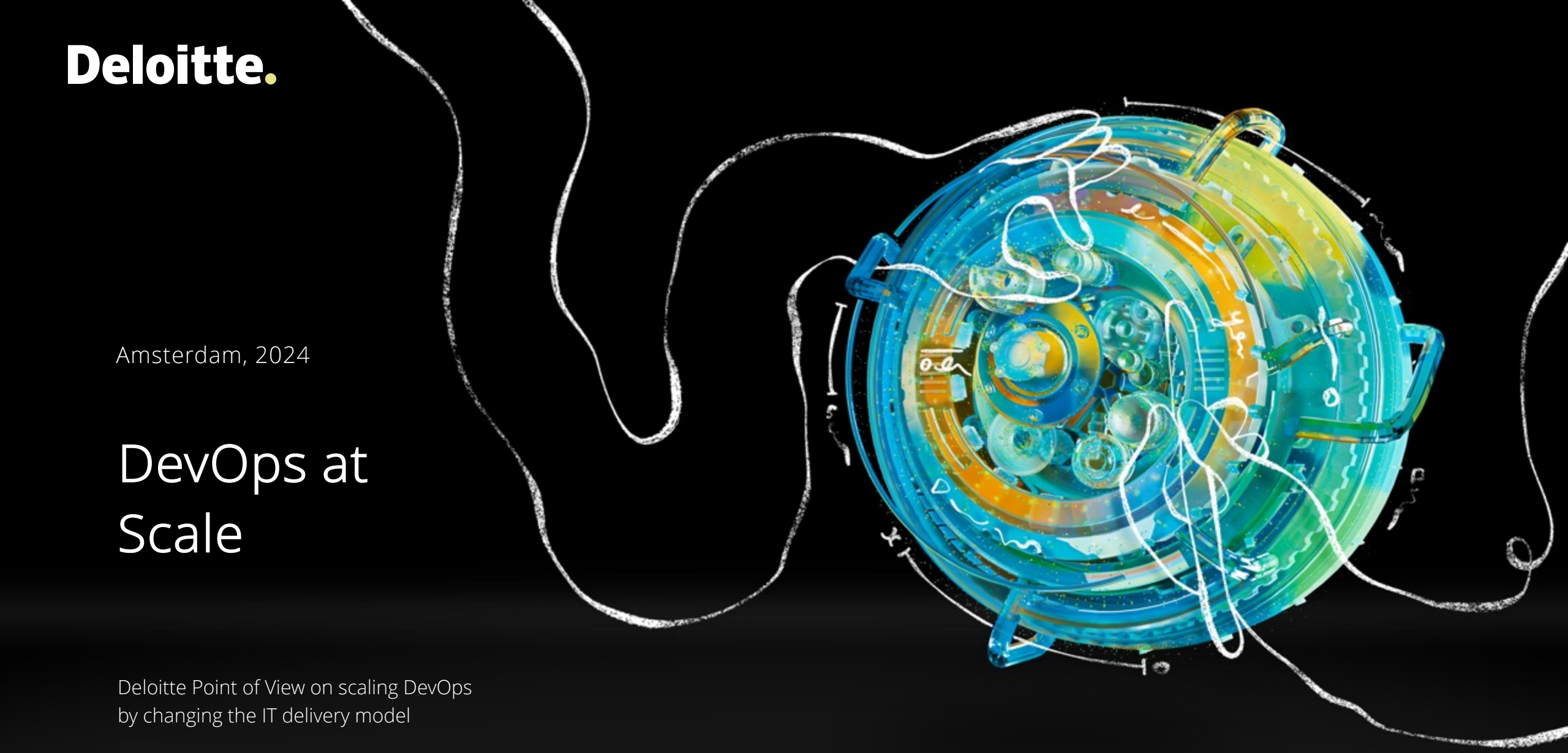
Deloitte.

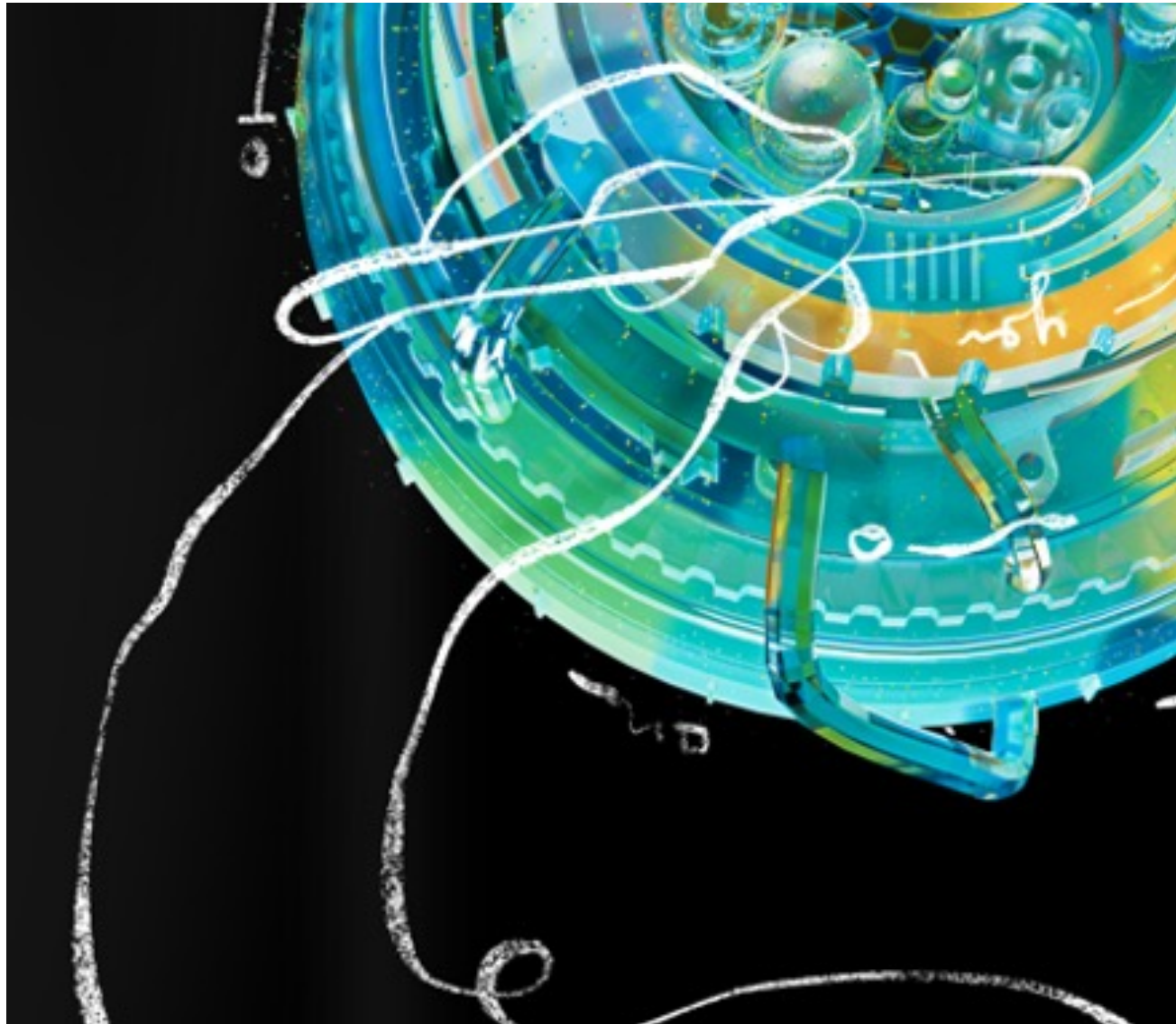


Amsterdam, 2024

DevOps at Scale

Deloitte Point of View on scaling DevOps
by changing the IT delivery model

Contents



DevOps introduction and relevance

What is DevOps?

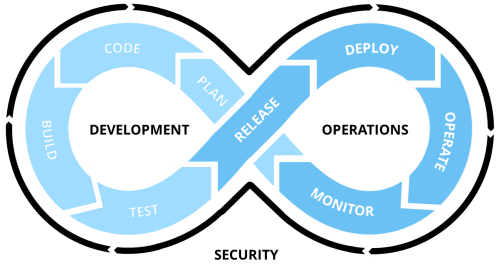
Definition

DevOps is approach to optimize and manage end-to-end service delivery and operations. It applies a set of principles to transform the entire software delivery lifecycle to introduce new practices enabled by technology

DevOps principles

- Culture of shared responsibility and collaboration
- End-to-end ownership of services
- Multi-disciplinary teams
- Incremental value delivery
- Flow optimization in the delivery process
- Automate (almost) everything
- Measurement of everything
- Continuous improvement

Software Delivery Lifecycle



Applying DevOps principles and practices to the SDLC will benefit both Development and Operations on several aspects

Introduces new practices:

- Continuous Integration
- Continuous Testing
- Continuous Delivery
- Continuous Operations
- Integrated Security

Goal

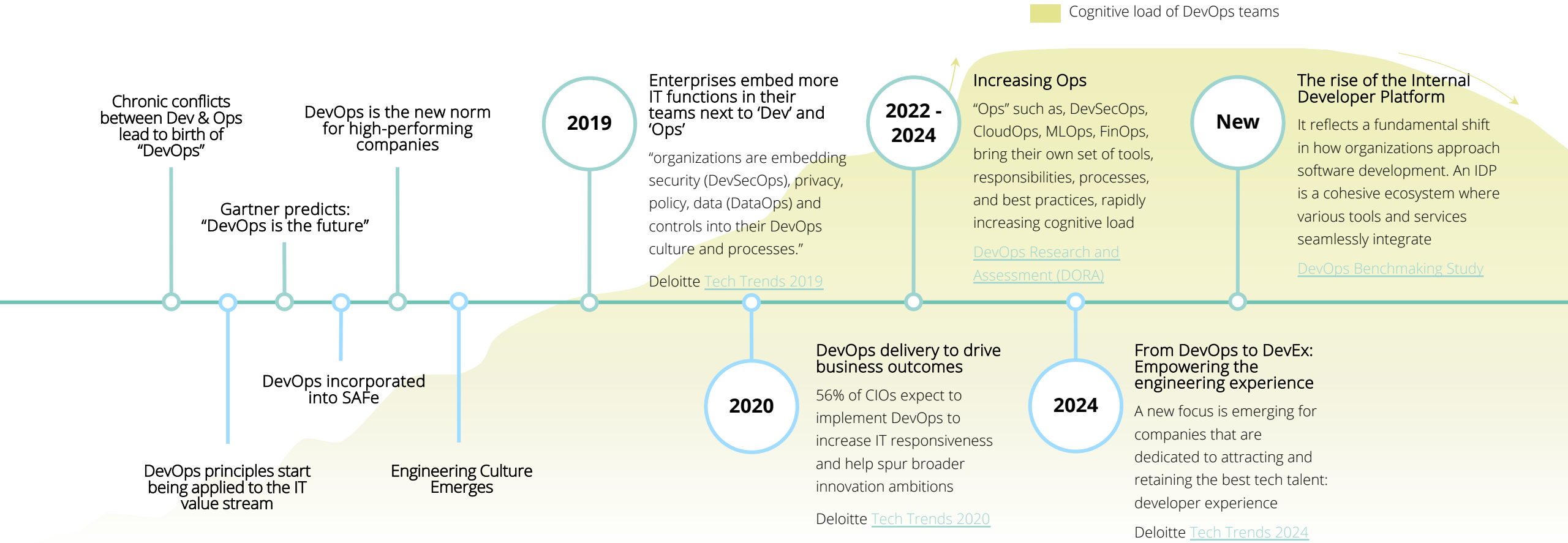
DevOps primary goal is to improve the flow from an idea towards value for the customer, enabled by an environment in which multidisciplinary teams work collaboratively to continuously deliver high quality solutions, in a faster pace, that qualify for operations

Benefits

- Increases the frequency and quality of deployments and releases
- Improves innovation and risk-taking
- Realizes faster time to market
- Improves solution quality and operational reliability
- Improves the Mean Time to Recover (MTTR)

The evolution of DevOps

DevOps is the norm in software delivery and is increasingly being adopted & matured across enterprises. However, the increasing cognitive load on DevOps teams leads to challenges when scaling DevOps enterprise-wide and demand for changes to the IT Delivery Model of organizations



Why DevOps is more relevant than ever

In today's digital landscape, DevOps allows swiftly addressing challenges and enabling organizations to develop, deploy, and iterate on software rapidly, efficiently, and securely, achieving increased business model adaptability

Lack of collaboration between development and operations teams, people working in silo's

- DevOps fosters a collaborative culture between development and operations teams, emphasizing on cross-functional teams and shared responsibilities
- DevOps encourages close cooperation during the design phase, consideration of operational requirements
- This results in innovative yet feasible, and manageable production designs

Limited scalability and resource constraints

- DevOps utilizes infrastructure as code (IaC) and cloud technologies to automate the provisioning and scaling of resources
- Enables automatic creation and flexible scaling of infrastructure for development, testing, and deployment needs
- DevOps promotes the use of monitoring and feedback loops, enabling rapid issue identification and resolution, resulting in faster and more reliable software development

No end-to-end integrated process with slow and manual integration and deployment processes

- DevOps automates the integration and deployment process using Continuous Integration/Continuous Deployment (CI/CD) pipelines
- By automating code integration, testing, and deployment, DevOps ensures rapid and reliable delivery
- Organizations can release new features and updates to customers faster and more frequently

