CLOCKWORK

//case-study

air up

How air up unlocked seven figure time-to-market savings by leveraging total automation

By transitioning to a microservices-based architecture and robust CI/CD pipeline, air up drastically reduced deployment times and enabled continuous delivery of new features.

over $1M
Time to market savings

8000
Deploys in 6 months

15m
DORA Average Lead Time

Introduction

air up experienced remarkable growth as it dominated the scent-flavoured drinks market. With over 250 employees and more than 6 million customers it became increasingly apparent that existing delivery practices and digital infrastructure were struggling to keep pace with their success.

The heart of *air up's* operations was a third-party ERP monolith. While the system had adequately served the company during its initial growth phase, it was now showing significant signs of strain. Performance degradation had become a pressing

issue, with the system struggling to handle the increased load from expanding operations and started to impact customers' satisfaction. Changes were also increasingly harder to deploy without introducing unexpected side effects and bugs. Entering new markets or even maintaining market position became significantly more difficult, leading *air up's* decision to modernise its infrastructure and delivery processes.

In order to address the company's need for more scalable system and improve delivery process, Clockwork assisted the team in adopting an Archipelago Architecture alongside Trunk-Based Development and Continuous Deployment. This approach was underpinned by a Shift-Left Engineering strategy, allowing for full end-to-end testing on developer workstations and enabling 8000 deployments across 20 microservices in just six months. The solution was customised to the *air up* needs through the use of shared library modules in a Monorepo, Contract Testing and supported by Living Documentation. The software delivery process had also been improved with use of modern XP Development Practices, ensuring both speed and reliability.

While the transition was challenging, requiring changes in architecture and mindset, the benefits were significant. It led to increased scalability of the platform during peak traffic periods, improved delivery speed with new warehouse integration in US achieved two months ahead of schedule as well as provided invaluable insights to optimise stock levels through inventory reporting. Additionally, confidence of integration and communication with partners was vastly improved. The transition amounted to estimated savings of time to market and stock inventory management of over $1M.

About air up

air up is a German startup that has found success with its innovative reusable drinks bottle system. Founded in 2016, the company has developed a unique bottle design that allows users to infuse their water with various fruit and vegetable "flavour pods" to create flavoured water with no additives. This approach has resonated with health-conscious consumers looking for an alternative to sugary sodas and juices.

Since its launch, *air up* has experienced rapid growth, expanding from its German home market into other European countries. The company initially raised venture funding capital and has recently recorded growth of its user base to over 6 million customers. *air up's* innovative product and smart marketing have enabled it to carve out a strong position in the competitive flavoured water category.

Sector	Location	Org Size	Project Duration
Retail	Germany	250 employees	12 months

Challenge

Resolving compounding performance and quality issues

As the company ventured into new markets and broadened its product catalogue, it became increasingly apparent that their existing digital infrastructure was struggling to keep pace with their success. The heart of *air up's* operations was a third-party Python ERP monolith. While this system had adequately served the company during its initial growth phase, it was now showing significant signs of strain. Performance degradation had become a pressing issue, with the system struggling to handle the increased load from expanding operations. This slowdown was not merely an inconvenience; it was beginning to impact customer satisfaction and operational efficiency.

Compounding the performance issues was a concerning lack of observability within the system. The development team found themselves frequently flying blind, unable to quickly identify and resolve issues as they arose. This lack of insight made it challenging to maintain the system's reliability and to plan for future scaling needs. The development process itself was outdated for *air up's* rapidly evolving needs. The team operated on a branch-per-environment model, with separate branches for development, staging, and production. This approach led to complex merge processes and increased the risk of errors when porting code across environments. The release cycle was bound to weekly sprints, which often felt too rigid for a company needing to respond swiftly to market demands and operational challenges.

Testing practices were another area of concern. Unit testing was not a priority in the development culture, and the system itself had very few tests. This lack of comprehensive testing made it difficult to ensure reliability when making changes or adding new features. Deployments, which could take up to an hour, were often fraught with uncertainty. The team relied heavily on manual testing and what could be described as a "hit and hope" deployment strategy, leading to a high-stress environment and increased risk of issues in production.

As *air up* continued to grow, these limitations were becoming more than just technical hurdles; they were potential barriers to the company's continued success. The need for change was clear. *air up* required a new platform that could not only handle their current scale but also support their ambitious plans for future growth. They needed a system that could efficiently manage an expanding product catalogue, integrate seamlessly with new warehouse fulfilment vendors, and provide advanced fulfilment mechanics such as address validation and smart warehouse routing.

