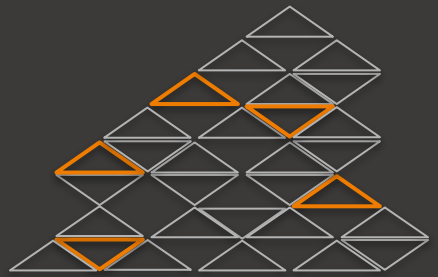
WIRES UNCROSSED

Understanding your delivery system

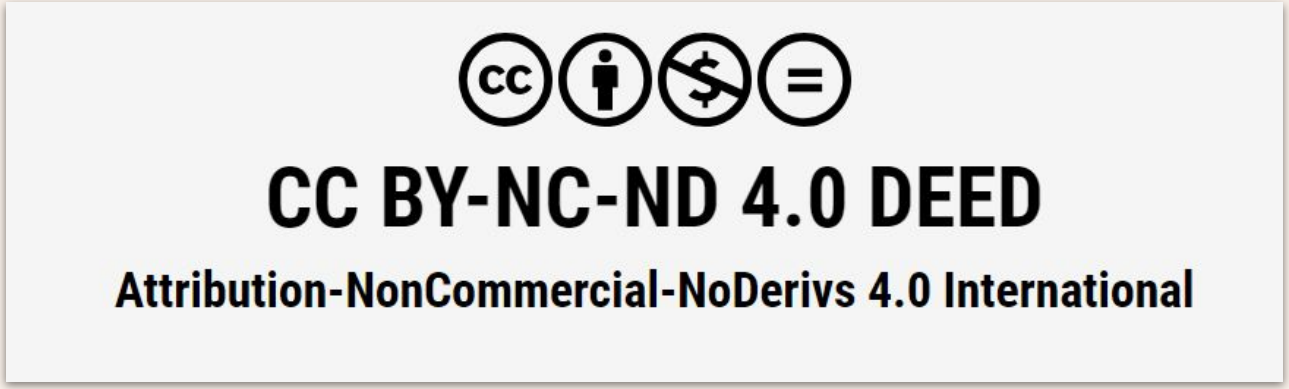


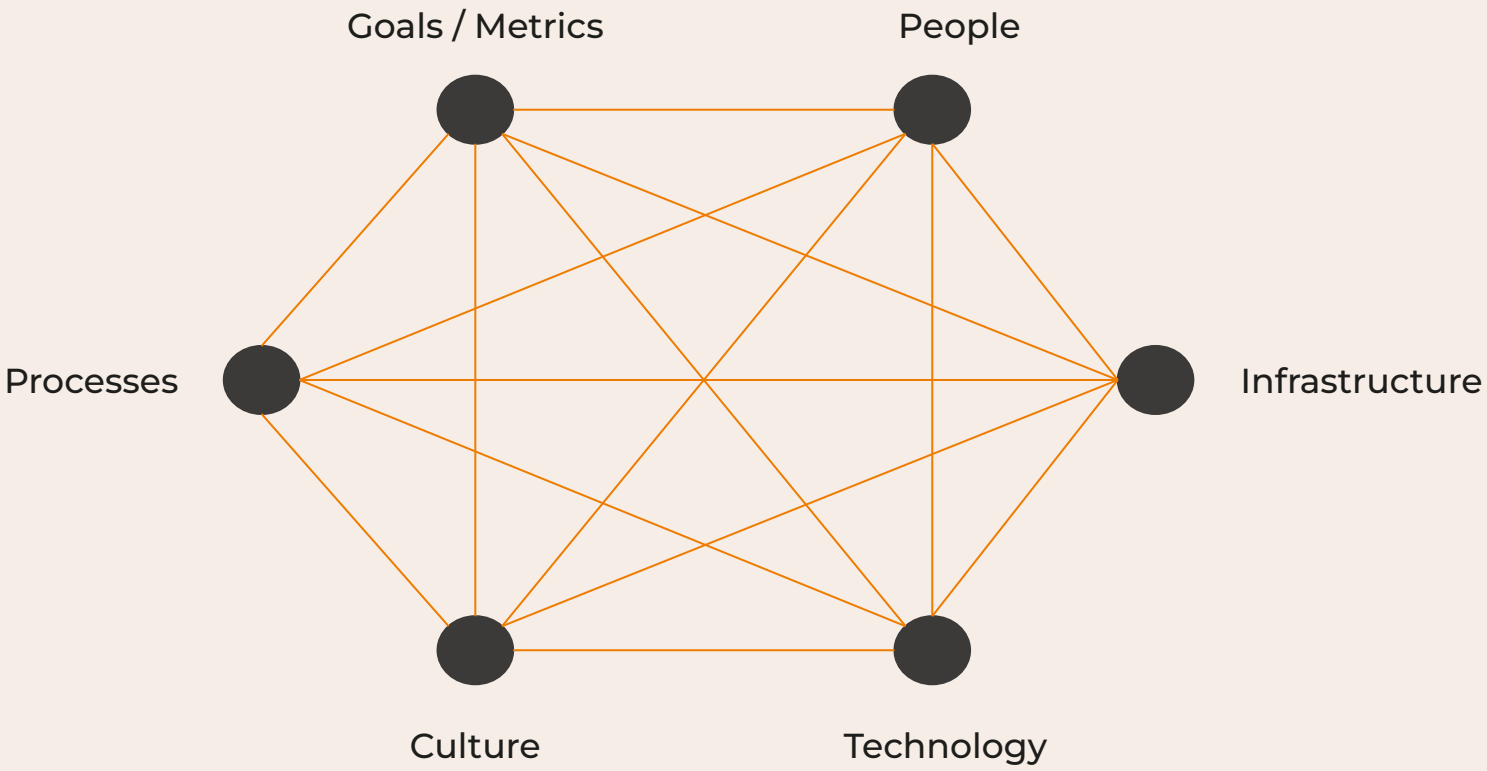
Auckland Software Leaders Group
Jan 2024