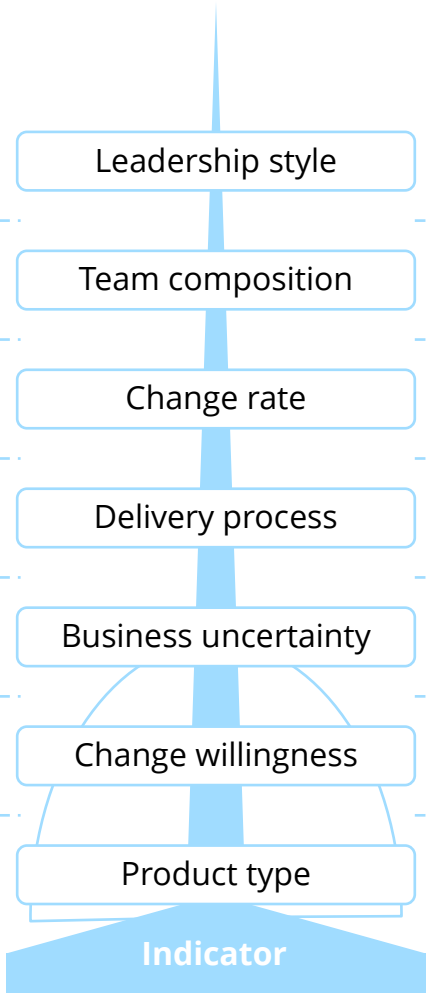
Increased cognitive load

- DevOps utilizes Internal Developers Platform (IDP) providing standardized, self-service tools and environments, simplifying complex development processes
- By automating repetitive tasks such as provisioning resources and setting up development environments, IDP frees developers from cognitive burden

Indicators for DevOps Applicability

The DevOps Applicability Indicator scale help to determine if DevOps is applicable in your organization based on an evaluation of leadership style, team composition, change rate, delivery process, business uncertainty, change willingness and product type

DevOps is Applicable		**DevOps is Not Applicable**
Management trusts delivery teams to work autonomously and only shares a product vision	Leadership style	Management requires direct involvement in the delivery process and makes all decisions
Multiple teams are responsible to manage the end-to-end lifecycle of a single product	Team composition	Product or service delivery does not require a multi disciplinary (cross-functional) team
Environments where IT solutions are changing rapidly	Change rate	Environments where IT solutions have low change rate
An incremental delivery process that focuses on early value delivery	Delivery process	Your delivery process has many sequential constraints, where outputs equal required inputs for consecutive process steps
Desired product end-state is unknown, changing business requirements give guidance on steering development	Business uncertainty	Desired product end-state is known and business requirements do not often change
People have great affinity with software and technology and are not change averse	Change willingness	People have no affinity for new technologies, and are change averse
Product is software that could be delivered as-a-service	Product type	Products are tangible, typically consisting of semi-finished products provisioned by multiple partners that don't have a direct relation with each other

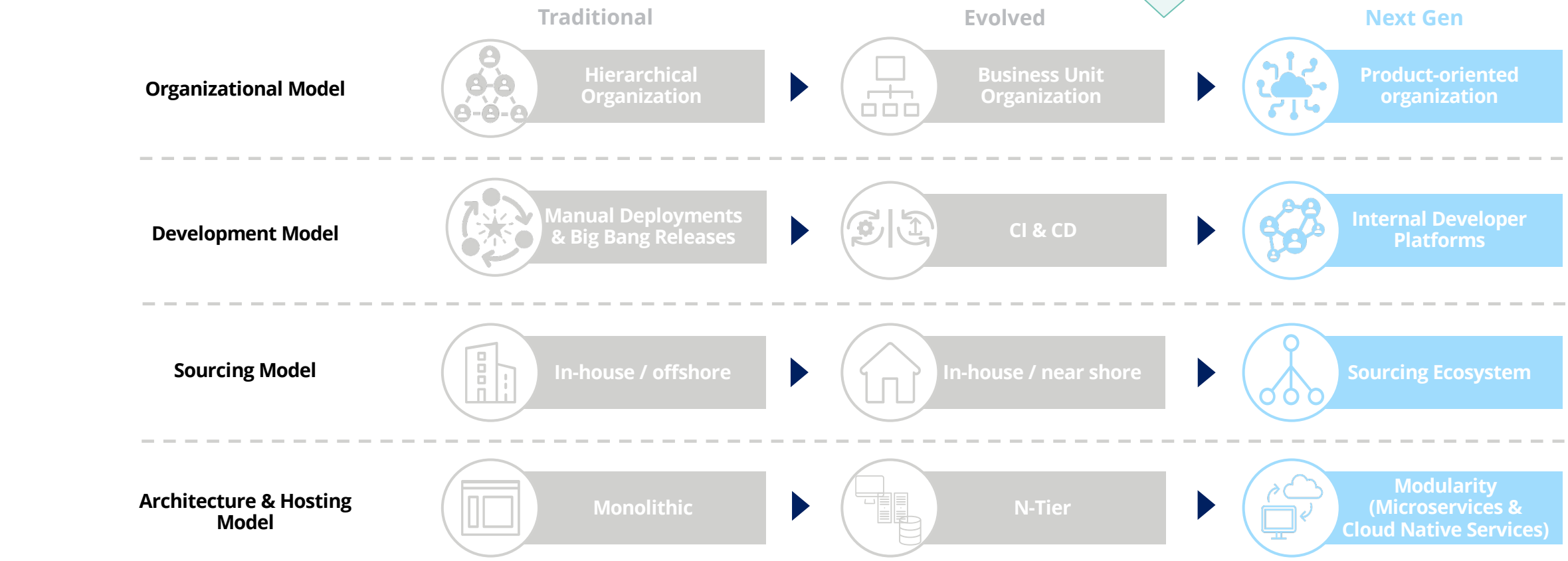


Changing IT delivery model

The changing IT Delivery Model

Overcoming challenges when scaling DevOps enterprise-wide (i.e. cognitive overload of DevOps teams and sourcing) requires four new patterns in the IT delivery model: organizing product-oriented, leveraging Internal Developer Platforms and modular architectures and implementing a sourcing ecosystem

We see that most organizations want to move from 'Evolved' to 'Next Gen' delivery. **This document presents Deloitte's View on 'Next Gen' patterns required to run DevOps at Scale**



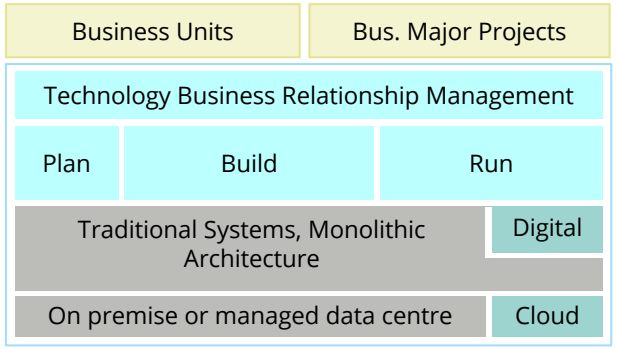
Organizational Model

Operating model archetypes overview

Generally technology operating models will follow one of the archetypes outlined below. While there is no right or wrong model, most companies are seeking to adopt either a ThinIT™ or product based operating model

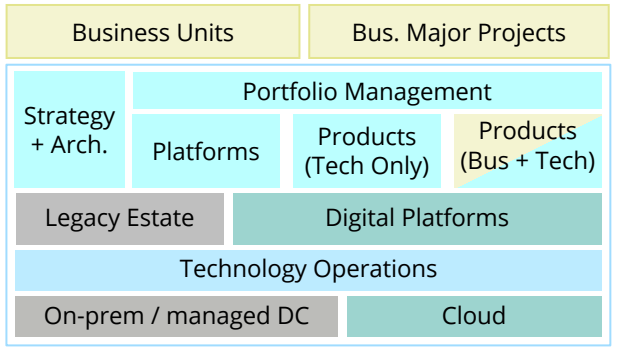
DRIVING EFFICIENCIES → ENHANCING WHAT'S IN PLACE → DRIVING BUSINESS VALUE

TRADITIONAL PLAN, BUILD, RUN MODEL



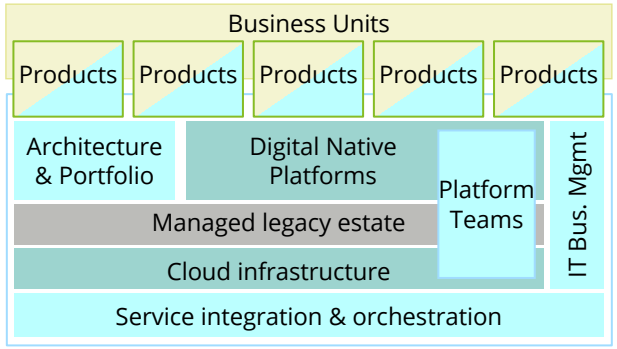
IT is segregated into core IT functional areas, has project based technology delivery and is focused on driving operational efficiency

PRODUCT & PLATFORM MODEL



Agility, DevOps and modern technology is adopted in part of the IT organisation (e.g. digital), with the remainder of IT operating in a traditional plan, build, run model

FULLY EMBEDDED BUSINESS PRODUCT MODEL

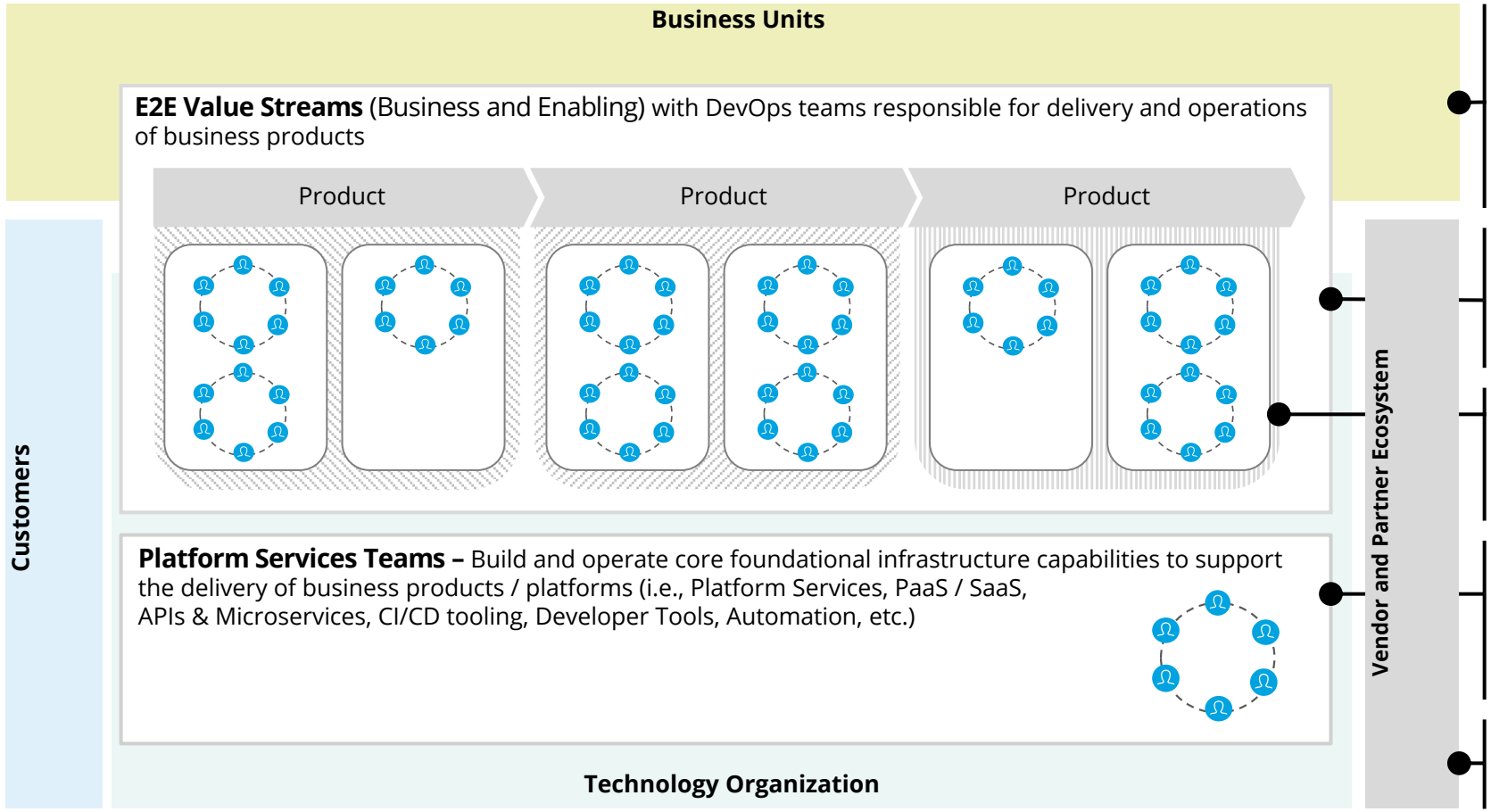


Agility principles are scaled across technology with capabilities structured around cross-functional business-technology product or platforms that deliver both Change and Run. This model is also referred to as ThinIT™

Source: Tech Tom in a Box May 2023

DevOps Organizational Model

The most advanced DevOps Organizational Model is a fully embedded business product model that allows scaling agility / DevOps principles across technology. Capabilities are structured around cross-functional business-technology products or platforms that deliver both Change and Run



This is a fully embedded business product model overview where **business and technology change capability is combined** into cross-functional product teams and value streams with the aim of **increasing business outcome realization** and **reducing the cost to deliver change**