Moreover, the company recognised the need for real-time inventory reporting to optimise stock levels and streamline manufacturing processes. Integration with a new web shop implementation was crucial for improving the customer experience. Additionally, connecting with a robust data warehouse would enable the advanced reporting and analytics capabilities necessary for data-driven decision-making in a competitive market.

This context set the stage for a comprehensive overhaul of both the technical infrastructure and the development processes at *air up*. The company was poised to embark on a transformative journey, one that would not only address their immediate technical challenges but also foster a new culture of innovation, collaboration, and continuous improvement within their engineering team.

Solution

Monorepo Implementation

One of the fundamental decisions made to address the challenges faced by *air up* was the implementation of a monorepo structure. This approach represented a significant shift from the previous development model and was chosen for its ability to streamline development processes, allowed for better refactorability of the code and improve code sharing across the organisation. The monorepo strategy involved consolidating all the *air up's* codebase into a single repository, including the various microservices, shared libraries, and support tools. This centralisation was not merely a technical choice but a strategic one, aimed at fostering collaboration and code reuse across the entire development team. In the previous system, code sharing was difficult and often led to duplication across different services. With a monorepo, the team could easily create common modules for frequently used functionalities, such as authentication, logging, or data access patterns. These shared libraries could then be utilised across all services, ensuring consistency and reducing redundancy.

Another significant advantage of the monorepo was its support for atomic commits across multiple services. When a change affected multiple parts of the system, developers could make these changes in a single commit, ensuring that all related modifications were applied simultaneously. This greatly reduced the risk of inconsistencies that could arise from changes being applied to different services at different times. From a developer experience perspective, the monorepo made it easier for team members to understand the entire system. New developers could clone a single repository and have access to the entire codebase, making onboarding smoother and encouraging a holistic understanding of the platform.

The monorepo also simplified the continuous integration processes and aligned well with *air up's* move towards a microservices architecture. With all code in one repository, it became easier to set up comprehensive CI/CD pipelines that could build, test, and deploy all services together. This was crucial for implementing the Trunk-Based Development model and achieving the high frequency of deployments that *air up* was aiming for.

Trunk-Based Development

Transitioning to Trunk-Based Development was a crucial step in *air up's* transformation of their development practices. This approach represented a significant departure from their previous branch-per-environment model and was integral to achieving their goals of increased deployment frequency and improved code quality. Utilising Trunk-Based Development eliminated the complex merging processes that were previously a source of delays and errors. It also fostered a culture of continuous integration, where code changes were immediately integrated with the work of other developers.

To support this model, the team heavily invested in automated testing. Each commit triggered a comprehensive suite of tests, including unit tests, integration tests, and end-to-end tests. This extensive testing provided developers with the confidence to push changes directly to the main branch, knowing that issues would be caught quickly.

The adoption of Trunk-Based Development also necessitated a shift in how features were developed. The team implemented feature flags, allowing new functionality to be deployed to production in a dormant state and activated when ready. This approach enabled continuous deployment while maintaining control over feature releases.

Archipelago Architecture

In designing the new system architecture for *air up*, the Clockwork team adopted an Archipelago Architecture, a nuanced take on microservices architecture. This model was chosen to balance the benefits of microservices with the need for cohesive domain-specific functionalities. The Archipelago Architecture organised services into “bubbles” - collections of closely related modules that shared a common database and were deployed as a unit. Each bubble typically consisted of one or more HTTP services or queue-driven workers, all centred around a shared domain layer core. This approach allowed for a logical grouping of functionalities while maintaining the flexibility and scalability of a distributed system. At the heart of each bubble was a hexagonal core, implementing the domain logic. This core was agnostic to the delivery mechanism, whether it was an HTTP API or a message queue worker. Separation of concerns ensured that the business logic remained clean and independent of infrastructure details.

The Archipelago Architecture provided several advantages. It allowed for independent scaling and deployment of different functional areas of the system. For instance, the order processing bubble could be scaled differently from the inventory management bubble, depending on load and business needs. At the same time, it avoided the extreme granularity and potential complexity of a pure microservices approach. This architecture also facilitated better data consistency within each bubble, as related modules shared a database. However, inter-bubble communication was designed to be asynchronous and resilient, respecting the boundaries between different domains of the business.