myles-henaghan

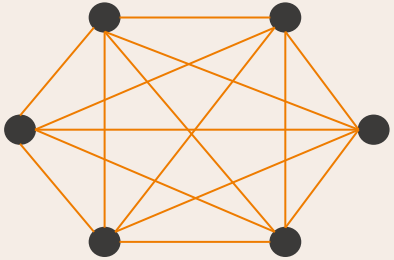
Free to share

Our contribution to
raising the standard of software delivery in New Zealand




Goals / Metrics
People
Processes
Infrastructure
Culture
Technology

01 Continuous Delivery



Measured by

Software Delivery Performance

DORA
DEVOPS RESEARCH & ASSESSMENT

Metrics & Benchmarks

Throughput
Quality
Reliability

Predicts

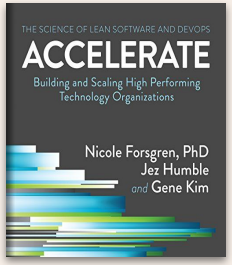
Organisational Performance

***2x** more likely to exceed their goals*

Quality of services provided
Number of customers
Customer satisfaction
Quantity of products
Operating efficiency
Market share
Productivity
Profitability

Limits

- Flow
- Sustainability
- Effective Ownership
- Managed Work
- Basic Needs



Harvard Business Review

McKinsey & Company

9 years
12 industries
36,000 professionals

aws

INTUIT

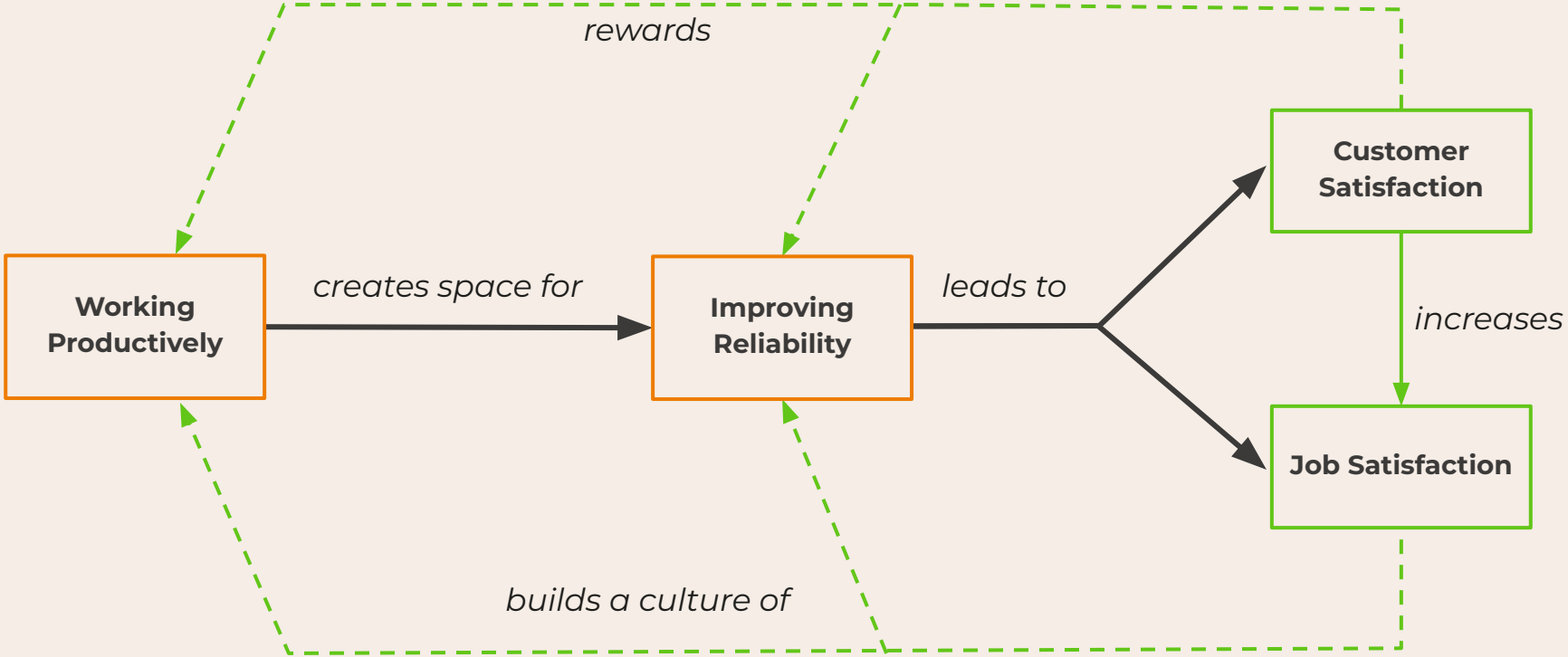
xero

Source: dora.dev/research, wue.co.nz/model

Hierarchy of Engineering Needs©

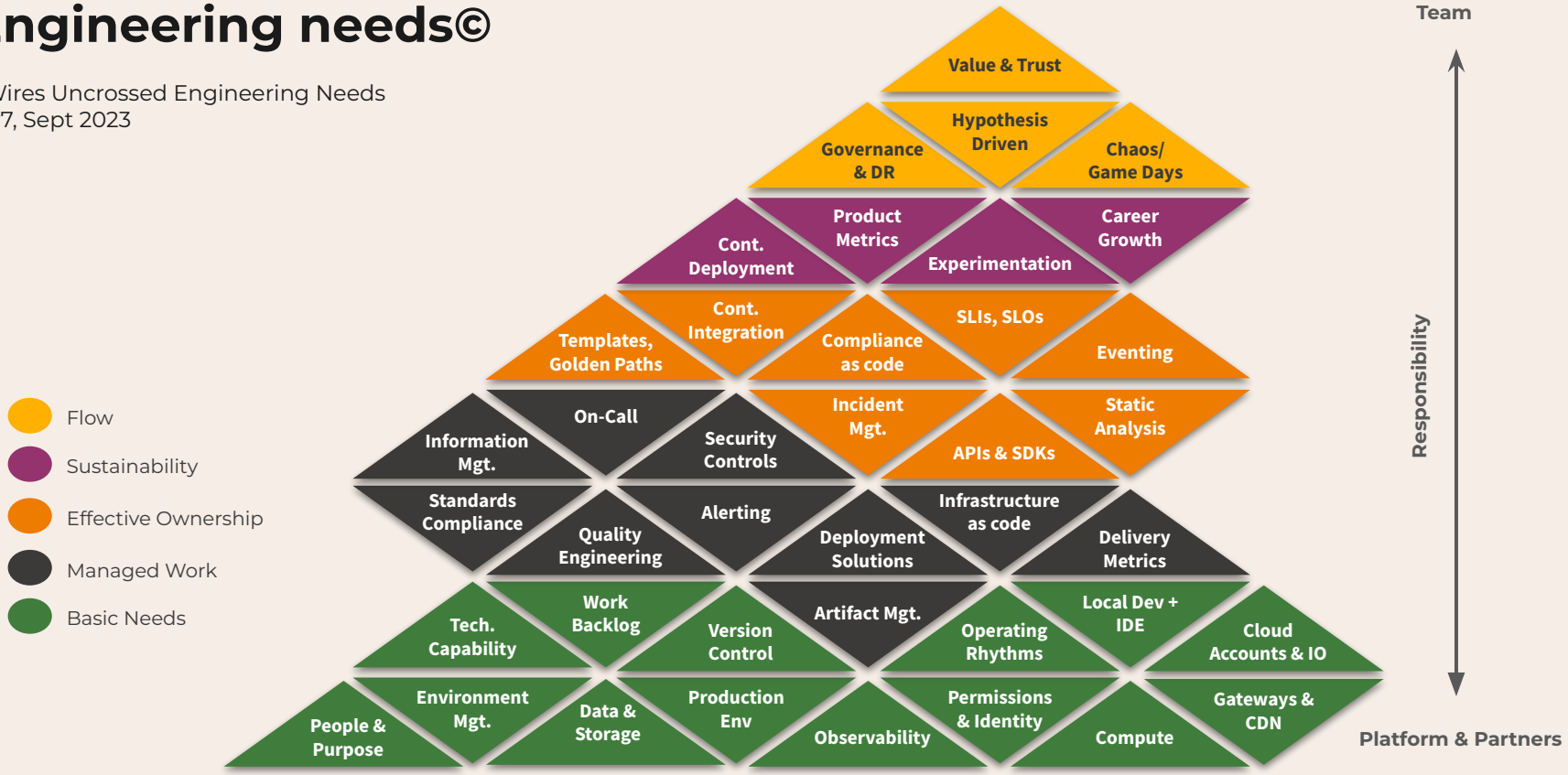
WIRES UNCROSSED

Positive Maturity Cycle



Engineering needs©