Business and IT co-create digital solutions in **cross-functional, outcome-focused product teams** aligned by product offering and / or domain

Cross-functional DevOps teams (or squads, pods etc.) apply DevOps processes to obtain continuous customer feedback and ensure products meet market demands

Solutions are **digital by design**, supported by **modern cloud and micro-service architecture**. Digital-native Platforms are built and **maintained by** cross-functional **Platform Teams**

Additional **emphasis** placed on **ecosystem and sourcing** to leverage evolving talent pools (e.g., crowd), assets and solutions

Zoom-in: DevOps Organizational Model

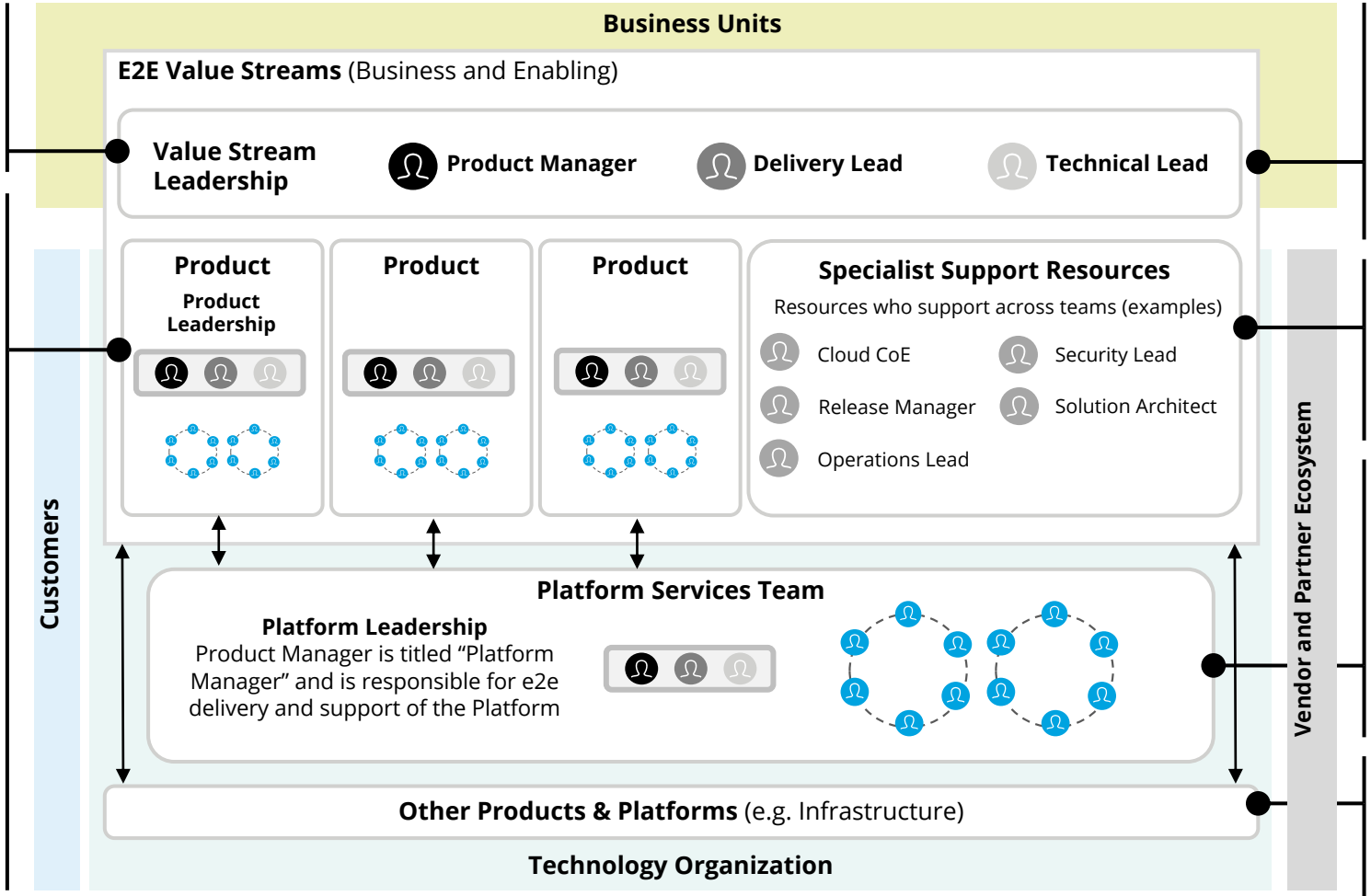
The model shows an example high level structure for a Value Stream, Product and Platform

Each **Value Stream is led by a group of leaders** focused on defining and realizing the business and technical vision for a Product Group

A **Product** is a named collection of business capabilities valuable to a customer segment (Gartner)

Each Product is **supported and enhanced end to end by one or more multi-disciplinary DevOps teams** (internal or vendors). This includes supporting and maintaining systems in their scope (or ensuring vendors provide these services)

A **Product Owner from the Business** is appointed to prioritize product and team backlogs, provide direction on business needs, and provide business acceptance of features that are produced. Product Owners can work with one team, or multiple depending on context



Value Streams are an organizational construct to group together similar smaller products. This is to 1) optimize reporting lines, 2) create a shared identity and focus for teams 3) optimize the flow of work

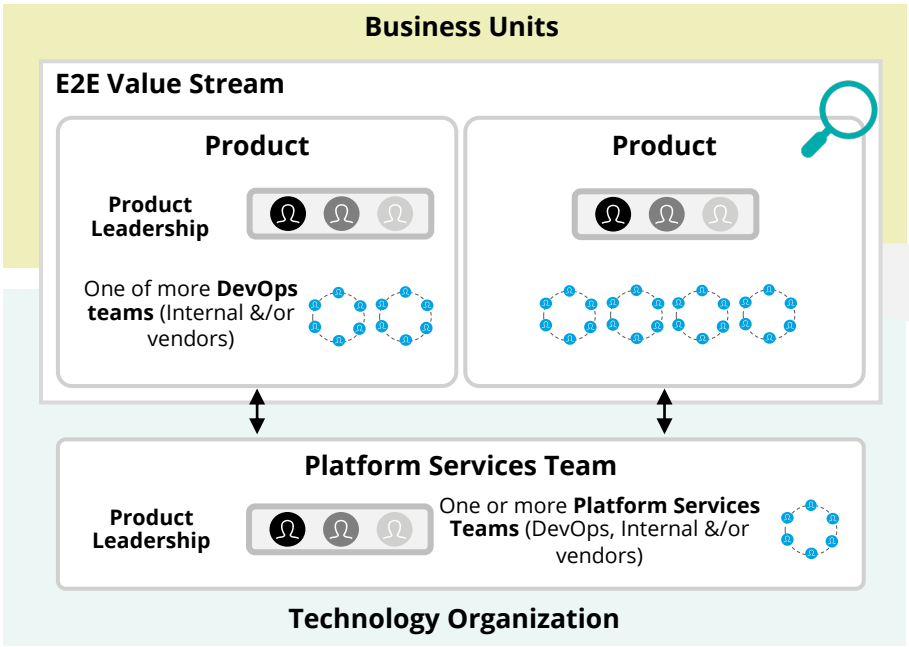
Some resources work across multiple DevOps teams within a Value Stream. These are examples only, not all Products and Platforms will require these roles

A **Platform** provides a technology solution that can be leveraged and built on by Products. Platform teams (DevOps, Internal &/or vendors) should aim to make their Platform as self-service as possible to simplify Product delivery

Teams leverage shared capability provided by other Platforms, and manage delivery dependencies with other Products

DevOps Team: Roles and Responsibilities

DevOps Teams are autonomous, service / product-oriented teams that possess multi-disciplinary capabilities. These teams will own the entirety of the software delivery and product lifecycle, including design, delivery and operations



The **Business Analyst** engages the business for requirements, helps defining features, user stories & test cases, and validates designs

The **Product Owner** leads the DevOps team and directs development to maximize business value

The **Developer(s)** writes and verifies code, fixes bugs, automates processes, collaborates across teams, and ensures secure, efficient, and continuous software delivery.

The **Test Engineer(s)** creates and executes test scripts, automates tests, supports usability testing & UAT, and manages test environments and test data

The **Scrum Master** facilitates the team on processes & approach, manages impediments and enables continuous improvement

The **Operations Specialist(s)** executes day-to-day technology operations (functional maintenance) of the application, monitors technology operations, performs Problem Management, manages change processes (Approves/Rejects)

The **DevOps Engineer*** configures the CI/CD toolset provided by the Platform Services team within the set guardrails to accelerate and de-risk application delivery

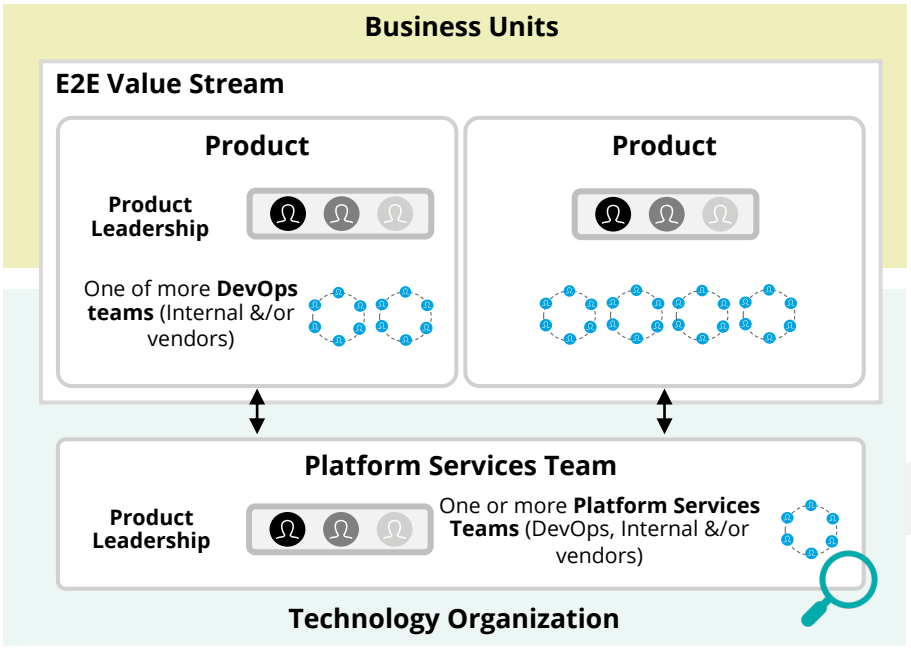
* If the organization does not have a Platform Services Team with a Platform Engineer, the DevOps Engineer in a DevOps team takes over the responsibilities/activities of the Platform Engineer

IT role

Business role

Platform Services Team: Roles and Responsibilities

The Platform Services Team provides a technology solution (i.e. Internal Developer Platform) that can be leveraged and built on by the DevOps teams. They specialize in building and maintaining the core foundational technology platforms, such as core infrastructure, monitoring and logging controls, security controls, in addition to supporting shared services and establishing centres of excellence (COEs) for innovation areas across the organization



The **Scrum Master** facilitates the team on processes & approach, manages impediments and enables continuous improvement

The **Platform Owner** oversees the implementation of the platform, ensures alignment with business goals and manages stakeholders expectations

The **Platform Architect** designs and plans the underlying platform architecture, ensures scalability and alignment with the business goals of the organization

The **Platform Engineer*** delivers, maintains and monitors the CI/CD toolset, implements and monitors the platform performance and deploys self-service capabilities

The **Infrastructure (e.g. Cloud) Engineer** implements and maintains the IT infrastructure, monitors the servers and administers the network

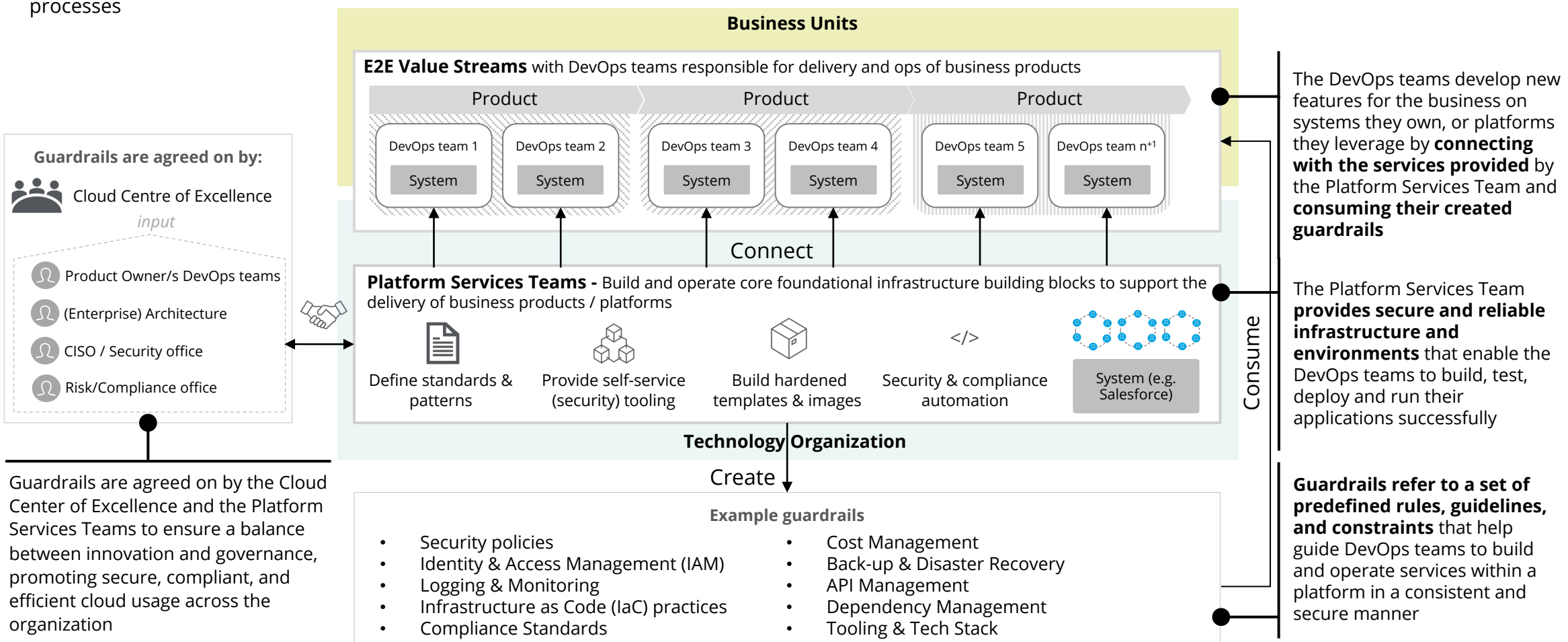
The **Security Engineer** builds and maintains security protocols, identifies and solves security vulnerabilities, conducts regular security assessments and responds to security incidents

