xfactr™

CASE STUDY

Migrating Application to Azure Landing Zone with Ingress Implementation

Business Needs/Background

Introduction Our application, Spec-Builder, is a comprehensive solution built using C#, SQL, TypeScript, JavaScript, npm, and Angular. Initially hosted on Azure, the architecture lacked a structured landing zone, leading to challenges in scalability, security, and cost management. To address these issues, we decided to migrate to an Azure landing zone for a more robust, scalable, and secure infrastructure.

Objectives

- Scalability: Enhance the application's ability to handle increased loads.
- Security: Implement stringent security measures to protect data and services.
- Cost-Effectiveness: Optimize resource usage to reduce operational costs.
- Integration: Seamlessly integrate with multiple services like SAS, Logz.io, and others

Migration Strategy

1. Assessment: Evaluated the current infrastructure and identified components for migration.

2. Planning: Developed a detailed migration plan, including timelines and resource allocation.

3. Azure Landing Zone Selection: Chose an appropriate Azure landing zone to meet our scalability and security needs. 4. Infrastructure as Code (IaC): Utilized scripts to automate the creation of infrastructure components.

5. Ingress Implementation: Integrated Azure Application Gateway and Azure Front Door for managing ingress traffic.

6. Testing: Conducted thorough testing to ensure the migrated application functioned as expected.

7. Deployment: Deployed the application to the new Azure landing zone.

Ingress Implementation • Ingress Definition: A collection of rules that allow inbound connections to reach cluster services.

• Tools and Services:

o Azure Application Gateway: Managed load balancing and secure ingress traffic.
o Azure Front Door: Provided global load balancing and fast failover.

Configuration Process:

• Configured the Application Gateway with necessary routing rules and SSL termination.

• Set up Azure Front Door for global traffic management and failover.

• Integrated with Kubernetes for dynamic service discovery and routing.

Challenges

• Integration with SAAS and other tools: Ensured seamless data flow and logging.

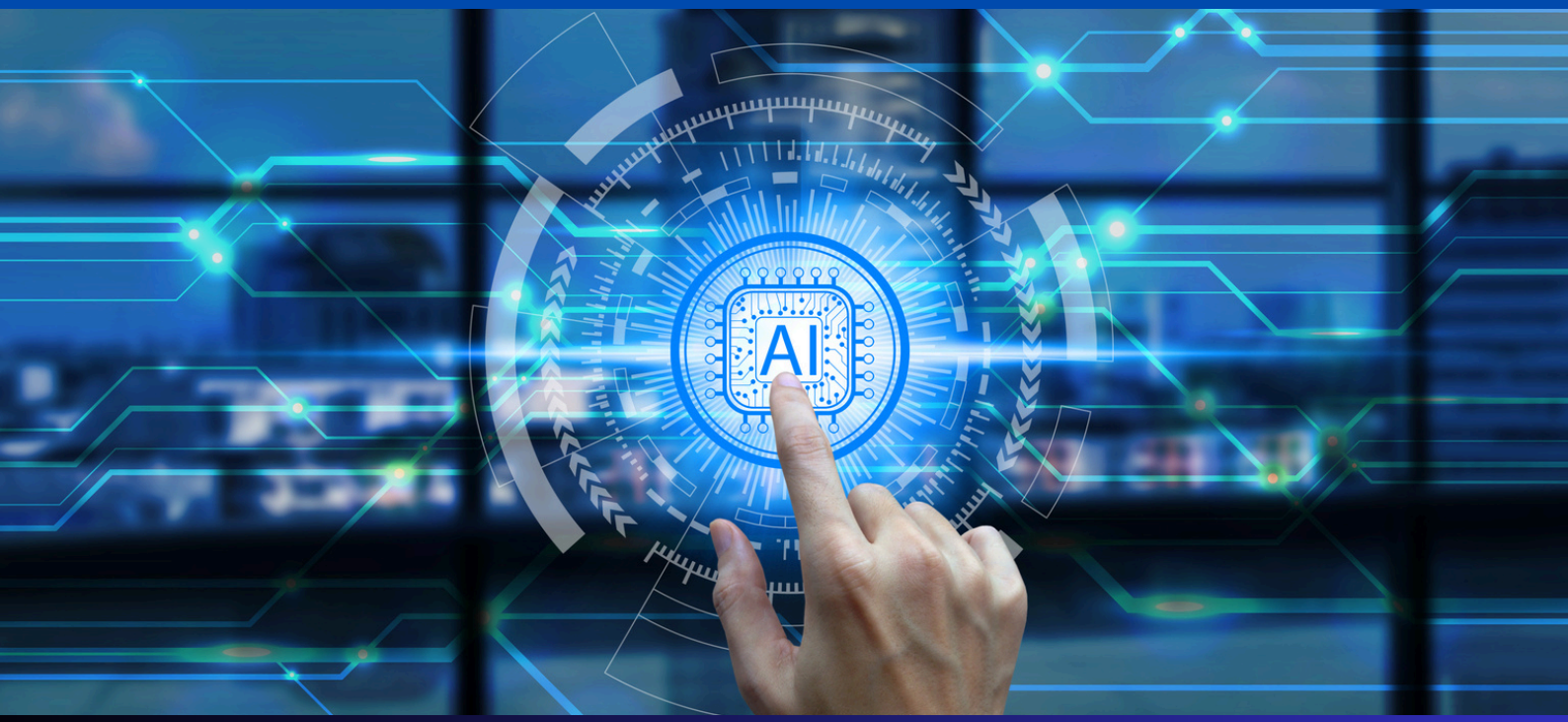
• Firewall Configuration: Opened necessary ports and configured firewall rules.

