CORTX, Containers, and Kubernetes:
The Why and The How

April 12th, 2022

Gregory Touretsky

Rick Osowski

Gregory Touretsky

Senior Director – Seagate

Gregory is a Senior Director at Seagate. He drives company’s Object storage SW and systems architecture. He has over twenty years of practical experience with distributed computing and storage as an architect, product manager, and systems engineer.

Gregory has an M.S. in Computer Science from Novosibirsk State Technical University and an MBA from Tel-Aviv University.

Rick Osowski

Principal Engineer – Kubernetes, Seagate

Rick pulls from over 18 years in the IT industry, with experience touching all the phases of Enterprise Software Development. His current focus is on the adoption of Kubernetes and Containers throughout CORTX.

Previously, Rick served as a Senior Solution Architect for IBM Cloud and led the Event-Driven Architecture domain with a focus on Apache Kafka-based reference architectures.

CORTX Solutions

OVA

- Pre-built single node VM image
- Functional demos for customers and partners
- Supported by the community

Community systems

- Open-source deployments
  - Full CORTX
  - IO Subsystem
- SW only
- Supported by the community

LYVE Rack

- Seagate-designed scale-out storage cluster (HW+SW+OS)
- Designed for Private Cloud / on-premises deployment
- Enterprise support by Seagate
- Built by Seagate

MSP - Cloud

- Managed cloud infrastructure
- Integrated and supported by the MSP admins team

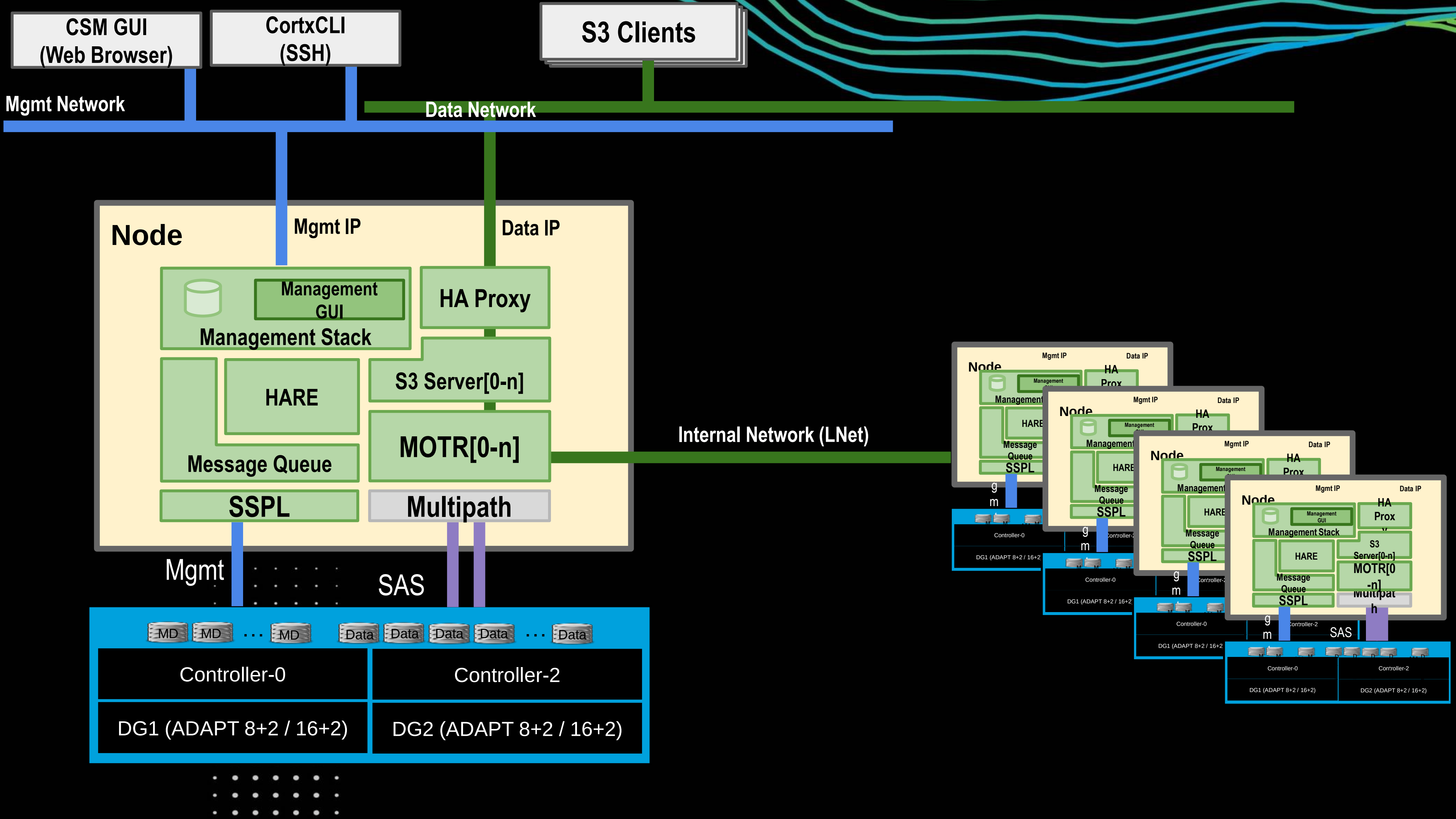
CORTX™

- S3-compatible object storage platform
- Software-defined storage
- Open source project under Apache2 license
- https://github.com/Seagate/cortx