The **Operations Specialist** executes day-to-day technology operations (functional maintenance) of the platform, monitors technology operations, performs Problem Management, manages change processes (Approves/Rejects)

* If the organization does not have a Platform Services Team with a Platform Engineer, the DevOps Engineer in a DevOps team takes over the responsibilities/activities of the Platform Engineer

Collaboration Platform Services Team and DevOps Teams

DevOps teams should adhere to agreed standards & patterns, and consume the tooling, images, templates & automation scripts/code built by the platform team. The sum of all technology and tooling provided by the Platform Team is encompassed in an Internal Developer Platform (IDP) to optimize the development processes

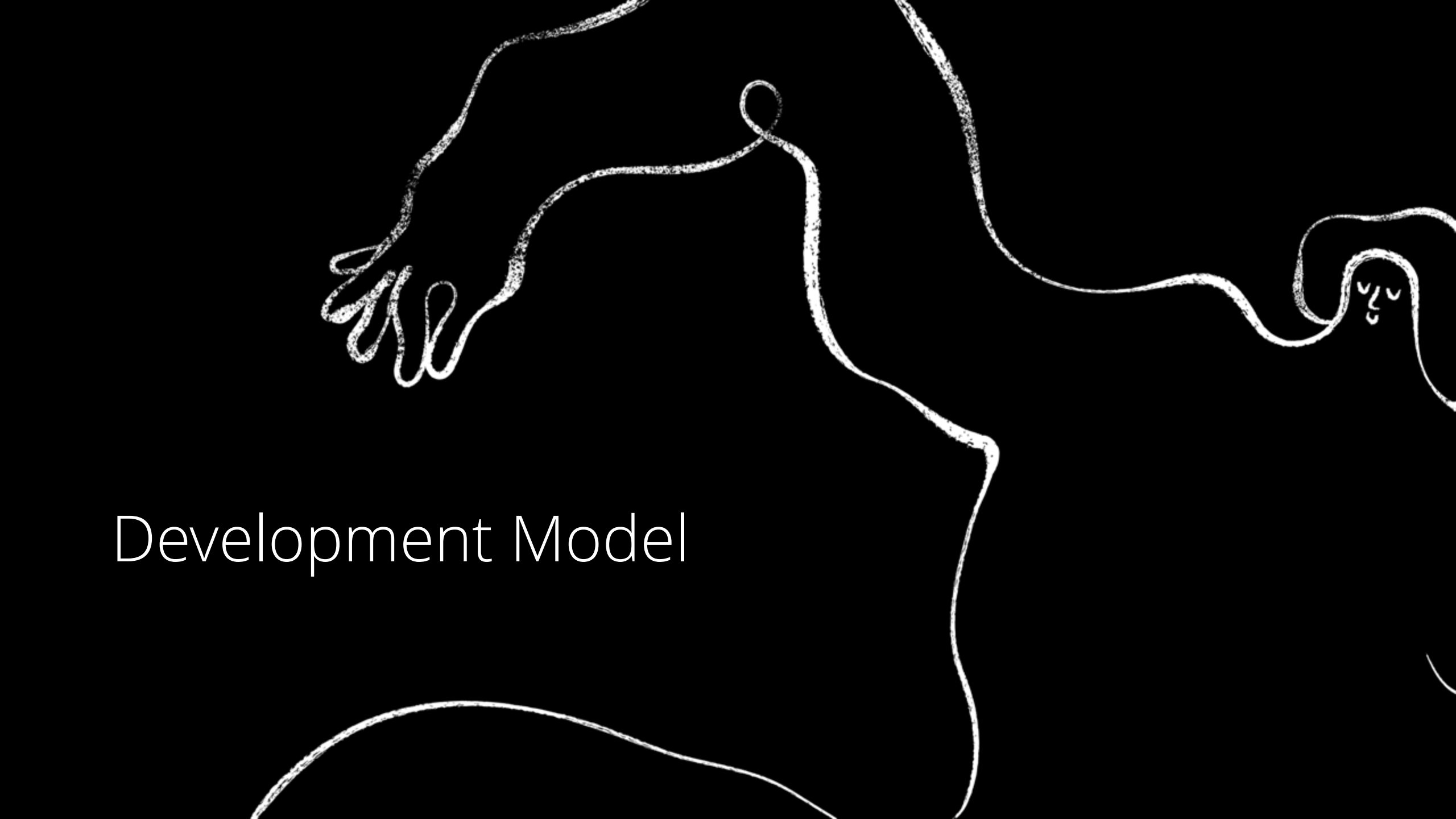


Guardrails are agreed on by the Cloud Center of Excellence and the Platform Services Teams to ensure a balance between innovation and governance, promoting secure, compliant, and efficient cloud usage across the organization

The DevOps teams develop new features for the business on systems they own, or platforms they leverage by **connecting with the services provided** by the Platform Services Team and **consuming their created guardrails**

The Platform Services Team **provides secure and reliable infrastructure and environments** that enable the DevOps teams to build, test, deploy and run their applications successfully

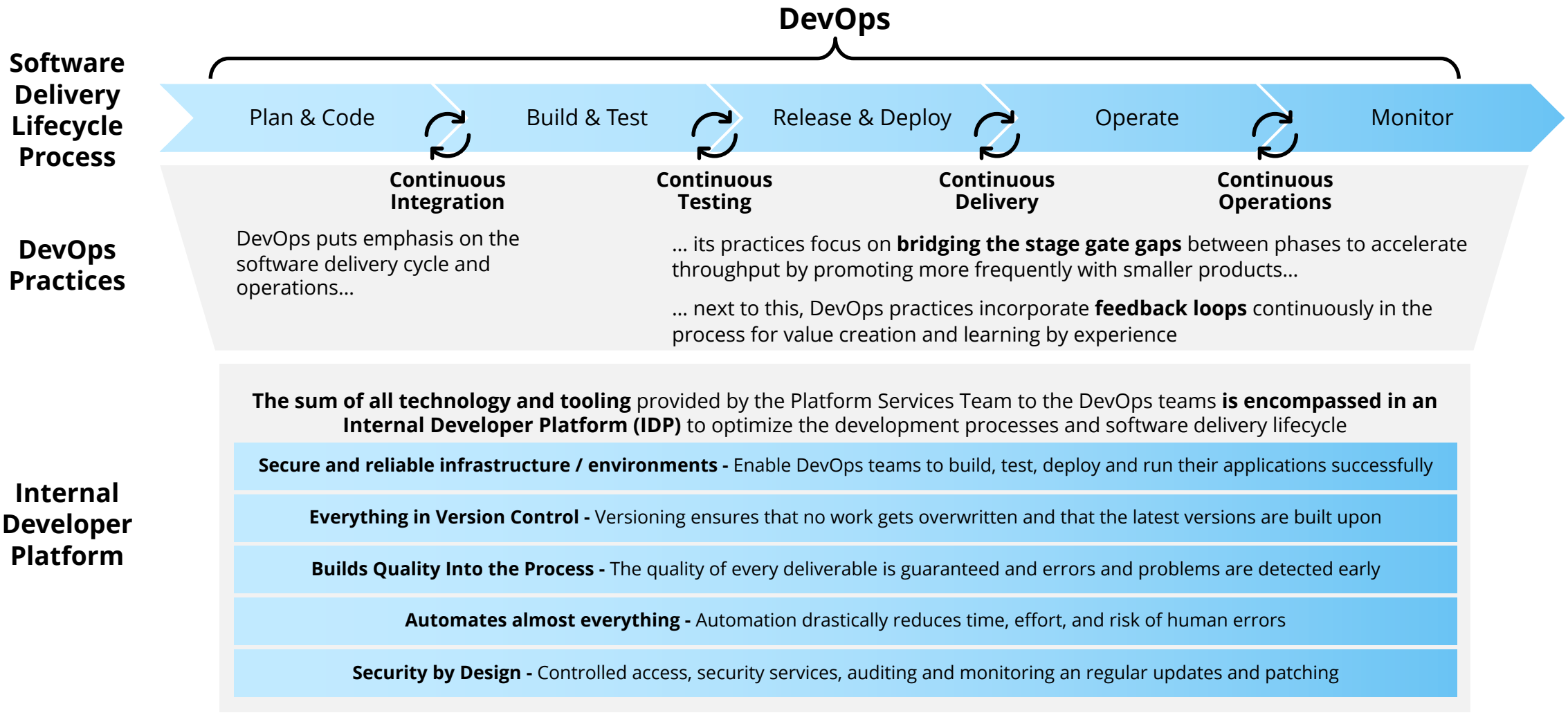
Guardrails refer to a set of predefined rules, guidelines, and constraints that help guide DevOps teams to build and operate services within a platform in a consistent and secure manner



Development Model

DevOps Development Model

DevOps optimizes the software delivery process by leveraging an Internal Developer Platform (including CI/CD pipeline) which automatically promotes developer's source code to operational solutions



Software Delivery Process and DevOps practices

DevOps practices apply continuous automation cycles throughout software development and operations processes

Continuous Integration

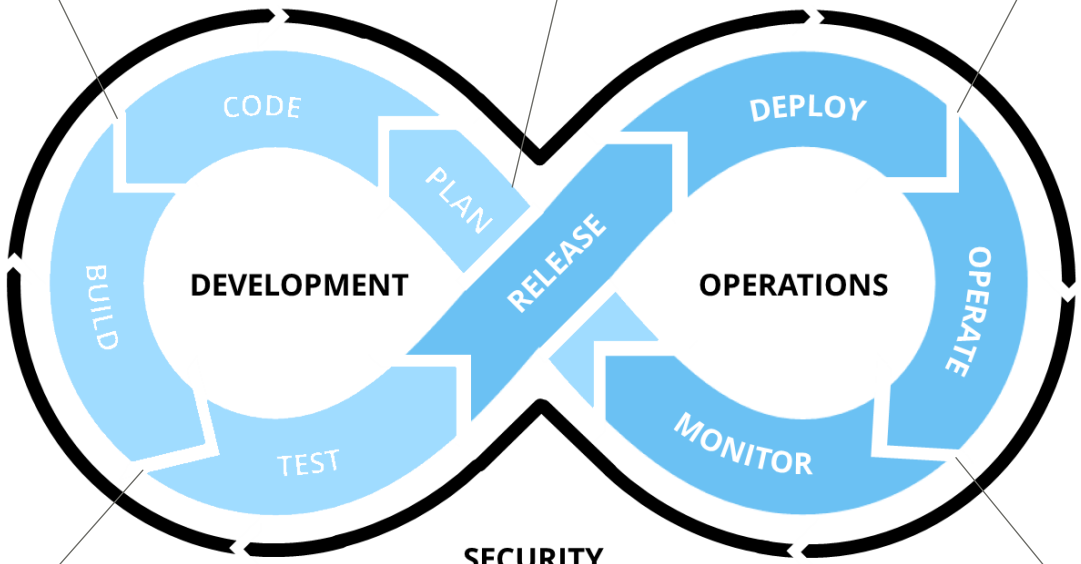
Streamlining of internal development by integrating code into a shared repository several times a day. Each check in is then verified by an automated build, allowing teams to detect problems early in the cycle

Continuous Planning

Identifying and prioritizing opportunities, changes and fixes, based on continuously captured end-user demand

Continuous Delivery

Delivering code that is production ready and is kept in an always releasable state, so it can be deployed (automatically) to production at any given time based on business needs



Continuous Testing

Automating and integrating tests into the software delivery chain, and automatically executing those tests against each build of the code base

Continuous Operations

Proactively managing the solution based on feedback loops. Monitoring and telemetry become part of the backlog. Processes such as patching also fall under this practice

Security practices should be integrated throughout the software delivery lifecycle process. Security integrated in the process increases an organization's ability to deliver applications and services at high velocity, securely

Internal Developer Platforms

DevOps evolvement has been a significant driver in the evolution of Internal Developer Platforms (IDPs). IDP is a platform within an organization that is designed to streamline the software development process for developers. The rise of IDPs has been influenced by several factors

Cognitive Load

- The rise in cognitive load for software developers, due to the **increasing complexity of technology stacks** and the need to **manage multiple systems and tools**, has led to the creation of internal developer platforms (IDPs)
- As software projects grew in complexity with multi-tier architectures, microservices, and cloud-native technologies, there arose a need for more enhanced tooling and resources to manage this complexity