• Scripted Infrastructure Creation: Automated the creation of infrastructure components using scripts.

• IP Shortage: Addressed IP shortage issues by implementing ingress solutions.

• General Workflow Dispatch: Utilized GitHub Actions for continuous deployment and integration.

• Onboarding New Countries: Collaborated with local teams to ensure compliance with regulations, translated content, and adapted features to meet local user needs.



Teamwork

Throughout the migration process, our team worked collaboratively to address various issues. Regular stand-up meetings fostered open communication, allowing us to share challenges and brainstorm solutions together. Each team member contributed unique insights, and we leveraged our collective expertise to overcome technical hurdles, streamline the onboarding process for new countries, and enhance application performance.

Results

- Performance: Significant improvements in application performance and load handling.

- Security: Enhanced security measures with better access controls and monitoring.

- Cost Savings: Optimized resource usage, leading to reduced operational costs.

Lessons Learned

- **Automation: Automating infrastructure creation and configuration significantly reduces errors and deployment time.**

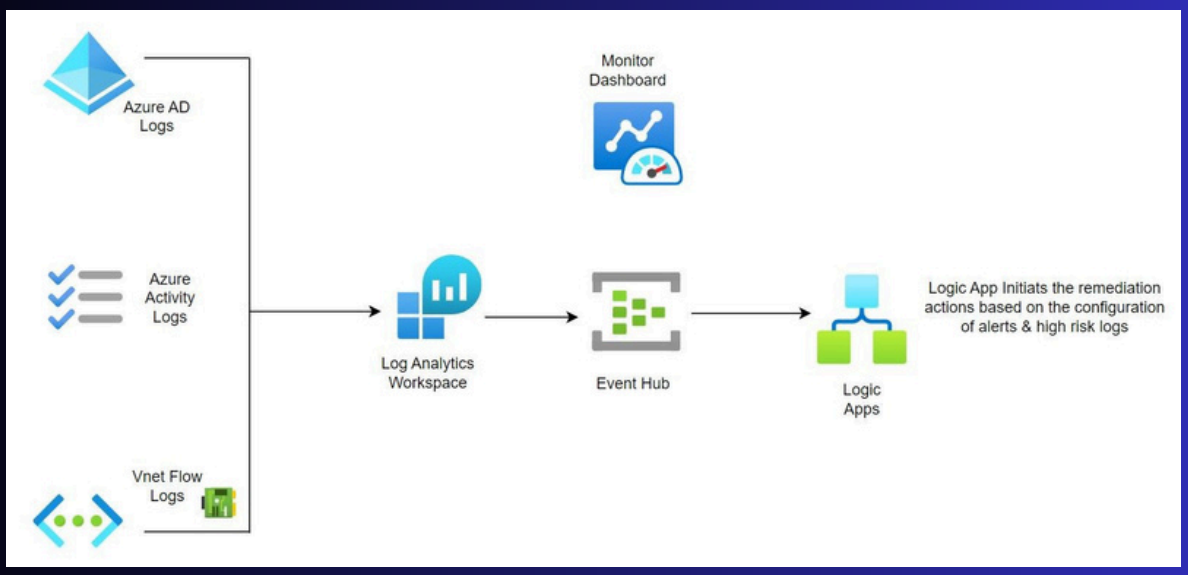
- **Testing: Comprehensive testing is crucial to identify and resolve issues early in the migration process.**

