How to Successfully Deploy Azure Kubernetes Services at Scale

INSIGHT REPORT

BreakFree Solutions

Introduction

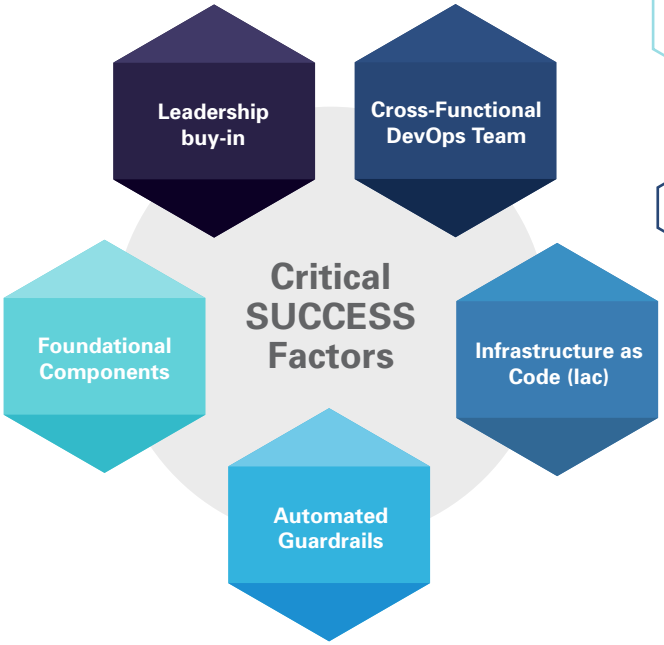
The future of enterprise IT applications is in the cloud, and enterprises understand that containers and cloud-native design are the promised land.

It's safe to assume that container popularity and usage will grow even more significant with Microsoft's introduction of Azure Kubernetes Services (AKS), a secure and fully managed Kubernetes service from Microsoft Azure that makes deploying and managing container applications easy.

However, AKS is like chess. It's simple to learn but challenging to master. The effort to operationalize AKS should not be underestimated and requires broad expertise. It can be extraordinarily complex if you don't have the correct foundational components in place or your IT team lacks the necessary skills. The value and the benefits of containers will certainly justify these efforts for enterprises looking to modernize software development by providing native, mature automation and DevOps capabilities.

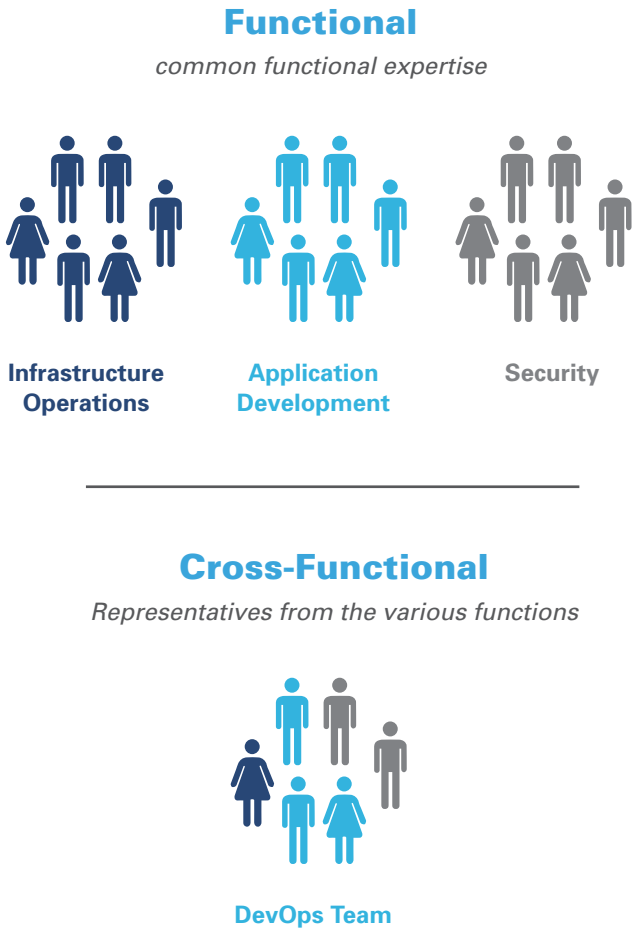
In this Insight Report, you will find an actionable roadmap that includes critical success factors, a reference architecture, an overview of the AKS deployment plan and a breakdown of backlog components necessary to fully establish and operationalize Azure Kubernetes Services.