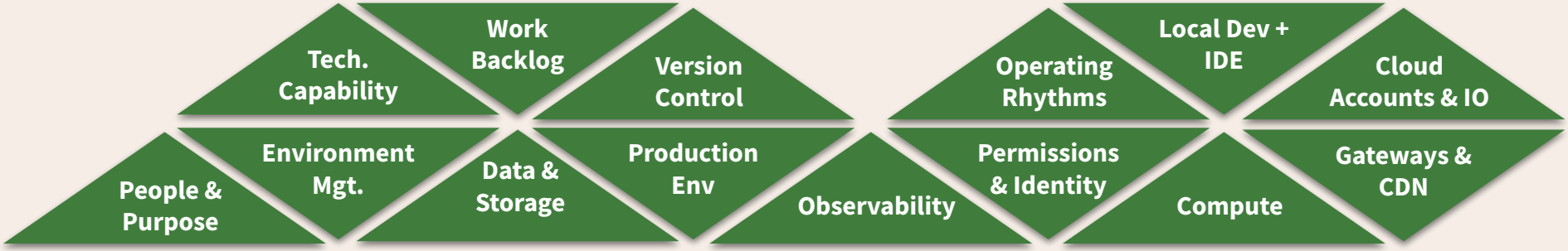
Wires Uncrossed Engineering Needs
v7, Sept 2023



Engineering needs©

1. Basic Needs

What does any team, regardless of age and stage, need to build an application and deploy changes to it?



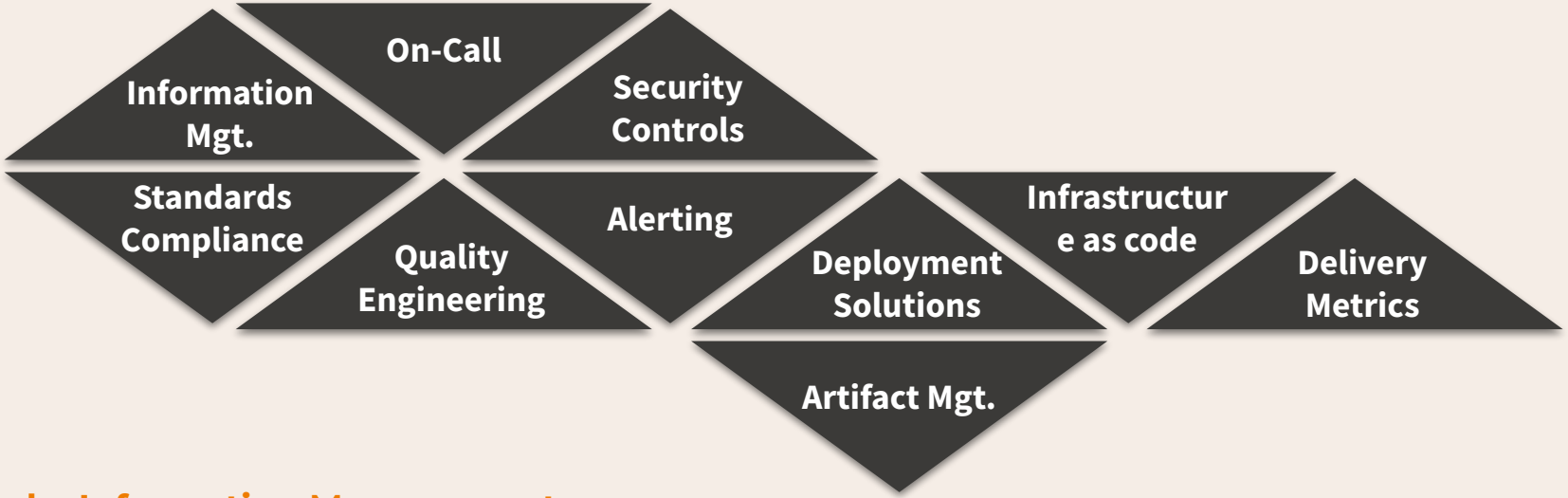
Example: Environment Management

I can easily test and debug changes in a prod-like environment.
An environment for my change.

Engineering needs©

2. Managed Work

What does the team need to make work repeatable and have controls to verify the quality and efficiency of new work?