License | Apache 2.0 code quality | A codacy-analysis-cli | passing

SEAGATE

CORTX Pre-historic (circa-2020)



SEAGATE

Key requirements

Adaptable

Resilient

Scalable

Easy to deploy

Easy to manage

Easy to troubleshoot

Easy to support

SEAGATE

Why Kubernetes?

Standard platform

Built for scale

Portable

Modular

Field-proven

Abstracts the infrastructure

Self-healing

Consistent deployment

SEAGATE

Kubernetes transition challenges

Skills

Fast evolving

Choice of technologies

Organizational inertia

Implementation strategy

SEAGATE

What It All Means…

CONTAINERS

A standardized application & dependency packaging method, which promotes mobility between many environments without change to the underlying application code.

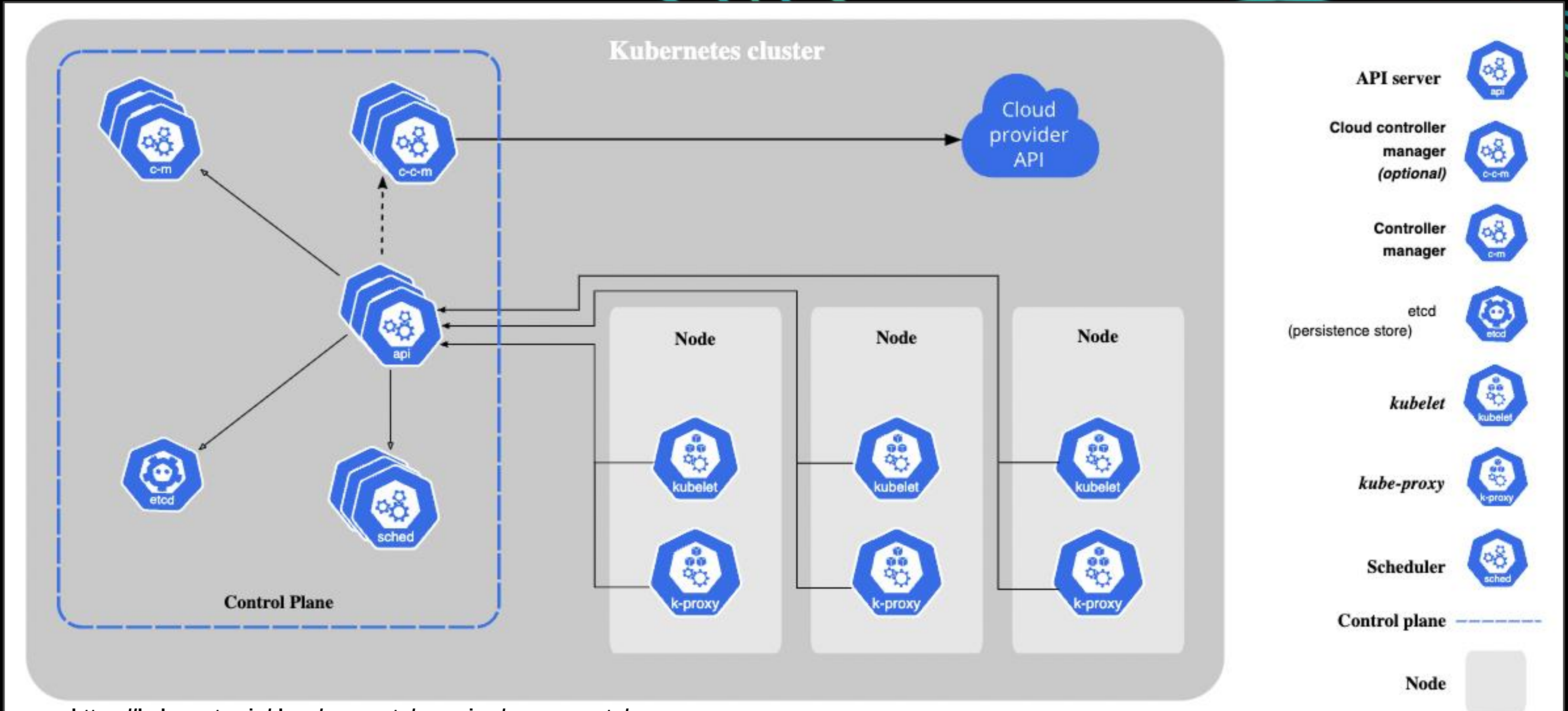
MICROSERVICES

An application architecture leveraging single-function modules owned by a single team that are loosely coupled, highly maintainable, and independently deployable with well-defined interfaces.