Critical Success Factors



Leadership buy-in

It's imperative that right from the start your leadership team understands utilizing Azure and AKS means building out a whole new development platform. You will essentially be building a product that you will be actively enhancing and improving over the next 5 - 10 years. When you deploy AKS, you establish minimally viable functionality with the idea that additional layers of functionality will be added over time. A team and investment for the long haul are necessary to be successful with AKS.



Cross-Functional DevOps Team

Having a cross-functional DevOps team is essential not only to deploy AKS but to meet the demands of today's evolving technology capabilities. Cross-functional teams eliminate silos, which often plague enterprise IT, by improving communication and collaboration. When instituted, cross-functional teams foster a culture of learning, promote innovation, enable quicker feedback loops and connect the team to the end goal through the use of Agile Scrum, which means sprints, daily meetings, product artifacts, and team dynamics.

If your current IT organization doesn't employ the use of cross-functional DevOps teams, make setting up teams a priority. Cross-functional teams ensure you have the right people and skill sets needed to develop or deploy.

For a successful AKS deployment, your cross-functional team must include the following two key roles: Application Development and Infrastructure Operations. Your application development role needs to understand how to communicate with, support and test your applications, while your infrastructure operations role needs to understand your infrastructure concepts, such as VNets, and have a strong networking background.

Infrastructure as Code (IaC)

Your AKS deployment must be built using branch-driven release automation Infrastructure as Code (IaC). Cloud should never be configured like on-premise infrastructure. Instead, cloud is configured through IaC, which is infrastructure built via source code. IaC requires the same software development lifecycle capabilities and Continuous Integration / Continuous Deployment (CI/CD) pipelines used in application software development. CI/CD pipelines are a must for cloud-native development platforms.

Automated Guardrails

In this new automated cloud-native world, automated guardrails are used to keep your system safe and secure without sacrificing delivery velocity. Platform-centric guardrails from Microsoft including Azure Policy and Azure Automation runbooks, as well as third-party guardrails such as Twistlock or Aquasecurity, will provide the ability to automate the management of security.

Infrastructure as Code (IaC) is a method to provision and manage IT infrastructure through the use of source code, rather than through standard operating procedures and manual processes. IaC helps you automate the infrastructure deployment process in a repeatable, consistent manner, which has many benefits:

- ***Speed and Simplicity***
- ***Configuration Consistency***
- ***Minimization of Risk***
- ***Increased Efficiency in Software Development***
- ***Cost Savings***