By adopting this Archipelago Architecture, *air up* created a system that was both modular and cohesive, setting the stage for improved scalability, maintainability, and alignment with business domains.

Continuous Integration

air up’s new Continuous Integration (CI) system was a critical component in their development transformation. Built to support the monorepo structure and Trunk-Based Development, it enabled rapid feedback and maintained code quality across the entire codebase.

The CI pipeline was triggered automatically with each commit to the main branch. It began with a build process that compiled all affected services and their dependencies within the monorepo. This ensured that changes in shared libraries or interdependent services were immediately validated. Following the build, a comprehensive testing suite was executed. This included unit tests for individual components, integration tests for service interactions, and end-to-end tests

simulating real-world scenarios. The team also implemented contract tests to verify the integrity of service interfaces.

Performance tests were included to catch any regressions in system efficiency. To manage the complexity of the monorepo, the CI system employed intelligent caching and parallelisation strategies. This allowed it to focus on affected areas of the codebase, maintaining quick feedback cycles despite the growing repository size. The CI dashboard provided real-time visibility into build and test statuses, allowing developers to quickly identify and address issues.

This rapid feedback loop was crucial in maintaining the high velocity of changes while ensuring system stability.

Shift-Left Practices

air up embraced a comprehensive Shift-Left Practices, moving critical practices earlier in the development lifecycle. This strategy significantly improved code quality and reduced issues in production. Observability was implemented from the outset of development. Engineers integrated logging, metrics, and distributed tracing into their services from the beginning. This early focus on observability allowed for better debugging during development and provided crucial insights into system behaviour in production.

End-to-end testing was integrated into the local development environment. Engineers could run full system tests on their workstations, simulating production-like scenarios. This capability allowed for early detection of integration issues and ensured that new features worked correctly within the broader system context.

Security considerations were also shifted left. The team adopted a “security as code” approach, incorporating security best practices into shared libraries and infrastructure-as-code templates. Automated tests were integrated into the CI pipeline, identifying potential inconsistencies in security stack.

Performance testing was conducted throughout the development process, rather than just before release. This allowed for early identification and resolution of performance bottlenecks.

By adopting these Shift-Left Practices, *air up* significantly reduced the number of issues reaching production, improved system reliability, and increased the team’s confidence in their ability to deliver high-quality software rapidly.

Results

Time-to-Market and Inventory Optimisation Savings of over $1 Million

The transformation of *air up*'s infrastructure and processes resulted in substantial financial savings, particularly in terms of time-to-market and inventory optimisation, amounting to estimated over $1 million. These savings were not just a boon to the company's bottom line; they directly translated into enhanced customer value. By streamlining the supply chain and improving inventory management through real-time reporting, *air up* was able to ensure that products were always available when customers wanted them. This optimisation reduced the occurrence of stock oversells, which could frustrate customers and lead them to seek alternatives.

Additionally, the improved efficiency in bringing new products to market meant that customers had access to the latest offerings much sooner than before, allowing them to enjoy new flavors and innovations without delay. The financial savings also enabled *air up* to reinvest in areas that directly benefited customers, such as expanding product lines, enhancing customer service, or offering more competitive pricing. By passing on the benefits of these savings to customers, *air up* strengthened its market position and further solidified its reputation as a customer-focused brand that delivers both quality and value.

Enhanced System Reliability and Observability

The transformation of *air up*'s digital infrastructure also led to significant improvements in system reliability and observability, which played a crucial role in enhancing customer value. The legacy system suffered from performance issues and lacked robust observability tools, leading to frequent downtimes and a subpar customer experience.

However, with the introduction of standardised observability tools, such as structured logging, distributed tracing, and real-time monitoring, *air up* was able to achieve a much higher level of system reliability. These tools allowed the development and operations teams to proactively monitor the system, quickly identify and address issues, and ensure that the platform remained stable even during peak traffic periods. For customers, this translated into a more reliable service with fewer interruptions, ensuring that they could access *air up*'s products and services whenever they needed them. The enhanced observability also allowed air up to optimise system performance continuously, leading to faster load times and a more responsive user interface, further improving the customer experience.

By investing in reliability and observability, *air up* demonstrated its commitment to providing a high-quality, dependable service that customers could trust, which is essential for building long-term customer loyalty and satisfaction.