Engineering Culture

- Engineering culture has increasingly valued efficiency, collaboration, and innovation, leading to the adoption of internal developer platforms that foster these values by reducing silos and streamlining workflows
- This cultural shift towards DevOps practices and continuous delivery has made internal developer platforms essential for maintaining a competitive edge by enabling faster, more reliable, and scalable software development life cycles

Need for Speed and Agility

- The need for rapid scaling and agility in software delivery has arisen due to the fact that businesses face increasing pressure to bring products and features to market quickly while maintaining high quality and compliance with regulatory standards
- Competitive pressures and the demand for rapid innovation have forced businesses to accelerate their software delivery cycles

Internal Developer Platforms

- Internal developer platforms streamline the development process, offering integrated environments that automate and abstract operational tasks, allowing developers to focus on writing code rather than managing infrastructure
- These platforms become relevant in large-scale organizations because they help manage the complexity and scale of big teams and projects by standardizing development environments, automating workflows, and ensuring consistency across multiple deployments

Internal Developer Platform features and benefits

Internal Developer Platforms typically consists of a range of tools, technologies, protocols, and best practices that are standardized across the organization to improve efficiency, consistency, and scalability of software development projects. Some key features and benefits of IDPs are:

Standardization

IDPs help standardize the development, deployment, and operational processes across multiple teams and projects. Standardization can **reduce complexity** and **the learning curve** for new developers

Automation

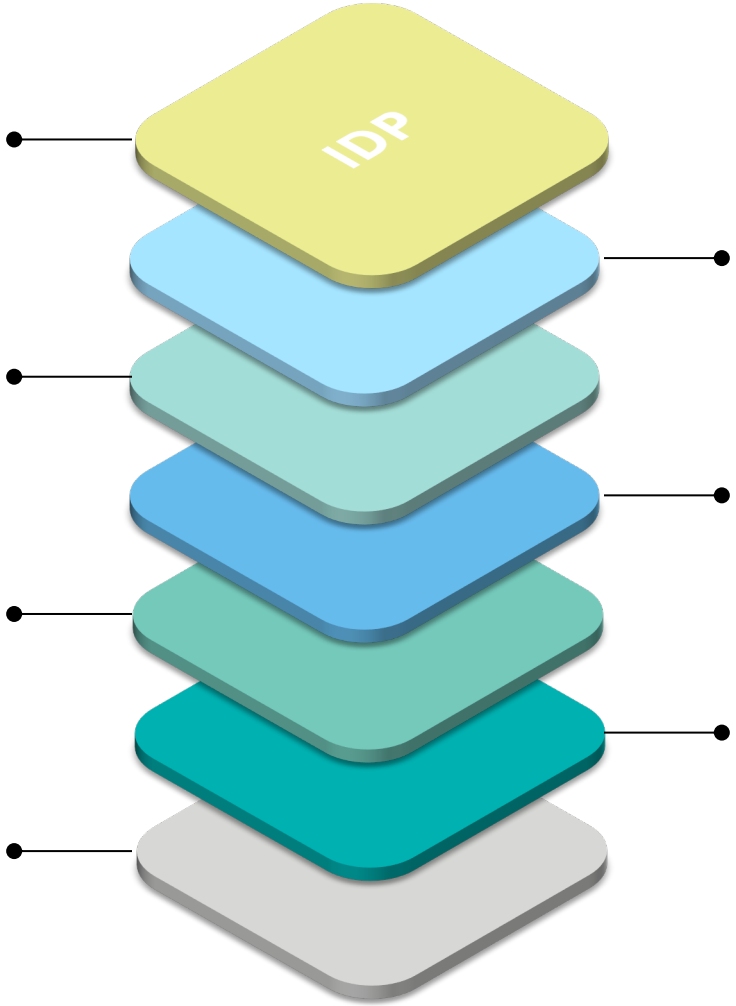
IDPs automate repetitive tasks such as code integration, testing, deployment, and infrastructure provisioning, which **speeds up the development cycle** and **minimizes human error**

Self-service capabilities

Developers can initiate processes, access resources, and manage environments on their own without waiting for operations or other teams, **fostering agility** and **productivity**

Integration of tools

IDPs integrate a variety of tools for version control, continuous integration/continuous deployment (CI/CD), monitoring, and more, providing a **cohesive workflow**



Best practices and governance

IDPs embed compliance, security, and architectural best practices into the development pipeline, ensuring that applications **meet organizational** and **industry standards**

Collaboration

By using a common platform, developer teams can **collaborate more effectively**, share code, and **maintain consistency** across different projects

Resource optimization

IDPs can help in **optimizing resource utilization** by managing **infrastructure as code** and **providing insights** into application performance and cost

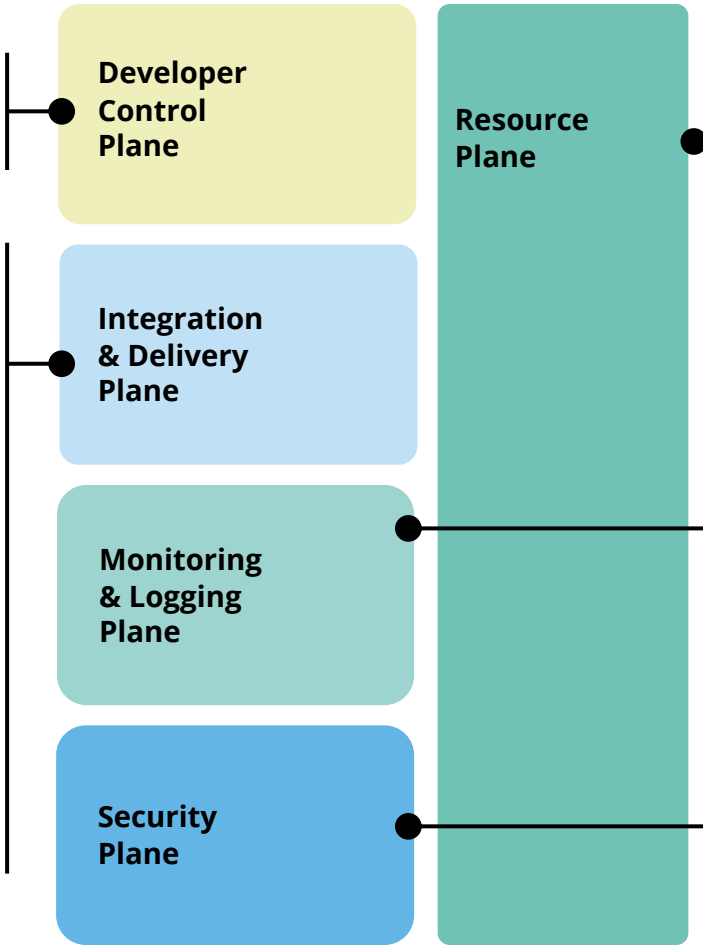
IDP Tooling Landscape – planes

The reference architecture of an Internal Developer Platform (IDP) can be broken down into five functional planes: Developer Control Plane, Integration & Delivery Plane, Monitoring & Logging Plane, Security Plane and Resource Plane

The Developer Control Plane provides developers with tools to **configure** and **interact** with their development environments, define **project-specific settings**, and **manage dependencies** efficiently

The Integration & Delivery Plane refers to the set of tools, processes, and infrastructure dedicated to **build, store, configure** and **deploy** requests from the developer control plane. It typically includes:

- **Continuous Integration (CI):** automatically building and testing code changes
- **Continuous Delivery (CD):** CD extends CI by automatically deploying code changes to various environments
- **Image Registry:** This involves storing and managing software artifacts in a central repository that ensures versioning, accessibility and easily
- **Platform Orchestrator:** tools designed to automate the deployment, management, and scaling of containerized applications across a cluster of machines
- **Infrastructure Control:** manages and orchestrates the underlying infrastructure resources (e.g., infrastructure as a code facilitation)

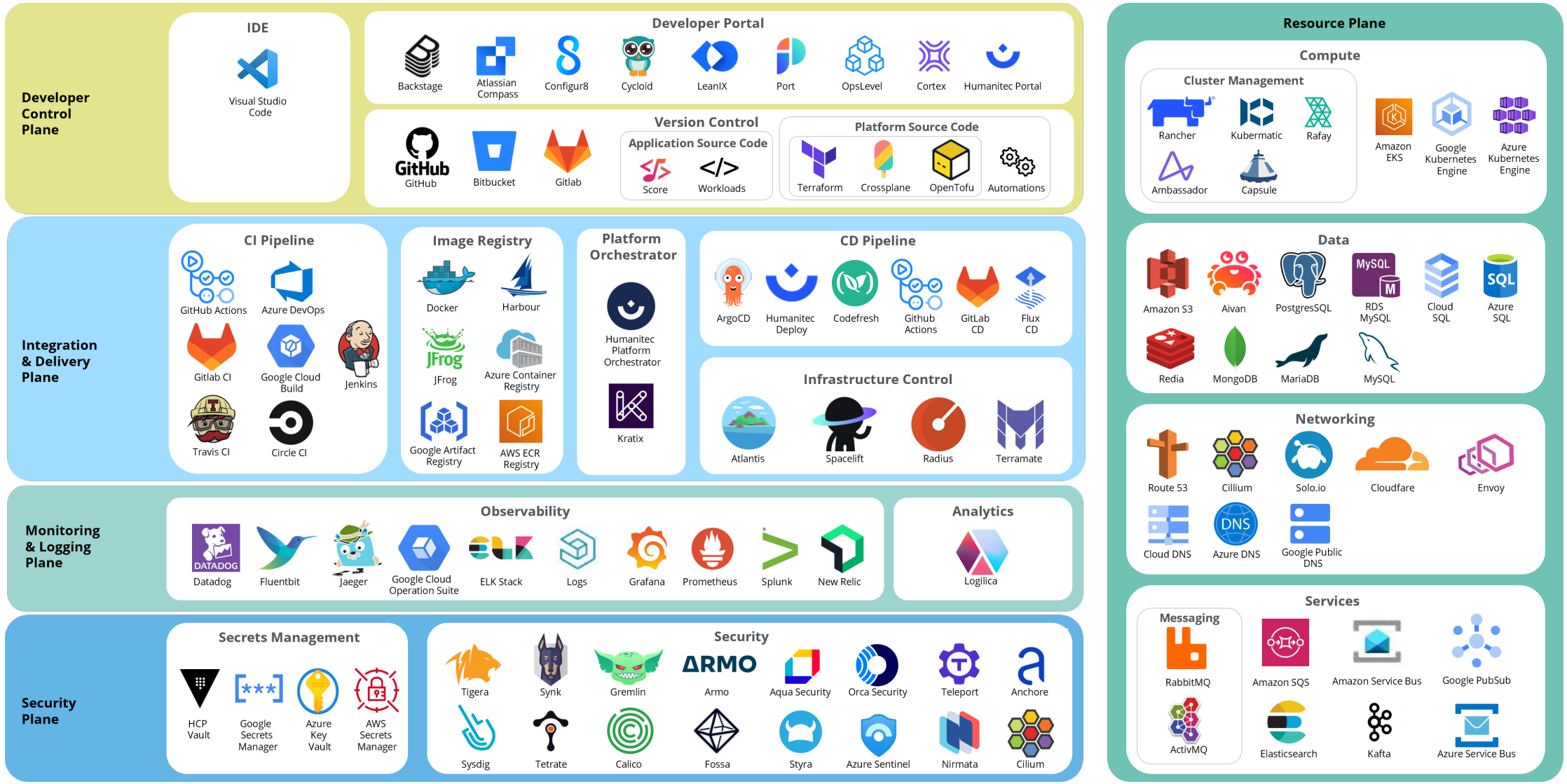


The Resource Plane **manages the allocation, provisioning,** and **utilization of computing resources** such as CPU, memory, storage, and network bandwidth within the platform

The Monitoring & Logging Plane involves tools for **implementing monitoring & logging solutions** to track the **performance**, **availability,** and **behavior of deployed applications**. It helps in identifying issues and optimizing system performance

The Security Plane includes features to **integrate security checks** and **compliance policies** such as access controls, authentication mechanisms, encryption vulnerability scanning, cyber threats