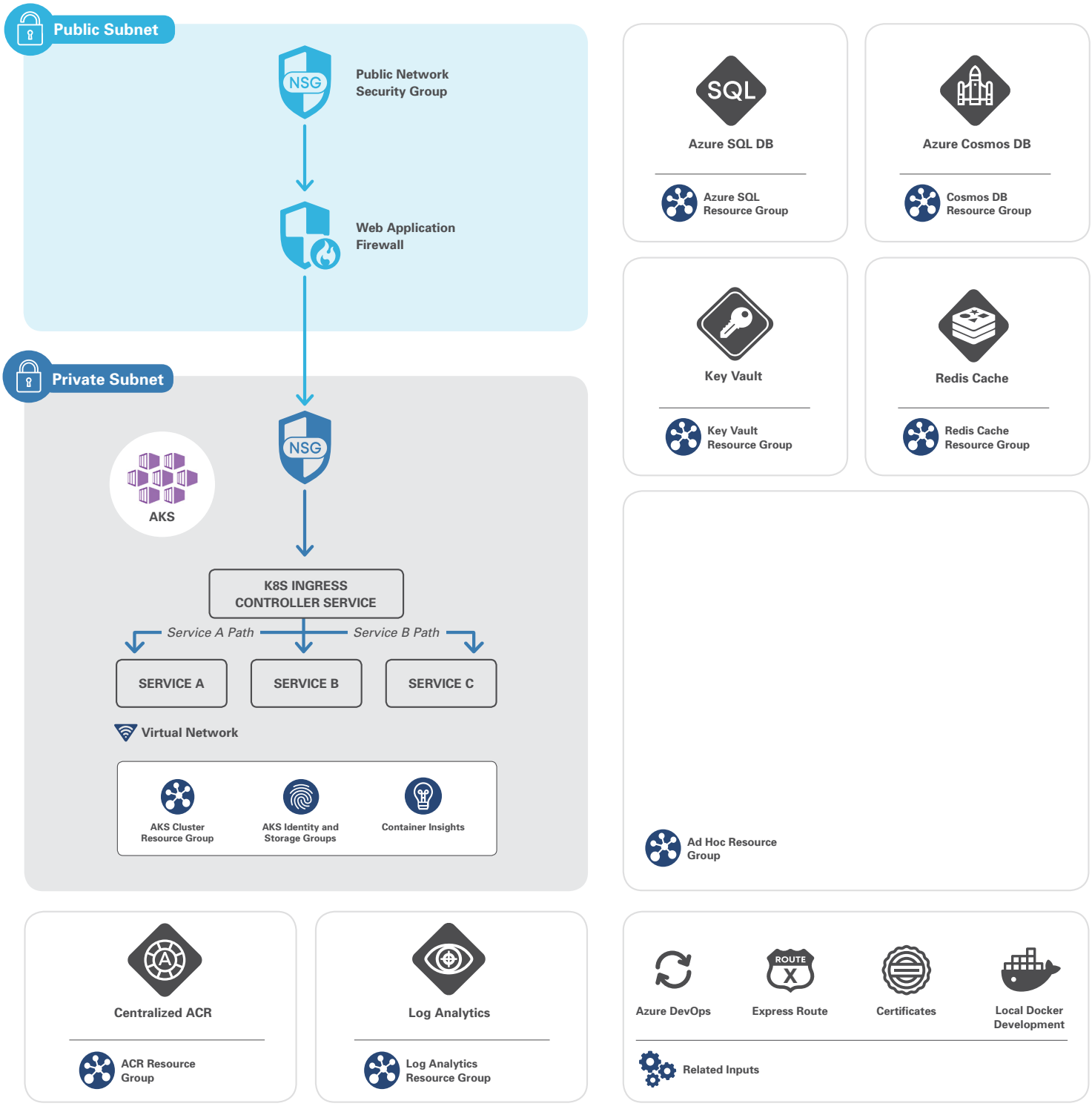
Foundational Components

Real production enterprise workloads will require the following foundational components in addition to the AKS service.

