry. This automatically triggers our custom IDP Engine, which updates the deployment tag in the configuration repository where service configurations live. Argo CD continuously monitors that repository, detects the new tag and deploys the service to development. Once Development checks pass, the IDP Engine promotes the same image to production, ensuring each environment is validated before moving forward. Throughout this process, Argo CD sends real-time notifications to the relevant teams about deployment progress and status.

IDP ENGINE

- The IDP Engine is the orchestrator that sits between the CI pipeline and Argo CD. It automatically updates deployment tags, enforces environment sequencing, and confirms each deployment's health before proceeding. By handling these tasks behind the scenes, the Engine eliminates manual procedures, and keeps stakeholders in the loop with timely notifications.

ADDONS

- The IDP supports a broad range of addons such as Grafana dashboards, Kyverno for policy enforcement, and External Secrets Operator for secure credential management to standardise best practices and reduce setup overhead. One key addon is Crossplane, which is the backbone of our Infrastructure Provisioning Catalogue. This solution uses pre-defined composite resource definitions (XRDs) to centralize the provisioning and management of cloud services. It automatically attaches the correct IAM policies and injects connection details as secrets, ensuring out-of-the-box security and compliance. By reusing these definitions across teams, we standardise the process of spinning up new infrastructure, making it faster, safer, and more consistent for everyone.



8. CHALLENGES IN IMPLEMENTING AN IDP

Building or extending an Internal Developer Platform (IDP) can transform the way teams work, but it's not without its challenges. Organisations often encounter technical, organisational, and cultural hurdles when implementing an IDP. Understanding these challenges upfront can help you navigate them effectively and set your platform up for success.

BALANCING STANDARDISATION WITH FLEXIBILITY

- **The Challenge:** Different teams often have varying workflows, tools, and preferences. Striking a balance between standardising processes across the organisation and allowing individual teams the flexibility to operate as needed can be difficult.
- **How to Address It:** Start with a baseline set of workflows and tools that solve common problems. Allow teams to extend or customise where necessary but maintain governance to ensure compliance.

MANAGING COMPLEXITY

- **The Challenge:** IDPs often integrate multiple tools, including CI/CD systems, infrastructure as code, and observability platforms. Coordinating these tools in a seamless, developer-friendly way can become overwhelming.
- **How to Address It:** Invest in a clear architecture and modular design. Begin with a minimum viable platform (MVP) and expand gradually. Avoid over-engineering by focusing on high-impact features first.

DRIVING ADOPTION

- **The Challenge:** Developers may resist adopting a new platform, especially if it feels like a top-down initiative or adds friction to their workflow.
- **How to Address It:** Engage developers early in the design process. Seek their feedback, incorporate their needs, and demonstrate how the IDP solves their pain points.
- Make onboarding intuitive and provide comprehensive documentation and training.

MAINTAINING ALIGNMENT BETWEEN TEAMS

- **The Challenge:** Building an IDP requires collaboration between platform teams, architects, and developers. Misalignment on goals or expectations can slow progress.
- **How to Address It:** Establish clear communication channels and shared objectives. Regularly involve stakeholders from all relevant teams to ensure alignment on priorities and direction.

MANAGING TECHNICAL DEBT

- **The Challenge:** Over time, an IDP can accumulate technical debt due to rapid iteration, custom integrations, or lack of documentation.
- **How to Address It:** Regularly review and refactor components of the platform. Set aside time and resources to address technical debt and keep the platform maintainable.

MEASURING SUCCESS

- **The Challenge:** Demonstrating the return on investment (ROI) of an IDP can be challenging, especially in the early stages.
- **How to Address It:** Define clear success metrics, such as reduced lead times, increased deployment frequency, or improved developer satisfaction. Regularly track and share progress to highlight the platform's value.

9. HOW RIVERSAFE CAN HELP

Implementing an IDP can be transformative, but navigating its complexities requires expertise. RiverSafe offers comprehensive services to guide your organisation through this journey, ensuring a tailored solution that aligns with your business objectives.

CUSTOMISED IDP DEVELOPMENT

RiverSafe can create a bespoke IDP that eliminate bottlenecks and provide developers with streamlined, self-service access to necessary tools and environments. By removing infrastructure complexities and standardising development workflows, we enable your teams to focus on delivering innovative solutions.

SECURITY-FIRST DESIGN

With deep expertise in cybersecurity and application security, RiverSafe ensures that security best practices are embedded into your platform from the outset, not added as an afterthought. This proactive approach safeguards your operations while maintaining development agility.