IDP Tooling Landscape – planes and tooling



Source: https://humanitec.com

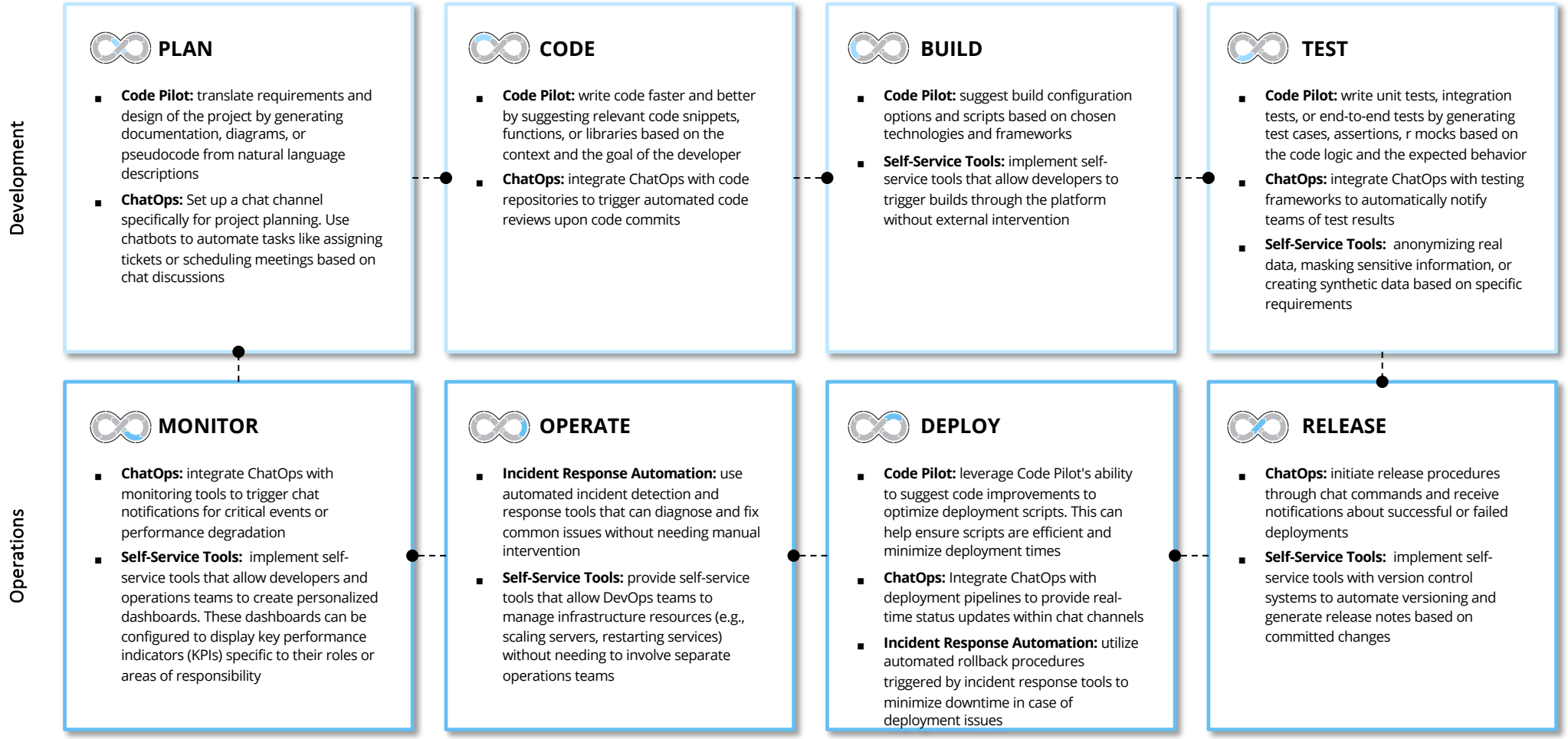
Key Trends influencing the Development Model

There are some key trends evolving related to the development model that empower DevOps teams. These key trends include the utilization of Code Pilot, ChatOps, Incident Response Automation, and Self-Service Tooling.

	Code Pilot	ChatOps	Incident Response Automation	Self-Service Tools
Key Trend Description	Code Pilot – also known as CoPilot – is an **AI-driven coding assistant** optimizing developer productivity through code suggestions, completion, and context-aware recommendations, fostering efficiency and best practices	ChatOps employs **chat clients, chatbots, and other real-time communication tools** to streamline software development and IT operations tasks and enhance an organizations' collaboration	Incident Response Automation utilizes **automated processes and tools for rapid detection, analysis and response to incidents** that would normally require human intervention	Self-Service Tools empower DevOps teams to **independently manage and provision resources, access information, and perform routine tasks** without extensive reliance on external assistance
Benefits for DevOps	☑ Reduces manual effort and boosts the developer's efficiency ☑ Enhances code quality by facilitating best practices and error detection, ultimately leading to fewer bugs and faster deployment cycles ☑ Developers benefit from Code Pilot's insights and recommendations, stimulating continuous learning	☑ Communicate and coordinate better during incidents in real-time, all within a chat environment ☑ Automate tasks, share knowledge, and monitor systems ☑ Enables teams to execute commands, view system performance metrics, and receive notifications directly in chat channels	☑ Ensures that predefined actions are taken consistently during incidents ☑ Decreases the incident detection, analysis and response time ☑ Reducing the likelihood of human errors, leading to more reliable incident resolution processes ☑ Includes tasks such as restarting services, scaling resources, and notifying the relevant stakeholders	☑ Provide user-friendly interfaces and predefined workflows ☑ Allowing team members to deploy applications, provision resources, and troubleshoot issues autonomously ☑ Developers and operators independently manage their workflows, reducing dependency bottlenecks and enabling rapid experimentation and innovation

Example Usage of Key Trends in the SDLC

How Code Pilot, ChatOps, Incident Response Automation, and Self-Service Tooling can be utilized in the Software Delivery Lifecycle



Sourcing Model

Sourcing in DevOps context

When scaling a DevOps organization, a sourcing strategy and ecosystem becomes increasingly relevant. A well-defined sourcing model plays a crucial role in supporting the DevOps practices by providing the necessary resources and capabilities to enable the software delivery lifecycle with speed, efficiency, and reliability

What is Sourcing?

Sourcing, in a DevOps context, refers to the strategic approach of acquiring and managing resources, services, and expertise to support the software delivery lifecycle

Sourcing involves the choice for an engagement model and selection of vendors and partners with who the organization will work within an ecosystem

Engagement Models

Engagement models define the framework for how both the organization and the vendors/partners collaborate and interact throughout the software delivery lifecycle

An engagement model specifies the roles, responsibilities, and commitments of each party, guiding the execution of tasks and deliverables

Vendor and Partner Ecosystem

The vendor and partner ecosystem comprises of a network of suppliers, collaborators, and stakeholders engaged in delivering IT solutions according a set engagement model

It involves various entities, from independent contractors to outsourcing firms, each contributing unique capabilities

Sourcing: the reasons behind it

Sourcing offers a strategic advantage by providing flexibility, access to specialized skills, cost efficiency, speed, and global reach, making it an essential component for modern DevOps organizations. Sourcing is often used to scale the DevOps organization, setup globally and/or realize 24/7 support

Scaling DevOps
Organizations experiencing rapid growth or expansion of DevOps initiatives often turn to sourcing to meet increasing demands efficiently

Global Organization
Multinational companies with dispersed teams and operations leverage sourcing to harness global talent and resources

Follow the Sun
Adopting sourcing allows for continuous development and support by leveraging teams in different geographical locations, ensuring 24/7 coverage

Benefits of sourcing

Enhanced Flexibility Sourcing offers scalability and adaptability, allowing organizations to quickly adjust resources around the clock based on delivery needs

Access to Specialized Skills Sourcing provides access to a diverse pool of talent and expertise, ensuring the right skills are available for each engagement phase

Cost Efficiency Sourcing models often result in cost savings through optimized resource allocation, reduced overhead, and competitive pricing

Faster time to Market By tapping into external resources, organizations can accelerate development cycles and bring products to market more rapidly

Global Reach Sourcing enables organizations to tap into global talent pools and establish a presence in multiple markets without geographical constraints

Sourcing engagement models

When the DevOps organization engages with vendors and partners to acquire capacity or services, it can utilize a variety of engagement models. Four distinct models are available, each suited to the specific scope of the Software Delivery Lifecycle being outsourced.

	Output Commitment (Time for Hire)	Outcome Commitment	End-2-End SDLC (Managed Service)	Asset / Product or Platform as a Service
Engagement Model				
Description	Utilizing Time and Material (T&M) contracts for loaned staff, ensuring capacity and quality commitments	Commitment to project deliverables, with ownership of specific parts (e.g., Development) of the Software Delivery Lifecycle	Outsourced Product Delivery Teams operating as independent entities or as subsidiaries of outsourcing firms being E2E responsible for the full SDLC; from Dev to Ops	Integration of supplier-owned assets or services, where supplier holds E2E accountability, promoting innovation and efficiency (e.g. Deloitte Release Orchestration Pipeline - DROP)
Key characteristics	• **Early Engagement:** Utilize for initial stages of the product where staffing needs may fluctuate • **Flexibility:** Provide flexibility to scale resources up or down based on project requirements • **Quality Assurance:** Ensure resources meet quality standards and possess necessary skills for project tasks	• **Clear Objectives:** Define clear project objectives and deliverables to align with client expectations • **Commitment to Results:** Take ownership of specific project components and commit to delivering high-quality outcomes • **Continuous Communication:** Maintain open communication channels with clients to ensure alignment and transparency throughout the project	• **End-to-End Integration:** Assume responsibility for the entire SDLC • **Cross-Functional Collaboration:** Promote collaboration between development, operations, and other stakeholders to ensure seamless integration and delivery • **Continuous Improvement:** Implement iterative development processes and continuous integration/continuous deployment (CI/CD) pipelines to drive efficiency and innovation	• **Proprietary Solutions:** Offer specialized solutions or assets tailored to meet specific client needs • **Supplier Accountability:** Solution/Asset consumed as a service, relieving clients of operational burdens • **Ecosystems:** Suited for matured partners working in service ecosystems
Sourcing scope	**Part of SDLC**	**Part of SDLC**	**E2E SDLC**	**E2E SDLC**

Sourcing of IT capabilities for products & platforms

Based on a 3-tier model of criticality and differentiating potential, product and platform capabilities should be insourced or outsourced for different lifecycle phases. Through this process, the DevOps organization integrates into a broader Vendor and Partner Ecosystem.

Application Tier	Plan	Code	Test	Release & Deploy	Operate & Monitor
Tier 1: Mission Critical Most critical operational and commercial systems that provide enterprise value	Insourced	Insourced (Exception: Hybrid)			
Tier 2 Specialized Systems which provide support for important business capabilities	Insourced	Hybrid			
Tier 3 Commodity Non-differentiating systems supporting non-core business capabilities	Insourced	Outsourced (Exception: Hybrid)			
Typical Roles	• Platform Manager • Enterprise / Domain Architects • Product Owner • CX / UX / UI Designer	• Cloud / Integration / Software Developer	• Cloud / Integration / Software Developer • Test Engineer	• Release Manager	• Operation Engineers • Support Analysts

Tier 1 and 2 applications generally **use in-house DevOps teams** for full SDLC management. Occasional outsourcing may occur to address specific expertise or capacity gaps. Additionally, **Tier 2 might leverage dedicated teams** for global setup and 24/7 support

Tier 3 applications are prime candidates for **outsourcing**, where an external DevOps team manages the entire SDLC, excluding planning. Development may remain in-house until the product or platform achieves stability

Contracting changes

Changes are needed across multiple aspects of the traditional model to enable agile ways of working through contracting

Supplier Performance Management changes

Components	Traditional Model		DevOps Model	
 ***Budget/Pricing***	**Fixed price / Time and Materials**	Time or deliverable based contracting paid as per payment schedule without link to outcomes or value delivered	**Value Pricing**	Incremental pricing with a story-point / shared-risk and reward approach, or a combination
 ***Scope***	**Contract-centred**	Contract includes detailed specifications and requirements	**Product-centred**	Contract focusses on outcomes and value delivered, with sufficient flexibility to accommodate devops ways of working in which the scope is set but the solution can evolve
 ***Relationship***	**Buying a specified product**	Statement of work serves as ultimate plan for the projects — detailed specifications, defined prices, firm deliverable deadlines	**Entering a Relationship**	Statement of work should define the expectations of the relationship. This can include pricing associated with a series of performance reviews, providing a "definition of done" and clarifying the roles of both parties
 ***Contract Management***	**Statement of Work Management**	SoW-management focused on terms of the contract: Are the correct resources allocated, hours accurately documented, invoices paid on time, has the work been completed etc.	**Entering a Relationship**	Continuous review of performance including outcomes and value delivered after each sprint in agile projects
 **Delivery of...**	**Outputs**	Delivery is focussed around specified deliverables (outputs), regardless of their associated outcomes	**Outcomes**	Delivery is focussed on outcomes to the business, not individual deliverables (outputs)
 ***Capacity***	**Fixed**	Fixed number of supplier resources; any changes in the resources required will require updates to the SoW	**Variable**	Supplier resources can be flexed as required under a single SoW, based on a pre-determined rate card

Architecture & Hosting Model

Architecture Design in DevOps

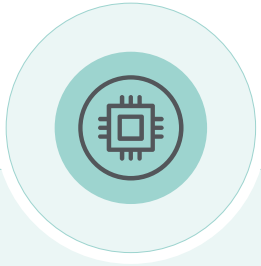
Realizing the full potential of DevOps requires a flexible, secure and agile technology stack, which often brings the need for architectural redesign

Hosting Models

Organizing your hosting models is essential to efficiently manage resources and enable seamless integration of development and operations processes, ensuring scalability and optimal utilization.

Organizations' Cloud native landscapes are getting increasingly complex, prompting organization to look for ways to optimize, such as:

- Hyper-converged infrastructures
- Containerized Infrastructure
- Serverless Infrastructure



Architecture Systems

Organizations have been moving away from monolith structures towards more modular systems allowing for greater scalability, flexibility, and maintainability of software applications.

Over the past years organizations have been adopting (micro)service-oriented architectures to achieve decoupling. One of the benefits of (Micro) Service-oriented architectures is that it creates an opportunity for complementary trends, such as:

- Event-driven architectures

Technology Stack

The combination of tools, technologies, and platforms (tech stack) is used to automate and streamline various stages of software enabling teams to achieve continuous integration and delivery while fostering collaboration and efficiency.