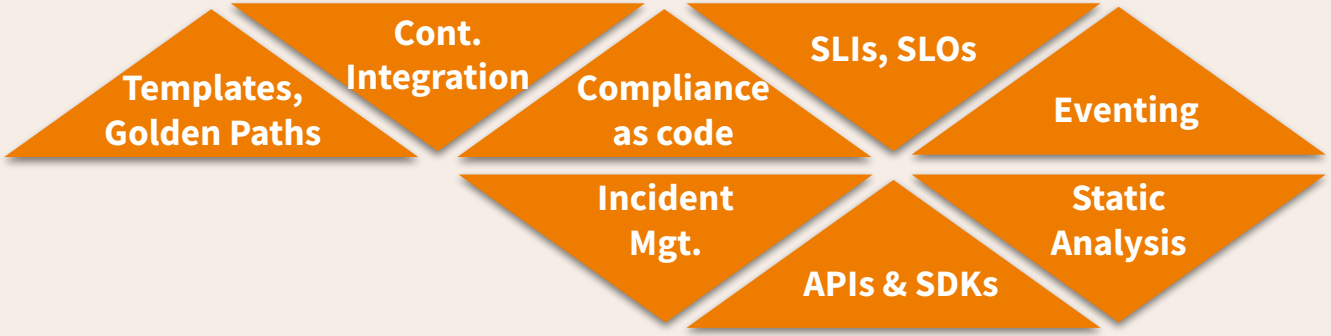
Example: Information Management

Team and service level Information is easy to self-discover and is trusted.
Auditing compliance and runtime dependencies is routine and self-service

Engineering needs©

3. Effective Ownership

What does the team need to effectively own and operate services already in production, regardless of new development work on those services?



Example:Templates, Accelerators/Golden Paths

We can consistently create new projects and deploy a 'hello world' version to production in under [3] hours.
Templated projects, builds, and IaC complies fully with our engineering standards.

Engineering needs©

4. Sustainability

What does the team need to grow and mature on a yearly level?

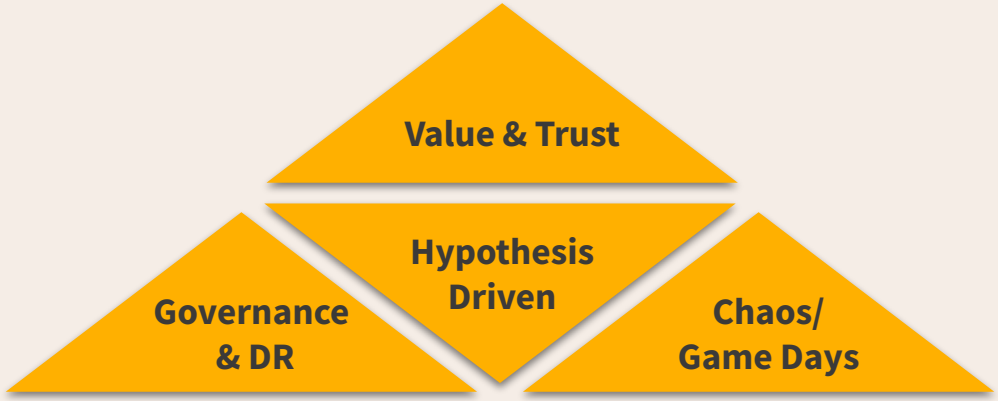
Example: Experimentation

The team can run multiple experiments in parallel.
Significant system changes can be safely tested in production.

Engineering needs©

5. Flow

Maturity here is marked by being able to work productively over long periods while maintaining, if not increasing, customer and shareholder trust.

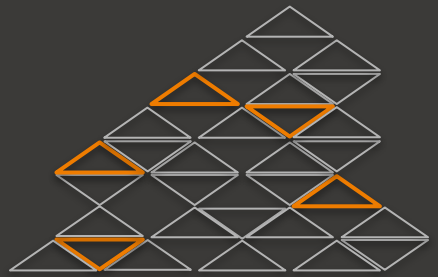


Example: Governance

Annual audits do not interrupt the team's regular work.
We proactively test our disaster recovery procedures.
It is easy to find service-level information on security, cost and privacy status.

WIRES UNCROSSED

Appendix



BASIC NEEDS

Need Name	Definition / The Ability to...
Production Environments	Trusted combined state of **configuration**, deployed **artifact(s)**, **infrastructure**, **runtimes, dependencies and access control.** The Prod ENV ensures a seamless user experience and customer satisfaction by providing a reliable software deployment platform. It is optimized & managed for high performance, scalability, and availability. The team can quickly access production log files and data to assist with troubleshooting issues.
Environment Management	Ability to test and debug changes in a **prod-like environment**. Infrastructure-as-code, data sets/data generation, dependency management & configuration, **Mocks, data generation**, **short-lived environments**. Real dependencies, similar/same technically to production. Composable environments. Setting up and managing different environments for my project is **efficient and cheap**. An environment for my change.
Observability	The ability to debug**, troubleshoot and monitor normal and abnormal system behaviour**. Curated patterns and tooling for the collection, storage, querying and aggregation of system data. It provides comprehensive visibility into the system's health, performance, and logs, allowing for timely detection and resolution of issues. An opinionated **operational health dashboard**.
Technical capability	**Most people can pick up most work on the team**. The team's technical capabilities align well with the requirements of the project. Team members can complete daily activities confidently. They have a solid understanding of the technologies, frameworks, and tools that are utilised, allowing them to design and develop robust solutions efficiently. They **regularly pair-program to help with productivity** and building experience.