✓	**Azure Account**	Establish Microsoft Azure account.
✓	**Azure Active Directory (Azure AD)**	Microsoft's cloud-based identity and access management service that provides secure access to Azure APIs and services.
✓	**Application Identity Management**	Application Identity Management services such as ADFS, Ping, Okta are required so that the applications have the appropriate identity services available to them.
✓	**Azure Key Vault**	Safeguards cryptographic keys and other secrets used by cloud apps and services.
✓	**VNet and Subnets**	Private IP-based networking configuration that enables secure network communications between platform components.
✓	**Security Groups and Security Roles**	Rules set on your networking components that allow or disallow traffic.
✓	**Azure Monitor for Containers**	Monitors the performance of container workloads deployed on AKS.
✓	**Logging Service**	Use a logging service that is both configured to take advantage of the Azure-native logging capabilities and can connect with your other centralized logging services such as Splunk.
✓	**Monitoring Service**	Utilize a modern SaaS-based suite of monitoring tools, natively available from Azure combined with more advanced SaaS-based tools. We've worked with and highly recommend Datadog.
✓	**Ingress Controller**	Provides reverse proxy, configurable traffic routing, and encryption termination for Kubernetes services and are used to configure the ingress rules and routes for individual Kubernetes services.
✓	**DNS Service**	Use DNS that is configurable via APIs, preferably Azure DNS.
✓	**Secure Azure Storage**	Configure storage to be secure and usable on-demand by AKS.
✓	**Database**	Your platform needs a comprehensive suite of database services to allow optimal use of cloud platform services. You should start with Azure SQL Database and CosmosDB. Later, be prepared to expand these data services to include data streaming, data warehousing, content storage, etc.
✓	**Azure Application Gateway**	Web traffic load balancer that enables you to manage traffic to your web applications.
?	**Azure ExpressRoute**	Azure ExpressRoute is different from the other foundational systems because it requires a decision from the project stakeholders. We recommend in most circumstances starting without it, and unless you can identify a specific workload that's going to need on-premise connectivity, then wait until you actually get there.

Reference Architecture

The following diagram represents a visualization of our recommended reference architecture for your first iteration of your AKS development platform. It's organized by lifecycle to ensure the least amount of interdependency and optimize flexibility between services when it comes to provisioning, reconfiguration, or deletion. Shared components have been minimized, but for existing shared services, ensure they are accessible via API to provide composability for the future.



AKS Deployment Plan

We suggest executing your AKS deployment using the following five phases.

PHASE 1: Planning

In phase one, ensure your cross-functional DevOps team is in place, Objectives and Key Results (OKRs) and project milestones are identified and create the backlog items and strategy for how the team will achieve the milestones, i.e., an actionable Roadmap.

PHASE 2: IaC Engineering

In phase two, your cross-functional DevOps team will use IaC to engineer and configure all the necessary Azure foundational components that will make up the AKS development platform.

PHASE 3: Application/Development Team Integration

In phase three, you will engage with motivated development resources who are willing to be beta testers for the development platform. Ideally, you'll find a real production development use-case to start running on your platform. This effort will allow you to test and improve your platform functionality. This time is also ideal for getting a better understanding of your development teams' awareness of cloud-native development patterns, containers, and modern DevOps operating modes. This information will be necessary for Phase 5.

PHASE 4: Operationalize the Platform

In phase four, your cross-functional DevOps team will transition into operating mode. To do that, you will establish SOPs for AKS platform operations, consumption, and security.

PHASE 5: Enabling Cloud-Native Development

In phase five, you will begin to leverage the capabilities of AKS at scale and train your developers on architecting and developing their applications utilizing cloud-native best practices. Cloud-native design best practices encompass all of the methods, architecture patterns, and techniques you will need to fully realize all of the benefits that come from moving to the cloud. By doing only some or a few of these best practices, you won't be fully optimized for cloud, and since you pay for cloud based on consumption, you want to ensure you are always optimized.

The Backlog

Now that we've discussed all five phases of a successful AKS deployment at a high-level, let's dive deeper and examine the actual backlog of work that needs to be completed in each phase.

Phase 1: Planning Activities

Assemble your DevOps team through functional decomposition

A comprehensive development platform based on AKS requires a DevOps team that crosses many traditional technology silos and requirements. By analyzing your existing technology silos and decomposing the functions needed to deploy, operationalize, and operate the AKS-based platform, you can determine whom to assign to the DevOps team. In most circumstances, this requires network, IT security, application development, infrastructure, and operations functions.

Institute an Agile work and Metrics-based Product Management System

Scrum is crucial when establishing high-performance DevOps teams. New DevOps teams will need the appropriate coaching and support required to institute Scrum and build a backlog with milestones and work items. A metrics-based approach to product management based on Google's Objectives and Key-Results (OKRs) system (in addition to what Scrum provides) is also essential. OKRs enable governance of the goals and work the DevOps team is performing over longer cycles then the typical two-week sprint. This strategy will allow you to build and manage an actionable Roadmap to successful AKS deployment.