As the number of tools increases, so does the complexity of configuring an appropriate tech stack. Through careful selection and set up plans, organizations can effectively navigate multiple DevOps trends

Landscape Optimization on Cloud Native Hosting Models

In the current business environment, landscapes are getting increasingly complex, prompting organization to seek optimization strategies. Here are some primary examples how we observe organizations optimizing their landscapes depending on their selected hosting model

On-Prem — *Public, Private, Hybrid or Multicloud*

Hyper-Converged Infrastructure (HCI)

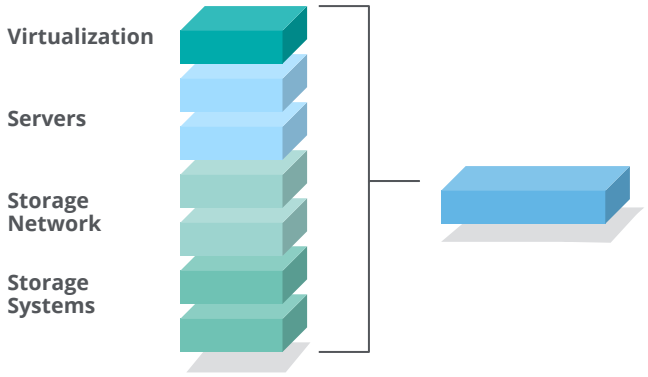
Hyper-converged infrastructure (HCI) combines storage, compute, and networking components into a single, integrated system managed through a unified interface. Organizations can benefit from cloud-native advantages within their on-prem environment. HCI's integrated approach aligns with the DevOps principle of breaking down silos between development and operations, fostering collaboration and accelerating the software development lifecycle

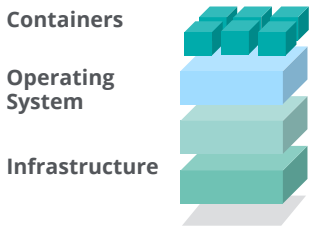
Containerized Infrastructure

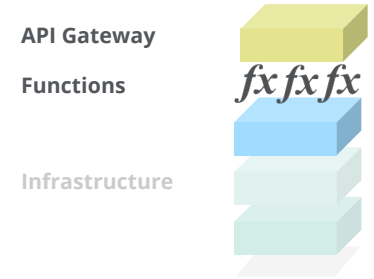
Containerization is a method of packaging, deploying, and running applications in lightweight, isolated environments called containers, allowing for consistency across different computing environments and easier scalability. It enhances agility, facilitates continuous integration and deployment, and enables more efficient resource utilization within software development and deployment workflows

Serverless Infrastructure

In a serverless model cloud providers dynamically manage the allocation and provisioning of resources, allowing developers to deploy code in the form of functions without needing to manage the underlying server infrastructure. The serverless model allows for automatic scaling, reduced operational overhead, and the ability to focus more on development and deployment tasks rather than managing server infrastructure







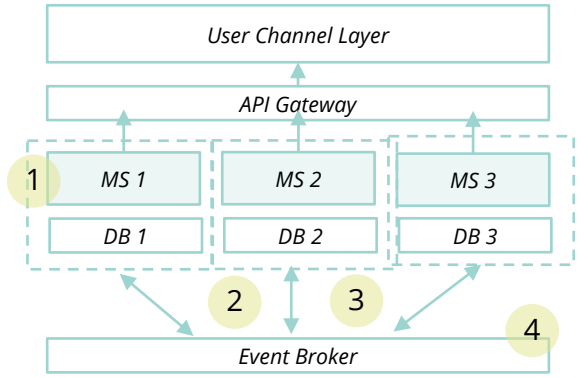
Event-Driven Architecture to Achieve further Decoupling

Next to microservices architecture, we also see the complementary trend: Event-Driven Architecture. Event-driven microservices communicate through events, helping to achieve further decoupling between services and enabling better scalability and responsiveness

Organizations have moved away from a traditional monolithic architecture towards a more modular service-oriented architecture model

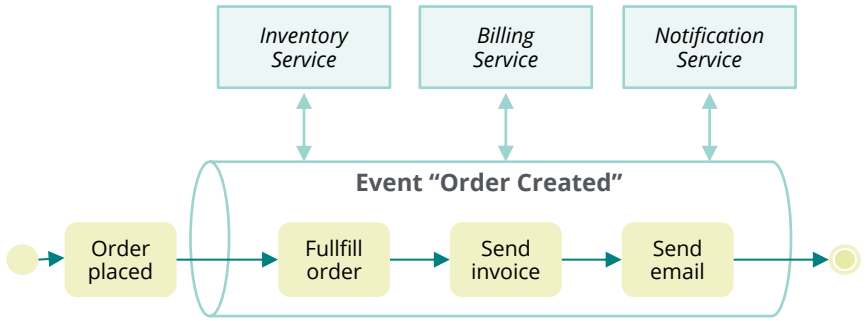
Nowadays we see that the modular models are allowing for additional & complementary architecture models, such as event-driven architecture

Event-Driven Microservices Architecture



1. Each Microservice is loosely coupled and independently deployable
2. Communication between microservices is asynchronous through events
3. Microservices react to events and perform their tasks accordingly without needing to know about the internal implementation of other services
4. If new functionalities need to be added, a new microservice can simply subscribe to the relevant events and handle it independently

Example flow in Event-Driven Microservices Architecture

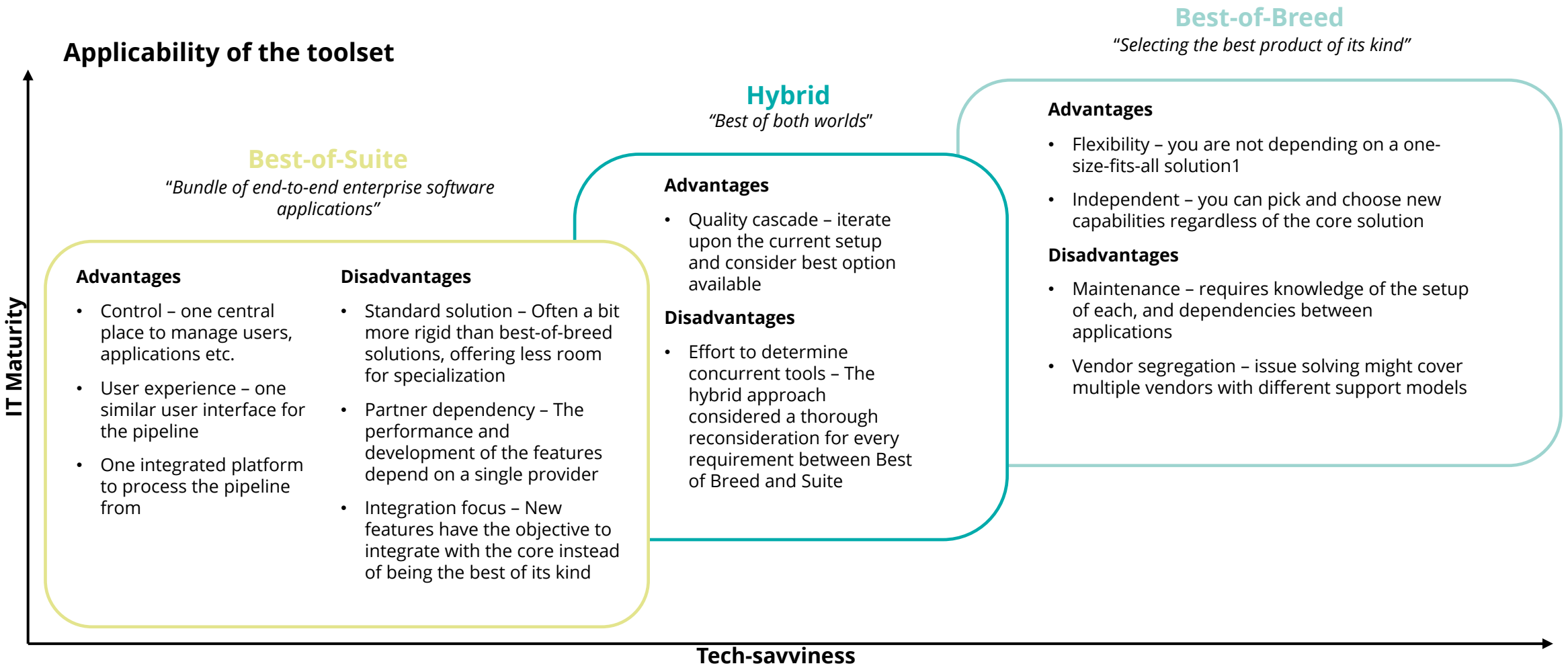


Why organizations are adopting

- Decoupled Deployment
- Improved Scalability
- Enhanced Fault Isolation
- Asynchronous communication
- Flexibility & Agility
- Streamlined Monitoring and Analytics
- Enhanced Collaboration

Patterns to set up your tech stack

Selecting the right set of tools (Best-of-Suite, Best-of-Breed or hybrid) for the tech stack depends heavily on IT maturity and tech-savviness of the organization



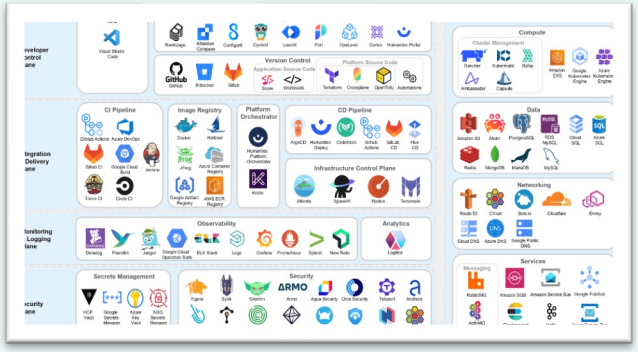
Plethora of tools to select your tech stack

A tech stack encompasses programming languages, frameworks, libraries, tools, and databases utilized in the development and deployment of software applications.

1. Select the set up pattern

Selecting the right set of tools (Best-of-Suite, Best-of-Breed or hybrid) for the tech stack depends heavily on IT maturity and tech-savviness of the organization

2. Select your tools

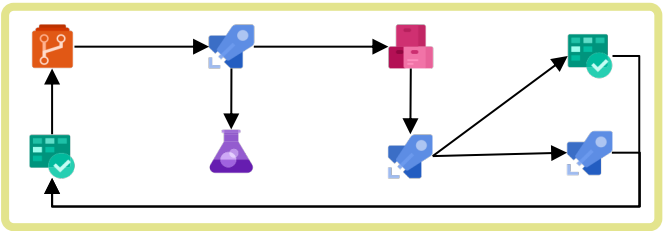


IDP Tooling Landscape displays the various tools across the software delivery lifecycle

3. Set up your Tech Stack

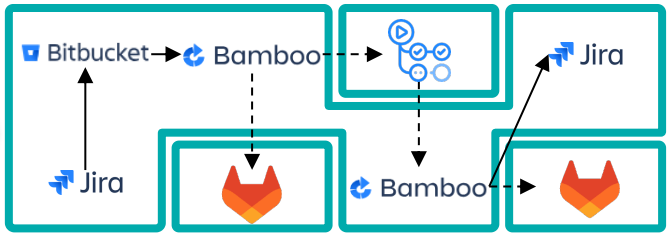
Zooming in on two of the planes in the IDP Tooling Landscape: **a CI Pipeline** & **CD Pipeline**. It can be built in various ways considering the desired tooling patterns, covering the same functional flow with different tools and integrations between them

Best-of-Suite



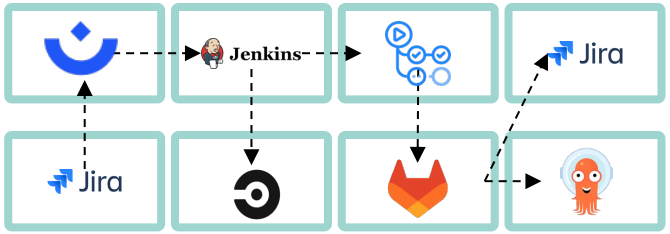
Azure DevOps covers the full extent of the CI/CD pipeline, with no external integration required

Hybrid



Only a few interfaces are required as Atlassian's suite covers the majority of required functionality