BASIC NEEDS

Need Name	Definition / The Ability to...
Operating Rhythms	The routines used by the team to manage **delivery and operations**. **Planning** on a daily, weekly and quarterly basis. **Familiar and effective ceremonies**. **Open communication** channels enable timely updates, issue resolution, and knowledge sharing, promoting a cohesive team dynamic and ensuring everyone works towards a common purpose. Regular team meetings, stand-ups, and planning sessions foster a **shared understanding of priorities and facilitate the coordination** of tasks. **Achievements are celebrated, learnings are sought**, and the team is empowered to **balance capacity investment** across new feature delivery, maintaining quality of service and improving efficiencies. There is Organisation level visibility on past and planned investments across new feature delivery, maintaining quality of service and improving productivity.
People & Purpose	**Enough team members to meet the ownership and delivery expectations** on them. I feel connected to the purpose and goals of the project I am working on. The project's vision and objectives are clearly communicated, creating a sense of shared purpose and motivation among team members.
Compute	The allocated computational resources sufficiently meet the demands of my project, ensuring optimal performance and scalability without hindering productivity. **Self-service compute solutions available** for the teams across VMs, containers, FaaS and edge workloads with opinionated sensible defaults. Clear options to support minimum capacity, elastic scaling, burst scaling and serverless. **Guidance and defaults in place** for performance, reliability and availability of compute platforms. Compute **fleets are kept young** and disposable theough routine fleet management. Running instances, configured containers younger than two weeks to improve **patching compliance and avoid stateful dependancy** Proactive **management of minimum OS and runtime versions** to avoid running non supported versions or dependancies in production.

BASIC NEEDS

Need Name	Definition / The Ability to...
Data	Opinionated **self-service data solutions are available** for the teams across SQL, NoSQL, Blob/Bucket and Caching. Guidance and defaults are in place for **performance, reliability, and data durability**. The **mean-time-to-restore is known** for its primary services. Sharding or partitioning policies support the scale and regional context of the services. The team can **independently manage Data migrations and schema changes**. There is up-to-date **documentation of schemas** and associated privacy or security classifications.
Work Backlog	The work backlog for my project is well-organized and prioritized. It reflects a clear understanding of project requirements and aligns with strategic goals. The backlog items are **appropriately sized**, ensuring efficient planning and task allocation. **Prioritization reflects a balance** of business, value, customer value, team productivity and service performance. With a well-maintained backlog, the project **team can effectively plan, estimate, and track progress**, ensuring a steady flow of value delivery.
Code Version Control	New **repositories, teams and projects and permissions are created in a consistent way**. Clear guidance on branching strategies implemented at organisation level. Easy to track changes, support concurrent development and rollback changes. Clear association between repository, team and any deployed assets and overall project status. There is guidance and examples on creation of repositories for internal only, internal + trusted partners, and open source use cases. The **removal of permissions for ex employees and contractors is automated**

BASIC NEEDS

Need Name	Definition / The Ability to...
Cloud Accounts & IO	Teams can create new cloud accounts to provision infrastructure and services related to a common domain. **Fast automated creation and deletion of cloud accounts** with consistent and compliant Identity and Access Management inherited from a centralised control pane. Accounts created with **consistent virtual networks** for deployed assets and across accounts
Local Dev + IDE	Provides a s**mooth and productive coding experience**, with the necessary tools, libraries, frameworks and associated licenses are readily available. **Highly automated opinionated setup** process gives developers a **consistent baseline** setup to support build, test and run activities for the primary services they work on. Service-specific local setup (e.g. install dependencies, mocks, test data). Related to environment management. **IDE is setup consistent for new team members**. Agreed coding conventions are codified in 'dot files' at machine or repository level.
Gateways & Web Delivery	The api gateways and web delivery mechanisms (e.g. CDNs) employed in my project ensure **efficient and reliable communication between various services for internal and external traffic**. These gateways provide a **unified entry point, handle routing and load balancing**, and ensure secure and optimized data transmission. With robust web delivery mechanisms, the project team can achieve high availability, scalability, and fault tolerance, delivering a seamless user experience and enabling effective integration with external systems. Examples: API GW, Azure API Management, Cloudflare, Fastly, Akamai, Cloudfront, Kong, NGINX
Identity & Permissions	The ability to create and manage accounts, credentials, roles and permissions for end users of the product or service the team is responsible for. Support for social logins, single sign-on, account verification and suspension. Multi-factor authentication flows. The ability to script user generation or resets to help with integration test automation.