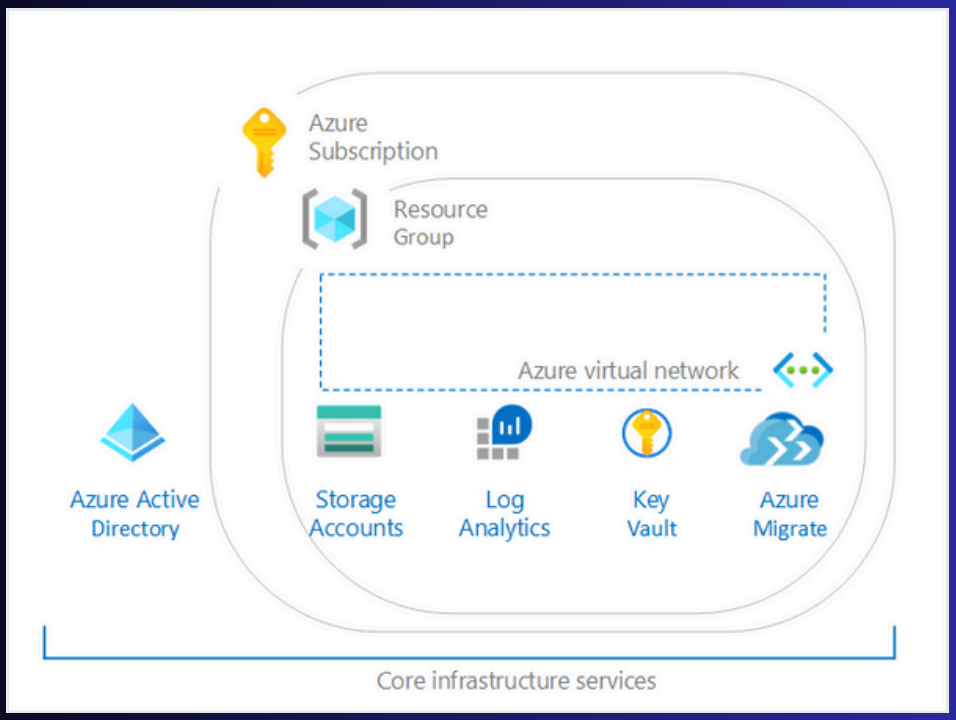
- **Documentation: Maintaining detailed documentation aids troubleshooting and future migrations.**

- **Time Management: The migration process took a year to overcome challenges and complete successfully.**

Conclusion

The migration to an Azure landing zone with ingress implementation was a success, resulting in a more scalable, secure, and cost-effective application. The structured approach and use of automation tools played key roles in the smooth transition and improved performance. The project took a year to complete, addressing various challenges such as IP shortage and integration with multiple services.


Azure AD Logs
Azure Activity Logs
Vnet Flow Logs
Log Analytics Workspace
Monitor Dashboard
Event Hub
Logic Apps
Logic App Initiats the remediation actions based on the configuration of alerts & high risk logs


Azure Subscription
Resource Group
Azure virtual network
Azure Active Directory
Storage Accounts
Log Analytics
Key Vault
Azure Migrate
Core infrastructure services

Value Delivered to the Customer

- Reduction in overall operational infrastructure and maintenance cost by 40%.
- Improved application scalability.
- Increased availability of the application by 70% with multi-region deployment.
- Robust disaster recovery strategy delivered for a dynamic workload.
- Enhanced security.
- Improved and efficient monitoring, logging and alerting system for the entire application.

For more information, write to us at hello@xfactr.ai

About XFactr.AI

At XFactr™.AI, we're revolutionizing industries with our bold 3D approach to digital transformation:

1. **Digital Full-stack Development**
2. **Data Science & AI ML**
3. **DevOps & Cloud**

Data is our backbone, and AI is our superpower. We blend traditional digital technologies with cutting-edge AI to create future-ready solutions that drive real impact. With a proven track record of building and exiting successful companies, our leadership brings decades of expertise in digital tech and AI to inspire trust, innovation, and results.

Whether empowering global enterprises or fueling disruptive startups, we turn bold ideas into intelligent products. Join us—we're on track to become the next unicorn by 2040! 🚀 People might call us mad, but we're focused, ambitious, and determined to make it happen.