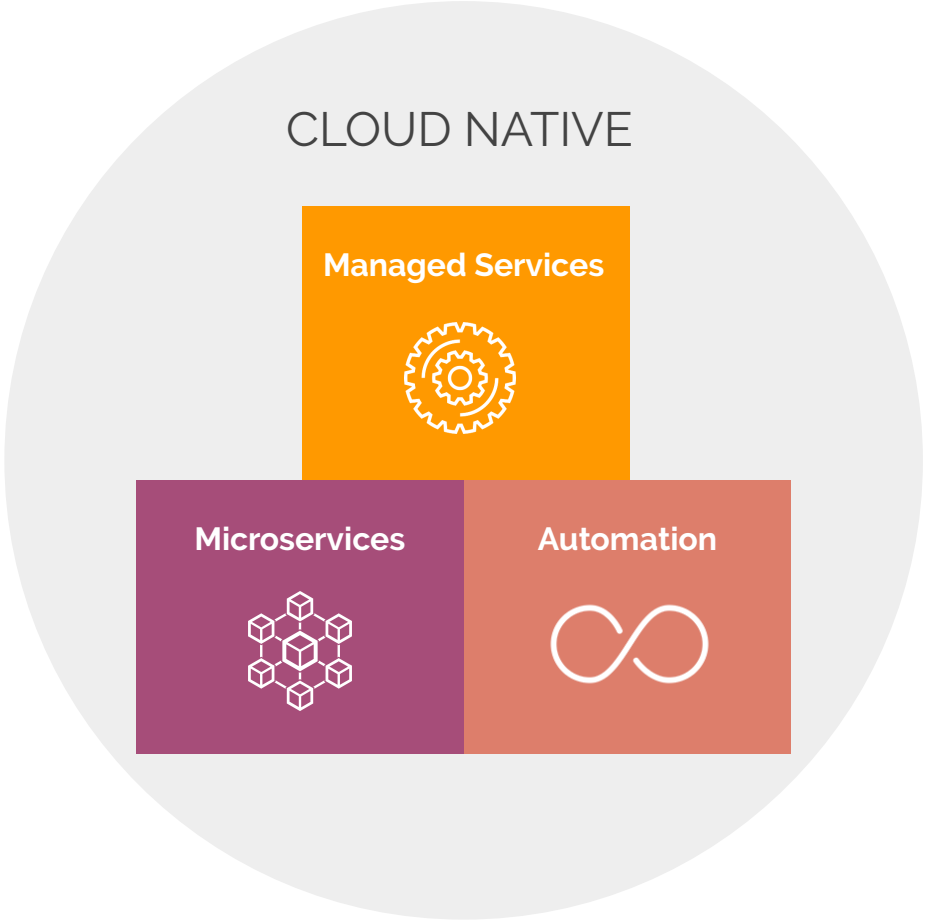
Journey to Cloud Native

LEARNINGS & BEST PRACTICES

QBurst

CLOUD NATIVE: THREE AXES

- ✓ Utilize managed services provided by the cloud provider
- ✓ Transition to a distributed, loosely-coupled, microservices architecture
- ✓ Adopt automation



CLOUD NATIVE JOURNEY: KEY OBJECTIVES

FOCUS	Reduce lead time for feature requests
DEVELOPMENT	Agile, DevOps
DELIVERY CYCLES	Continuous
ARCHITECTURE	Loosely coupled, microservices-based modular architecture
INFRASTRUCTURE	Container-driven, scalable horizontally on-demand, flexible infrastructure

MOVING UP THE CLOUD NATIVE MATURITY MODEL (CNMM)

Cloud Services

Start using basic cloud building blocks: Compute, storage, networking, monitoring.

Managed services: Databases, caching, directory services, load balancers, data warehouses, search.

Automation tools are available as managed services.

Use advanced services such as serverless and AI/ML services.

01 02 03

MOVING UP THE CLOUD NATIVE MATURITY MODEL (CNMM)

Application Design Changes

01

Start with 12-factor app design.

Make the application cloud-ready by utilizing basic cloud features/managed services (Lift and shift).

02

Convert Monolithic or SOA architectures to MSA.

Use serverless and event-driven approach where required.