Develop Initial Versions of DevOps Process Artifacts and the Supporting AKS Solution Design

Create DevOps process Value Stream Maps and detailed AKS and foundational systems solution designs that will support the anticipated workloads and development use cases for the platform. These artifacts will be used to demonstrate and ensure alignment on the technical vision required to support the identified milestones and key results.

Phase 2: IaC Engineering Activities

The successfully launched DevOps team can now focus on the IaC engineering activities required to deploy, configure, and scale the AKS development platform.

After analysis of our previous AKS deployments, we've established the following minimally viable backlog of IaC engineering activities that are common across the majority of AKS deployments. The backlog items consist of two sections: 1) Preparation and Support of Systems and Environments for IaC Automation Engineering and 2) writing the code.

Preparation and Support of Systems and Environments for IaC automation Engineering

1. Configure the basic set of foundational Azure services required to support IaC engineering activities: Azure account structure for production and non-production, tagging and asset identification, AzureAD and, if on-premise network connectivity is needed, establish express-route connectivity.
2. Set up a Git repository, code repository project structure, and CI/CD system. Ensure the entire team understands and has the skills and access to work with the Git repository and CI/CD systems. Don't overlook helping everyone understand how to work with pipelines and utilize Git. CI/CD can be intimidating to non-application developers.
3. Ensure all centralized shared services such as DNS, application identity management, logging, and monitoring are available for automated configuration both in production and non-production environments.
4. Initialize your IaC project and supporting automation tooling such as Terraform and confirm all environments and shared services are configurable via automation.
5. Configure a multi-environment IaC engineering release pipeline and confirm deployment, roll-back, and testing capabilities of the pipeline. Don't overlook automated testing in the IaC space. Add it into your definition of done from the get-go.

IaC Engineering – Writing the Code

Engineer IaC automations for the platform and supporting platform components. In most scenarios, the following is the order by which components and operations are automated through IaC pipelines:

1. Provisioning and configuration of the foundational networking components: Resource Groups, VNet Configuration, Subnet Role-based IAM configuration, Security Groups TCP-based configuration, Firewall configuration and Ingress Controller setup
2. Provisioning and configuration of Azure Kubernetes Services
3. Provisioning, configuration, and use of Azure Key Vault
4. Provisioning, configuration, and pipeline and platform integration with Azure Container Registry (ACR)
5. Provisioning, configuration, clustering and registry integration of the container security tool (Twistlock)
6. Creation and modification of hardened and standardized container image creation and testing with Azure Container Registry and guardrails tooling
7. Provisioning and configuration of AzureSQL and CosmosDB
8. Data operations required for operations of AzureSQL and CosmosDB (data loads, scheme changes, data quality testing, etc.)
9. Configuration of logging and monitoring of all platform components and platform hosted applications
10. Configuration of centralized application identity services, DNS, and asset management capabilities (tagging)
11. Provisioning, configuration, and integration of Azure-based guardrail technologies that ensure desired state configuration and platform use

Phase 3: Application / Development Team Integration Activities

Confirmation that applications and development teams can integrate and utilize the platform is critical to formalizing it. IaC automations and CI/CD pipelines must be tested rigorously before being put into the developer's hands. Address Issues and required enhancements before production of the AKS-based development platform goes live. The following approach is recommended to complete this phase:

1. Work with developers to document the SOPs for releasing an application update.
2. Select the ideal "pilot" candidate to be the first containerized application.
3. Developers build a container image for the "pilot" application that runs on their local workstation. Please don't test any features yet, but ensure it runs minimally.
4. Push the application source code to Git.
5. Create an application specific CI/CD build that pulls source code from Git, builds and tags a new container image, and stores it in ACR.
6. Create an application specific CI/CD release job that grabs the tagged container image from ACR and successfully deploys the container image into AKS.
7. Test that the application runs as expected. Again, not testing features yet, but only ensuring it runs minimally.
8. Once the end-to-end is complete and working, add any application-specific tests (lint, QA, regression, unit, etc.) to our two new CI/CD jobs.
9. Migrate your existing db schema and application data into Azure SQL database or CosmosDB depending on requirements.
10. Now, the development team can work on testing application features until the app performs as expected.
11. Integrate other Azure services that are required by the "pilot" application, including Key Vault, SQL Database, Cosmos DB, etc.

