LIVEWYER Platform Principles

LiveWyer's core architectural principles lay the essential groundwork for your platforms long-term success. We have used these principles with our clients for the last ten years, and these architectural principles underpin our many **success stories**.

Start your Cloud platform journey, and embed our ten LiveWyer architecture principles into your foundations.

1 Utilise open source (where feasible)

Incorporate open source solutions to avoid vendor lock-in, ensuring flexibility, competitive pricing, and minimal exit costs. Open source drives innovation with cutting edge tools and adaptability. Using open standards, APIs, and containerisation boosts interoperability and future-proofs your platform, providing strategic control, efficiency, and sustained competitive advantage.

2 Well-planned & well-defined taxonomy

Develop and implement a clearly defined taxonomy to ensure your platform is predictable and scalable. This approach enhances resource management and security with consistent naming conventions and hierarchical structures. Well planned and well defined taxonomy allows you to successfully maintain your platform while it remains adaptable and efficient in its longevity.

3 Infrastructure as Code

Implement ***Infrastructure as Code (IaC)*** to enhance your platform, enabling automated, consistent, and repeatable infrastructure deployments. It allows for version control, making changes trackable and auditable. IaC reduces human error, streamlines processes, and improves team collaboration, ultimately leading to more efficient and scalable infrastructure management.

4 Adopt Infosec industry best practices

To ensure top-tier security of your platform, ***adopt Infosec best practices***. Implement least privilege access to limit user permissions, use robust encryption to protect data in transit and at rest, and keep systems up-to-date with the latest security patches. These measures help safeguard against vulnerabilities and protect sensitive information.

5 Declarative configuration

Incorporate ***declarative configuration*** into your platform to ensure systems converge to a desired state. This approach enhances efficiency and reliability by defining the target configuration, allowing automated tools to manage changes and maintain consistency. It simplifies management, reduces errors, and ensures that infrastructure aligns with defined specifications.

6 Focus on interfaces and frameworks

To maintain a clean and scalable platform architecture, ***focus on well-defined interfaces and frameworks***. This approach preserves maintainability by clearly separating concerns and respecting the boundaries between abstractions. Doing so ensures components can evolve independently, reduce complexity, and facilitate easier debugging and updates, leading to a more robust system.

7 Don't repeat yourself

Incorporate ***DRY (Don't Repeat Yourself)*** principles into your platform design to minimise code repetition. This practice simplifies future code changes by centralising logic and reducing redundancy, making the system easier to maintain and extend. Avoiding duplication enhances code clarity and consistency, leading to more efficient development and debugging.

8 Adopt a Pull Model

Adopt a ***Pull Model*** in your platform architecture to improve maintainability and scalability. In this model, agents regularly retrieve updates from a central source of truth, ensuring consistency across systems. This approach simplifies update management, reduces the risk of inconsistencies, and allows for scalable and controlled deployment of changes, enhancing overall system stability.

9 Implement GitHub Flow

Implement ***GitHub Flow*** as a lightweight branch-based model for source control of your platform. This approach encourages collaboration from all team members, not just developers, by simplifying the process of proposing changes. It allows for seamless integration of contributions, enhances transparency, and ensures a streamlined workflow for code reviews and deployments.

10 GitOps and continuous delivery (CD)

Adopt ***GitOps and Continuous Delivery (CD)*** as a platform principle to streamline the deployment process. Using a CD system to deploy tested and reviewed changes ensures a consistent and automated workflow. This integration maximises the benefits of both methodologies, enabling rapid, reliable updates and maintaining a clear audit trail for all deployments.

Get in touch and see how we can help with your Cloud Native journey

LiveWyer has worked with many global enterprises to help implement robust, sustainable, and elegant architectures so that their Platforms can withstand the test of time.

Book an initial consultation **here** if you would like to speak to one of our experienced architects for support or guidance on your Platform journey.