Make the application cloud-optimized by utilizing the power of cloud managed services.

03

Adopt cloud native design with built in instrumentation, security, parallelization, and resiliency.

Comprehensive monitoring setup, CI/CD, Orchestration, microservice architecture, and use of cloud infra/managed services.

MOVING UP THE CLOUD NATIVE MATURITY MODEL (CNMM)

Automation

Environment management

- Immutable Infrastructure
- Infrastructure as Code
- Configuration management
- CI/CD

Monitoring, compliance, and optimization through automation.

AI/ML: Predicting failures, self-optimizing, self-healing.

01 02 03

BUILDING A CLOUD LANDING ZONE

Ensure all services are configured before deploying workloads.

Adopt cloud native engineering and best practices to maximize benefits.

- Account structure design (master and sub-accounts)
- Virtual network design (create subnets, isolate workloads and data)
- Centralize shared services (logging, directory services, auth services, monitoring, service catalogs)
- Security and audit requirements (logging framework, configuration audit process, snapshot of environment configuration at specific intervals)
- Set up an automation framework, Infrastructure as Code

DEVELOPMENT

- Develop or re-architect based on microservices architecture pattern
- Make services stateless
- Plan to utilize managed services such as **RDS, Aurora, DynamoDB, and Redshift.**
- Take advantage of elasticity — instance count changes with load
- Use serverless technologies such as **AWS Lambda, Azure Functions, and Kinesis**
- Place business functions behind APIs
- Automate tests — unit, APIs, acceptance

OPS

- Plan for Immutable infrastructure
- Build resilient services — if a service goes down, it should be easy to restart another one or redirect traffic to a working instance
- Infrastructure as Code (**Terraform, AWS CloudFormation**)
- Deploy services/applications in containers
- Use container orchestration tools such as **Kubernetes, Swarm**
- Implement an automated CI/CD pipeline
- Traffic management
- Use techniques such as **Blue/Green deployment**
- Find configurations drifts through automation
- Tagging strategy: Use standardized tags and implement them consistently across resources

STORAGE

- Storage lifecycle policy
- Organize data based on attributes such as frequency-of-access and planned retention period
- Enforce retention policies using code (and OS properties, where possible)
- Implement a cloud storage data aging management mechanism that tracks state of data and moves it to a different cloud storage device or deletes it after a defined lifecycle
- Automate backup

SECURITY

- Adopt DevSecOps approach
- Identity and Access management, detective controls, infrastructure and data protection
- Implementation of logic-based security solutions with custom scripting and monitoring
- Key/secret management
- Ensure continuous monitoring and threat prediction with stacks such as **ELK and OSSEC**
- Sensitive data encryption
- Harden servers and containers
- Environment/configuration drift detection
- Use managed services such as web application firewall
- Define cloud-based backup and disaster recovery strategy

COMPLIANCE & AUDITS

- Architect the solution based on applicable security standards
- Ensure inherent component level security, stronger interface security and resource life cycle management
- Create Compliance as Code framework to mitigate risks and facilitate smooth governance
- Automated compliance: Automate audit checks
- Logging
    - Logging strategy
    - Log centralization, correlating system and application level logs, dashboards
    - Real-time analysis and alerts on streaming logs, to detect anomalies, intrusion, etc.

MONITORING

EXTERNAL POLLING

White box monitoring: An approach that detects problems before they become externally visible and can provide valuable information for in-depth debugging.

CUSTOM METRIC COLLECTION

*Open source tools such as **Prometheus**, that integrate with Kubernetes, are effective in this space. It is a monitoring and alerting toolkit that stores metrics with a multi-dimensional time series database.*