8,000 Deployments Across 20 Services in 6 Months

The ability to achieve 8,000 deployments across 20 services within just six months stands as a testament to *air up*'s transformation into a truly agile and customer-centric organisation. This remarkable feat was made possible through the adoption of Trunk-Based Development and Continuous Integration practices, which allowed the company to confidently and safely push updates to production with every commit. For customers, this meant that their feedback, suggestions, and needs could be addressed rapidly, leading to a platform that was continually evolving to better serve them. The high frequency of deployments enabled *air up* to roll out new features, performance enhancements, and bug fixes almost in real-time, significantly improving the overall user experience.

The agility demonstrated by these rapid deployments also ensured that *air up* could quickly adapt to changing market conditions, introduce new products, or respond to emerging customer trends without delay. This responsiveness not only increased customer satisfaction but also helped build trust and loyalty, as customers felt heard and valued. The ability to frequently update and improve the platform also reduced the risk of large-scale failures, as issues could be identified and resolved incrementally, further enhancing the reliability and stability of the service provided to customers.

Improved Development Velocity

The overhaul of *air up*'s development processes led to a significant improvement in development velocity, which had a profound impact on customer satisfaction and experience. With the adoption of Trunk-Based Development, Continuous Integration, and modern ensemble programming practices, the development team was able to work more efficiently and effectively. This increased velocity meant that new features, improvements, and fixes could be developed and deployed much faster than under the old system. For customers, this meant that their needs and feedback could be addressed more quickly, leading to a platform that was always up-to-date with the latest enhancements.

The faster development cycle also allowed *air up* to stay ahead of competitors by rapidly innovating and introducing new products and features that met the evolving demands of the market. This agility in development ensured that customers consistently received a service that was cutting-edge and tailored to their preferences, enhancing their overall experience. Moreover, the increased

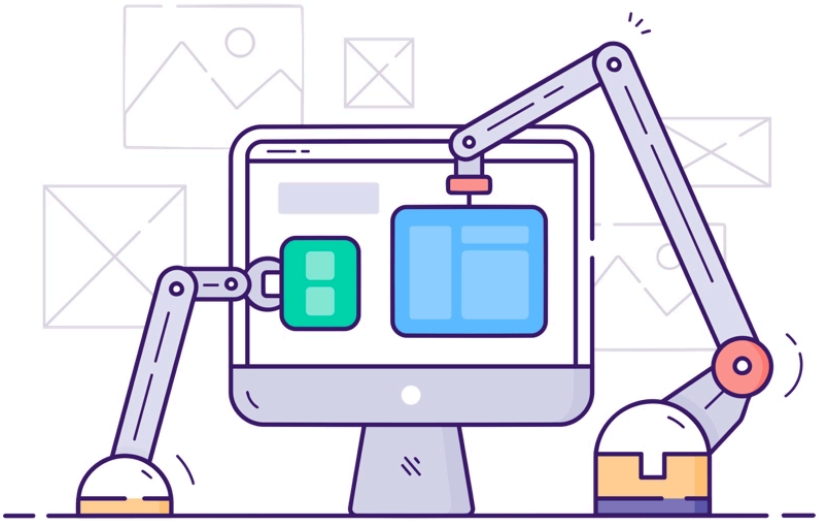
development velocity reduced the time taken to fix any issues, ensuring that customers faced minimal disruptions and could enjoy a smooth, reliable service. By prioritising speed and efficiency in development, *air up* was able to deliver more value to its customers in less time, reinforcing its commitment to excellence and customer satisfaction.

Conclusion

Clockwork's assistance in the comprehensive transformation of *air up*'s digital infrastructure and development processes not only resolved critical performance issues but also positioned the company for sustained growth and innovation. By embracing a modern architecture and quality-prioritising XP development practices, the engineering team has significantly improved its operational efficiency, reduced time to market, and enhanced overall system reliability.

> **"We chose Clockwork as a partner to drive forward a technically challenging, mission-critical initiative and 'up our game' in the process. Our shared mindset and commitment to engineering excellence have turned out to be a great match, and I look forward to the opportunity to work together again in the future."**
>
> - Patric Fornasier, CTO, air up

Read more Clockwork case studies →



Choose technology professionals not technology *pretenders*

Clockwork is an invite-only network of **world-class independent technology consultants** who help companies to deliver ***without the hefty markup*** of traditional consulting firms.

CLOCKWORK

https://clockwork.ing