MANAGED WORK

Need Name	Definition / The Ability to...
Quality Engineering	The tools and practices teams use to validate changes early and throughout the development lifecycle continuously. The management of quality ensures controls are trusted and highly automated. It is familiar and routine to maintain unit, contract and integration testing. Core workflows for each service are continually tested in production.
On-call	The on-call process is well-structured and familiar to all team members. The appropriate people are available and contacted promptly to respond to alerts or incidents. Escalation paths are known and used. Expectations with responders are clear and respected for out-of-hours and holiday periods. Handovers are acknowledged and include a debrief on any closed or ongoing incidents.
Standards Compliance	The combined organisation obligations (e.g. audit, regulations) and agreed best practices are documented, transparent and respected by teams. There is a process to propose and accept changes to established standards. The level of compliance with standards is visible across the organisation.
Artifact Management	There is a standard solution for hosting, managing and distributing binaries and artefacts across the organization. Artefacts include versioned binaries, library feeds (e.g. npm, NuGet, Maven), container images and non-sensitive configuration files. The solution supports high availability and durability. The team publishes logs and test artefacts according to their audit requirements.
Alerting	The team has implemented an intelligent alerting mechanism that monitors key metrics in agreed thresholds across request rates, errors and latency. Scheduled tests regularly check the primary service flows or user actions for availability. When detecting abnormal system behaviour, the team is alerted before the customer or stakeholder reports an issue.

MANAGED WORK

Need Name	Definition / The Ability to...
Deployment Solutions	**For each component type** the team supports, there is a **standard workflow to build, test, publish artefacts and deploy** new versions. Build pipelines are highly automated, including promoting successful builds across environments. The deployment solution also orchestrates supporting automation for infrastructure-as-code (IaC), configuration and secrets management. Central teams publish and maintain common build steps inherited across all builds (e.g., publishing, tagging, static analysis). Build capacity and speed is high enough to keep wait times under [10] minutes.
Security Controls	Tests and security controls are followed throughout the development lifecycle (design, build and run) to help identify potential security issues. The security review process does not slow down the development process for the primary systems the team works on. The team understands their security posture and manages new issues in consultation with experts or a supporting team.
Infrastructure as Code	Infrastructure configurations are defined in code, enabling version control, reproducibility, and automated provisioning. The team manages workload infrastructure creation, configuration, maintenance and deletion through source code change control. Cloud or data centre portals are used for diagnostics and learning but not for applying changes.
Information Management	The team maintains a consistent repository of information on component ownership, support routines, team composition and solution architecture. The information is accessible across the organization to support efficient onboarding, communication and incident management across teams. All deployed infrastructure is identifiable to the component and associated team level.
Delivery Metrics	Key metrics, such as lead time, cycle time, and throughput, are regularly monitored and reviewed. These metrics provide insights into delivery efficiency, predictability, and capacity to deliver value. The team can identify bottlenecks by analyzing delivery metrics, optimizing processes, and continuously making data-driven decisions to improve delivery performance.

EFFECTIVE OWNERSHIP

Need Name	Definition / The Ability to...
Templates & Golden Paths	A comprehensive set of templates and accelerators that expedite development and ensure consistent implementation of best practices across our common component types. Component / Project templates implement blueprints in preferred architectures, coding conventions and engineering standards. Accelerators combine templates into workflows to scaffold, build, test and deploy new services in hours.
Static Analysis	The team integrates static analysis tools that effectively identify code issues, potential vulnerabilities, and maintain code quality standards. These tools automatically analyze code, check for common programming errors, and enforce coding conventions. Tools run locally and as part of automated builds to detect and resolve new issues as early as possible. Data on software composition & supply chain is available centrally to manage licensing and security vulnerability risk
Continuous Integration	All code changes are continually **merged to the main branch (trunk/master) several times a day**. Test-driven development (TDD) and behavior-driven development (BDD) practices are diligently followed by all team members. TDD ensures that tests are written before the code, promoting a thorough understanding of requirements and facilitating comprehensive test coverage. BDD focuses on defining system behavior through scenarios and specifications, improving collaboration between stakeholders and developers. The team **may also practice Trunk-based development** (TBD) or non-blocking pull requests where Engineers work on a single branch as much as possible to encourage continuous integration. Pushing changes directly to trunk or main is made safe with mature TDD and quality engineering practices.
Eventing	The ability to coordinate functionality across distributed systems and services using events. Standard patterns, SDKs and tooling to publish, store, subscribe and consume events across APIs, workers, Web and mobile clients. Schema validation is in place to block foreign or malformed events.