Best-of-Breed

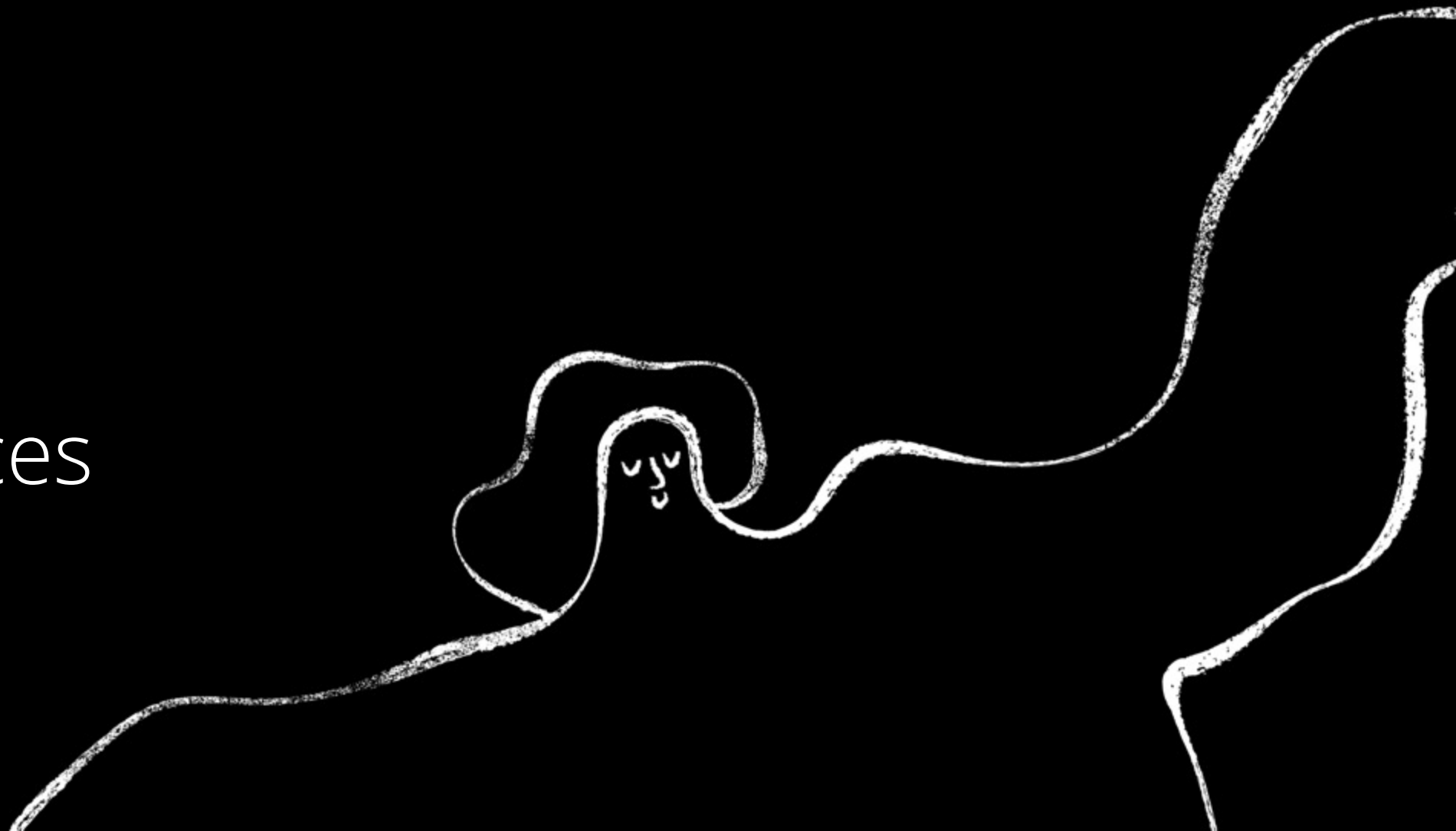


The pipeline orchestrator (Jenkins) becomes the central component to integrate all applications

Same suite, no interface

Interface between tools/suites

Appendices



DevOps Services and Propositions

DevOps Service Overview

Contact us!

Quekel, Marlies
Senior Consultant TS&T
mquekel@deloitte.nl

Our proposition has integrated services readily available as asset or accelerator to introduce you to DevOps, determine your DevOps appetite, or support your DevOps transformation

	INTRODUCTION TO DEVOPS & CI/CD			**ASSESSMENT OF DEV(SEC)OPS POTENTIAL**			**TRANSFORMATION OF ORGANIZATION**	
Objective	Gain understanding of DevOps and CI/CD concepts and realize their potential			Assess your DevOps maturity and (potential) scope of your DevOps transformation			Deliver complex tech enabled business transformations leveraging the DevOps delivery model	
Services	**DevOps Deep-Dive Training**	**DevOps Service Scoping**	**DevOps CIO Lab**	**Dev(Sec)Ops Quick Scan**	**Dev(Sec)Ops Mat. Assessment**	**OKR Dashboarding**	**DevOps Transformation Journey**	**Accelerate Tech Delivery**
Realized Benefits	Training to get an in-depth understanding across DevOps dimensions: • Organization & culture • Processes • CI/CD Technology, Architecture & Security • Operating Model & Transformations	The Service Scoping workshop aims to help an organization with setting their DevOps ambitions and defining their service entity; how should the organization organize itself and scope their DevOps service organization	The DevOps CIO Lab is organized for (new) CIO's that want to gain a better understanding of DevOps and CI/CD concepts and their potential. The CIO is triggered with the changing IT operating model and its effects on a DevOps organization	Quick scan into an organizations Software Delivery Lifecycle process and it's bottlenecks and dependencies. Deloitte provides a report-out on the observations and potential areas for improvements	Detailed assessment of DevOps and CI/CD capabilities in the existing organization and architecture using Deloitte's DevOps Maturity Assessment	Dashboard that visualizes the objective, key results and corresponding metrics of an organization with regards to their Software Delivery Lifecycle targets / ambitions. The dashboard is set-up to increase accountability and steer on continuous improvements	Implement / Mature / Scale the DevOps Delivery Model at an organization leveraging a full-scale Deloitte service driving and coordinating the complex tech enabled business transformation (with a focus on DevOps)	Implement / Use the DevOps delivery model to drive large tech enabled transformations (e.g. Gen AI, Operate to Innovate)
Investment	1 day	1 day	1 day	½ day	6 - 8 weeks	3 weeks	3 – 12 months	Depending on program

CLIENT CASE

DevOps Training Curriculum

Global Delivery Service

SITUATION

- The client seeks to foster a productive and innovative engineering culture through the creation of strong DevOps teams, where processes are streamlined, feedback is efficient and effective, and team members feel empowered
- The client aims to shorten the time to market by enabling frequent releases (several times an hour), reducing the time from feature review to production deployment by 50%, establishing a robust testing strategy, and holding teams accountable on agreed objectives and key results (OKRs)

SOLUTION

- Conducted a current state maturity assessment to identify Dev(Sec)Ops maturity levels, identified bottlenecks through qualitative interviews, and subsequently created and implemented an implementation roadmap and backlog
- Created an Objectives and Key Results (OKR) framework and an OKR dashboard with data insights for the set objective, key results and corresponding metrics
- Developed a training curriculum, combining business and development-oriented trainings

IMPACT

- An implementation roadmap and backlog with numerous improvement items to increase the maturity levels and accelerate the delivery service's Software Delivery Lifecycle
- Data insights through the OKR dashboard to increase accountability and to steer on continuous improvements
- High attendance during the trainings enhancing the organizational understanding of Dev(Sec)Ops, culture and accelerating the Software Delivery Lifecycle

CLIENT CASE

DevOps Service Scoping Assessment

Medical Technology & Service Provider

SITUATION

- The client wants to mature and scale their DevOps capability in general and accelerate global platform delivery with Salesforce as launching technology
- In addition, they want to perform a reality check with Deloitte on their DevOps plans, ambitions and potential next steps for the future

SOLUTION

- Validate DevOps Ambition to challenge and (re)define their DevOps mission, customers, services and capabilities
- Based on the organization's DevOps ambition, identify the required capabilities to enable end-to-end ownership of products and services
- Assess the Software Delivery Lifecycle for bottlenecks and establish a wish list on how to improve their SDLC

IMPACT

- Validating the DevOps ambition helped the client in reassessing and refining its DevOps mission. It encourages a critical evaluation of current practices and identify gaps in capabilities
- Identify the bottlenecks in the Software Delivery Lifecycle (SDLC) and provides a roadmap for improvement, enhancing overall delivery efficiency
- Create a cohesive DevOps strategy that is aligned with company goals
- The focus on customers in the DevOps ambition and the end-to-end ownership of services ensure a service-oriented approach, leading to products that better meet customer needs

CLIENT CASE

Dev(Sec)Ops Quick Scan

National Safety Organization

SITUATION

- The client has the ambition to deliver a CI/CD platform that enables developers to deliver software faster and more efficiently by using secure and standardized products and services.
- The Platform team is responsible for configuring and maintaining development tools and services. This allows DevOps teams to focus on work that delivers value
- The clients wants to deliver secure software faster and easier

SOLUTION

- Conducted a Dev(Sec)Ops Quick Scan to identify Dev(Sec)Ops areas for improvement in products and services and in the production process outside of the scope of the platform team
- Place these areas for improvement in a broader perspective using a reference operating model

IMPACT

- A report-out that elaborates on the areas for improvement in various domains of the Product & Platform operating model that are preventing the organization from achieving its Dev(Sec)Ops ambition to develop secure software faster, simpler and together

CLIENT CASE

Dev(Sec)Ops Maturity Assessment and recommendations for further growth

Global Delivery Service

SITUATION

- The client seeks to foster a productive and innovative engineering culture through the creation of strong DevOps teams, where processes are streamlined, feedback is efficient and effective, and team members feel empowered
- The client aims to shorten the time to market by enabling frequent releases (several times an hour), reducing the time from feature review to production deployment by 50%, establishing a robust testing strategy, and holding teams accountable on agreed objectives and key results (OKRs)

SOLUTION

- Conducted a current state maturity assessment to identify Dev(Sec)Ops maturity levels, identified bottlenecks through qualitative interviews, and subsequently created and implemented an implementation roadmap and backlog
- Created an Objectives and Key Results (OKR) framework and an OKR dashboard with data insights for the set objective, key results and corresponding metrics
- Developed a training curriculum, combining business and development-oriented trainings

IMPACT

- An implementation roadmap and backlog with numerous improvement items to increase the maturity levels and accelerate the delivery service's Software Delivery Lifecycle
- Data insights through the OKR dashboard to increase accountability and to steer on continuous improvements
- High attendance during the trainings enhancing the organizational understanding of Dev(Sec)Ops, culture and accelerating the Software Delivery Lifecycle

CLIENT CASE

DevOps at a Global Location Platform Provider

CLM value delivery improvements based on DevOps principles

SITUATION

- The client started the implementation of their Customer Lifecyle Management (CLM) platform. This platform is key in supporting its growth strategy and enabling roll-outs of new products and services to the (global) market
- The client wants to **accelerate their value delivery** and **make it measurable** focusing on quality, predictability, and traceability to support its growth strategy

SOLUTION

- **Restructure Test Management** by coordinating test efforts & shift left to improve product quality and increase speed (lead time for change)
- **Streamline Release Management** and **deployment flow** to reduce execution time (lead time for change), improve business agility and reduce deployment errors (change fail rate)
- **Setup of OKR dashboard** with outcome-based (value) metrics to measure the quality of service of the CLM platform

IMPACT

- The DevOps based improvements for the CLM value delivery increased delivery speed**, improved delivery quality** and **improved business agility**, contributing to overall business value.
  - **Improved allocation of team capacity** from 43% towards 60% development time
  - **Reduced lead time for change** from months to weeks
  - **Stabilized CLM systems** in terms of incidents and reduced P1 support requests

CLIENT CASE

Technology Driven Transformation in Operations Domain

Global Logistics Provider

SITUATION

- The client needs the capability to regularly update their systems across ~180 physical locations all over the world
- Client lacks insight in their system and operational performance and needs to align with multiple service partners in the development process

SOLUTION

- Implement a new operating model (incl. DevOps way-of-working) in the organization by re-structuring existing teams and processes
- Introduce new (CI/CD) technology to streamline the delivery of new upgrades across the landscape
- Establish ‘snowball effect’ in the organization to foster adoption of new way-of-working and new technology
- Integration of multiple service partners in the ecosystem
- Execute a rollout plan to force standardization and ease of updating and maintenance

IMPACT

- New DevOps teams are end-to-end responsible for the full lifecycle of a service, thereby reducing handover times between Development and Operations
- From lacking insight to full transparency of both system and operational performance
- Updates are delivered simultaneously to multiple locations, saving a significant amount of resources that can be used for other tasks

DevOps Practitioners

DevOps Practitioners

A big thanks to the authors of this Point of View

Mark Maijs

Sponsoring Director & QA

Director
Technology, Strategy & Transformation
Deloitte Consulting
+31 6 83 89 02 14
mmaijs@deloitte.nl

Donald Pondman

Sourcing Model

Senior Manager
Technology, Strategy & Transformation
Deloitte Consulting
+31 6 83 33 02 56
dpondman@deloitte.nl

Marlies Quekel

Capability Lead & Organizational Model

Senior Consultant
Technology, Strategy & Transformation
Deloitte Consulting
+31 6 50 07 02 16
mquekel@deloitte.nl

Thomas Grimmeissen

Architecture & Hosting Model

Senior Consultant
Technology, Strategy & Transformation
Deloitte Consulting
+31 6 50 15 15 99
tgrimmeissen@deloitte.nl

Lianne Veenhuis

Architecture & Hosting Model

Consultant
Technology, Strategy & Transformation
Deloitte Consulting
+31 6 15 69 04 09
lveenhuis@deloitte.nl

Koen Meijer

Development Model

Consultant
Technology, Strategy & Transformation
Deloitte Consulting
+31 6 29 67 37 96
kmeijer@deloitte.nl

Diana Zubchenko

Development Model

Analyst
Technology, Strategy & Transformation
Deloitte Consulting
+31 6 24 28 51 80
dzubchenko@deloitte.nl

Wouter Eversdijk

Sourcing Model

Analyst
Technology, Strategy & Transformation
Deloitte Consulting
+31 6 31 02 40 41
weversdijk@deloitte.nl

Ting Fung Lee

Deliver & General Support

Analyst
Technology, Strategy & Transformation
Deloitte Consulting
+31 6 21 89 31 44
tinglee@deloitte.nl

Sander Boot

Deliver & General Support

Analyst
Technology, Strategy & Transformation
Deloitte Consulting
+31 6 50 01 80 99
sanboot@deloitte.nl

Additional contributors

An additional big thanks to all our colleagues that contributed to this Point of View

Andries van DIjk
Specialist Director

Technology, Strategy & Transformation
Deloitte Consulting

Lewis Young
Director

Technology, Strategy & Transformation
Deloitte Consulting

Jonne Kocken
Manager

Technology, Strategy & Transformation
Deloitte Consulting


Thank you

Deloitte.