CENTRALIZED LOGGING

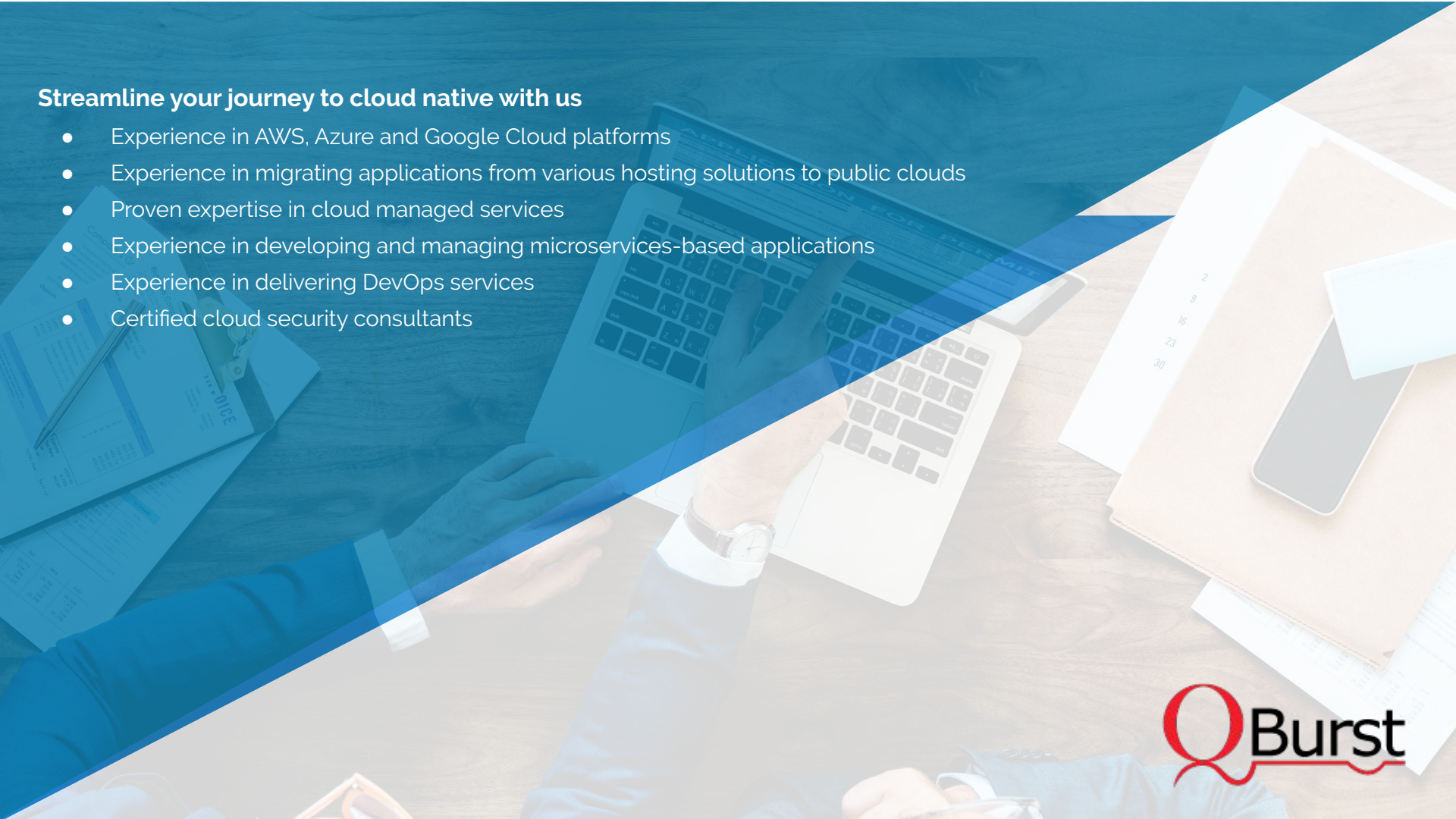
***ELK stack:** Components offer a set of open source tools for log storage, collection, and visualization respectively.*

REQUEST TRACING

*Enables end-to-end visibility across microservices - **Jaeger and Zipkin**.*

UNDERSTANDING THE BIG PICTURE

- Having the three axes in place is critical to the success of a cloud native journey
- Transition to cloud native should be driven by business needs
- Adopt cloud native engineering and best practices to maximize benefits
- The ability to deploy to production quickly and efficiently is an undeniable advantage
- Adopt a cloud native security platform that integrates reporting and compliance



Streamline your journey to cloud native with us

- Experience in AWS, Azure and Google Cloud platforms
- Experience in migrating applications from various hosting solutions to public clouds
- Proven expertise in cloud managed services
- Experience in developing and managing microservices-based applications
- Experience in delivering DevOps services
- Certified cloud security consultants

QBurst