EFFECTIVE OWNERSHIP

Need Name	Definition / The Ability to...
SLO/SLIs	There are clear service level objectives (SLOs) and service level indicators (SLIs) that help measure performance, ensuring the product consistently meets the defined quality and reliability standards. These SLOs and SLIs define synthetic tests of primary workflows and key metrics, such as response time, availability, and error rates, enabling the team to monitor and continuously improve the product's performance against established benchmarks and commitments made to customers.
Templates & Golden Paths	A comprehensive set of templates and accelerators that expedite development and ensure consistent implementation of best practices across our common component types. Component / Project templates implement blueprints in preferred architectures, coding conventions and engineering standards. Accelerators combine templates into workflows to scaffold, build, test and deploy new services in hours.
Compliance as Code	My team incorporates compliance requirements as code, automating compliance checks and ensuring adherence to regulatory standards. Compliance rules and checks are codified, allowing for automated enforcement during the development and deployment process. By treating compliance as code, the team can efficiently implement and maintain compliance controls, reducing human error, enhancing auditability, and minimizing the time and effort required to maintain regulatory compliance.
API & SDK(s)	The team provides well-designed APIs and software development kits (SDKs) that facilitate integration and extensibility. The APIs offer clear documentation, well-defined contracts, and consistent interfaces, enabling seamless interaction with the product's services and functionalities. The SDKs provide comprehensive tooling, libraries, and code examples, simplifying the development process for external and internal consumers. Internal publishers support fake or mock implementations of their APIs to help consuming teams automate testing.
Incident Management	The incident management process on my product is robust and well-structured, enabling swift response and minimizing the impact of incidents. The team follows established incident response procedures, including incident identification, communication, prioritization, and resolution. Clear roles and responsibilities are defined, ensuring effective coordination and collaboration during incident handling.

SUSTAINABILITY

Need Name	Definition / The Ability to...
Career Growth	My product team provides ample opportunities for career growth and professional development. There is a mix of junior and senior talent on the team. Everyday pairing and coaching on specialised areas avoid key person dependencies. The team and company encourage continuous learning, supports skill enhancement, and provides access to training resources and mentorship programs. Regular performance evaluations and constructive feedback help identify areas for improvement and define career paths.
Product Metrics	The team effectively captures and analyzes product performance metrics, enabling data-driven decision-making and continuous product improvement. The team can independently form and test a new hypothesis using standard user event capture, aggregation and visualization tools. User events and flows are well-documented and accessible to all teams. Event processing filters activity from automated testing.
Experimentation	Team actively promotes a culture of experimentation to support both hypothesis-driven development and technical risk management. The team conducts controlled experiments, A/B tests, and user research to validate assumptions, gather feedback, and inform decision-making. The team can run multiple experiments in parallel. Significant system changes can be safely tested in production.
Continuous Deployment	Building on continuous Integration, all successfully builds are automatically promoted and deployed to production. The team is likely practicing Trunk-based development (TBD) or non-blocking pull requests where Engineers work on a single branch as much as possible to encourage continuous integration. Pushing changes directly to trunk or main is made safe with mature TDD and quality engineering practices. Changes are implemented with the intention it will be deployed directly to production. High degree of release controls (e.g. feature flags, experiments) ensure it's safe to deploy partially implemented features. High maturity of SLO and alerting to detect and revert problem changes.

FLOW

Need Name	Definition / The Ability to...
Governance & DR	The team enjoys adequate governance controls and disaster recovery (DR) measures for their services and operations. Wherever possible, these controls are automated into the Engineering activities. The team proactively tests their DR procedures (e.g., backup, recovery, replication and failover) to meet their business continuity commitments. Annual audits do not interrupt the team's regular work.
Hypothesis Driven	My product embraces a hypothesis-driven approach, where experiments and data analysis systematically test and validate assumptions. The team formulates clear hypotheses, defines success criteria, and designs experiments to gather data and insights. By continuously testing and refining ideas, the team can make informed decisions, prioritize initiatives, and drive innovation, creating more customer value.
Chaos / Game Days	My product team regularly conducts chaos/game days to proactively test and validate the system's resilience and readiness for unexpected scenarios. During chaos/game days, the team simulates real-world failure scenarios, such as service outages or network disruptions, to identify vulnerabilities, assess the system's response, and fine-tune incident response processes. By regularly subjecting the product to controlled chaos, the team can strengthen the system's robustness, improve incident response capabilities, and enhance overall system reliability.
Value & Trust, Mastery	Fostering a culture of delivering value and building trust, the team prioritizes customer needs and consistently delivers reliable outcomes. Simultaneously, they strive for mastery through continuous learning and professional development, ensuring expertise in their respective fields. This combination drives innovation, collaboration, and excellence, exceeding customer expectations and maintaining a competitive edge.