PROVEN EXPERIENCE

Our extensive experience in enhancing developer experiences has helped organisations remove barriers and accelerate developer velocity, transforming software delivery processes. For instance, we assisted a leading energy supplier in saving £4 million annually by standardising DevOps tools, streamlining processes, and improving visibility.

COMPREHENSIVE DEVOPS SERVICES

Beyond IDP development, RiverSafe offers a suite of DevOps services, including strategy consulting, knowledge management, and maturity assessments. Our on-demand engineering support connects you with experts in Infrastructure as Code, CI/CD, and Cloud Services, seamlessly integrating talent into your team for scalable solutions that drive results.

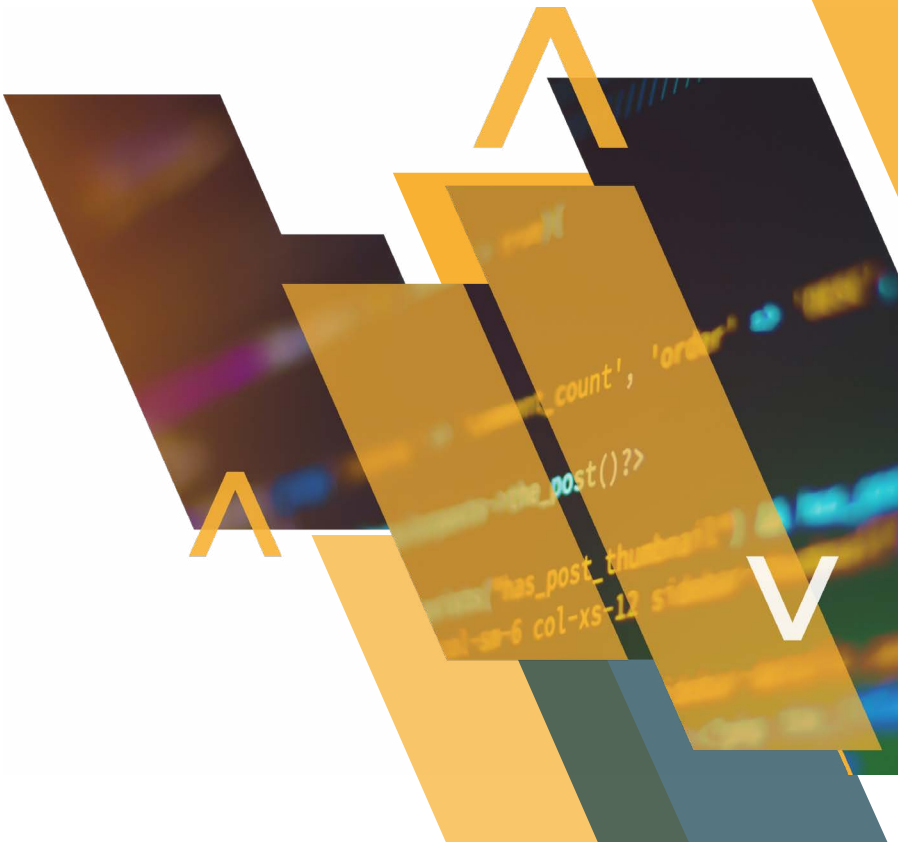
Partnering with RiverSafe means leveraging our deep knowledge of IDPs and platform engineering to unlock greater value from your software development lifecycle. We work alongside your teams to create self-service platforms that transform the developer experience, reduce cognitive load, and maintain security and standardisation across your organisation.

10. CONCLUSION

IDPs are transforming the way engineering teams work, enabling faster delivery, greater consistency, and improved developer satisfaction. By providing a centralised, self-service platform, IDPs empower developers to focus on creating value while ensuring governance and operational excellence across the organisation.

However, as discussed earlier on, implementing an IDP is not without its challenges. It requires thoughtful planning, cross-team collaboration, and a deep understanding of both technical and organisational needs. Whether you're starting from scratch or extending an existing platform, the journey to a successful IDP is as much about aligning people and processes as it is about technology.

At RiverSafe, we understand the intricacies of designing and building IDPs that work. From addressing organisational challenges to embedding best practices, we partner with organisations to unlock the full potential of their development teams. Our expertise in platform engineering, combined with our commitment to delivering tailored solutions, ensures that your IDP becomes a strategic enabler for your business.



READY TO EMPOWER YOUR TEAMS AND ACCELERATE YOUR SOFTWARE DELIVERY?

Let RiverSafe help you build a platform that drives innovation, reduces complexity, and delivers measurable value.

Get in touch with us today to learn how we can help transform your developer experience.

www.riversafe.co.uk

Enquiries@RiverSafe.co.uk

RIVERSAFE