Phase 4: Operationalize the Platform

Once there is a high degree of confidence in the production readiness of the IaC, CI/CD pipelines, and platform SOPs, operational functions need to be established to ensure long-term platform stability, security, and compliance. The DevOps team will be responsible for maintaining the foundational AKS platform components. This team will need to establish and document their SOPs. At a minimum, the following must be created.

- **How to push an IaC update**
- **How to delete an IaC Stack**
- **How to roll back an IaC update**
- **How to deploy IaC to an entirely new environment**
- **How to handle an Azure Service outage**
- **How to push an application update**
- **How to delete an application**
- **How to roll back an application update**
- **How to handle an application outage**
- **How to monitor and control via guardrails**
- **How to harden and update container images**
- **How to restore database configuration and data**
- **How to manage the scaling of the platform**
- **How to manage and report on cost of the platform**

Once this minimal set of operational procedures has been instituted, the platform can be considered operational.

Phase 5: Enabling Cloud-Native Development Activities

Now the platform is ready for use, but it is paramount to ensure that developers understand developing with a cloud-native approach. Since cloud-native is not the standard practice in most development organizations, the activities in this phase are focused on training, knowledge sharing, consulting with your developers about how to architect their applications in a cloud-native way, and how to consume containers properly so the full benefits of your AKS platform can be realized.

- ☑ ***Pair with and train developers on cloud-native development and proper container consumption.***
- ☑ ***Create and publish standard patterns and best practices for distribution to the application development teams. A wiki is your best bet here if you want to create a centralized knowledge base that will actually be utilized.***
- ☑ ***Shift deployment processes to take advantage of CI/CD.***
- ☑ ***Ensure the developers understand how to build applications that are optimized for the AKS platform.***
- ☑ ***Publish sample applications that provide examples of optimal cloud-native architectural patterns on the platform.***

This phase should continue until the development organization has standardized on cloud-native as the preferred architectural pattern.

Accelerating Your AKS Deployment

The process may be complex depending on your IT group's current state and skills. There are opportunities to accelerate the AKS deployment process by leveraging the *BreakFree RapidDeploy for AKS* service.

BreakFree RapidDeploy for AKS compresses the timeframe for getting AKS operational. Leverage our substantial DevOps expertise to ensure you quickly realize the benefits of AKS by accelerating through all five phases without sacrificing safety, security, and operational ability. This allows your team to focus on truly taking advantage of AKS' capabilities and reach the end goal of cloud-native development. *BreakFree RapidDeploy for AKS* also affords your team the opportunity to learn during deployment by pairing and knowledge sharing with our AKS subject matter experts.

BreakFree RapidDeploy for AKS equips you with BreakFree developed automations, reference architectures, and standard operating procedures to deploy, operationalize and onboard development teams in only 10 weeks.

Learn more about accelerating your deployment of Azure Kubernetes Services with *BreakFree RapidDeploy for AKS* by visiting BreakFreeSolutions.com.

About BreakFree Solutions

BreakFree Solutions are a diverse team of IT experts with skills in cloud native design, cloud architecture, infrastructure automation, software development, continuous deployment, security, DevOps, and Agile delivery. BreakFree offers practical cloud, DevOps, and automation professional services to enterprise companies, and BreakFree's services are based on clearly defined positions on the future of IT operations shaped by our decades of real-world experience.

Insight Report Contributors:

Mitch Northcutt
Bradley Clerkin
Westly Edge
Ryan Schubert
Richard Giraldi

To learn more about BreakFree Solutions, visit BreakFreeSolutions.com