CLOUD NATIVE

An application architecture built to leverage the strengths and withstand the weaknesses of cloud environments, including elastic scaling, immutable deployments, and ephemeral underlying infrastructure.

Kubernetes 101

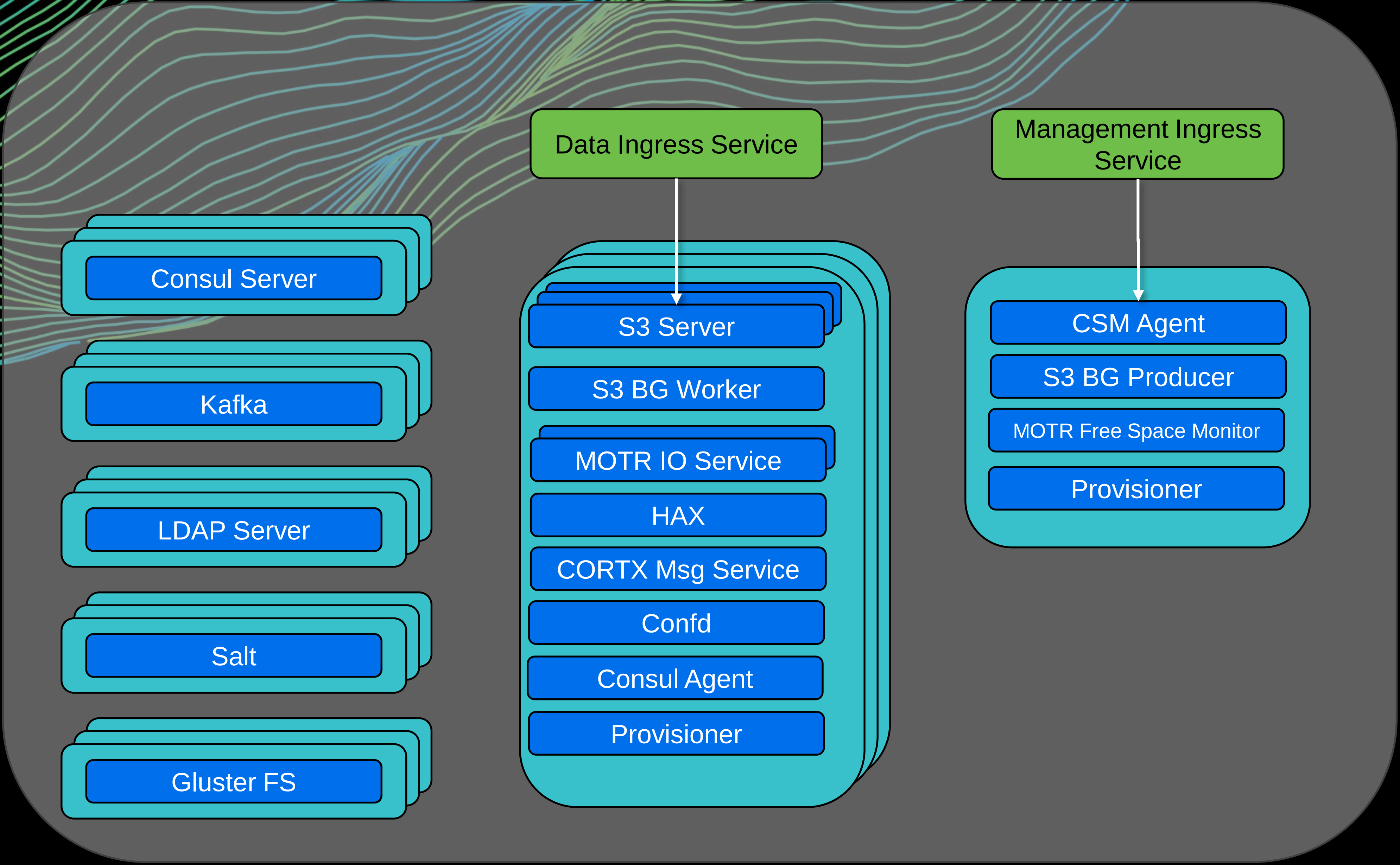


https://kubernetes.io/docs/concepts/overview/components/

It Is All About The Journey

Gen1

"Get it in the box!"



🟩 Kubernetes Svc

🟦 Kubernetes Pod

🟦 Container

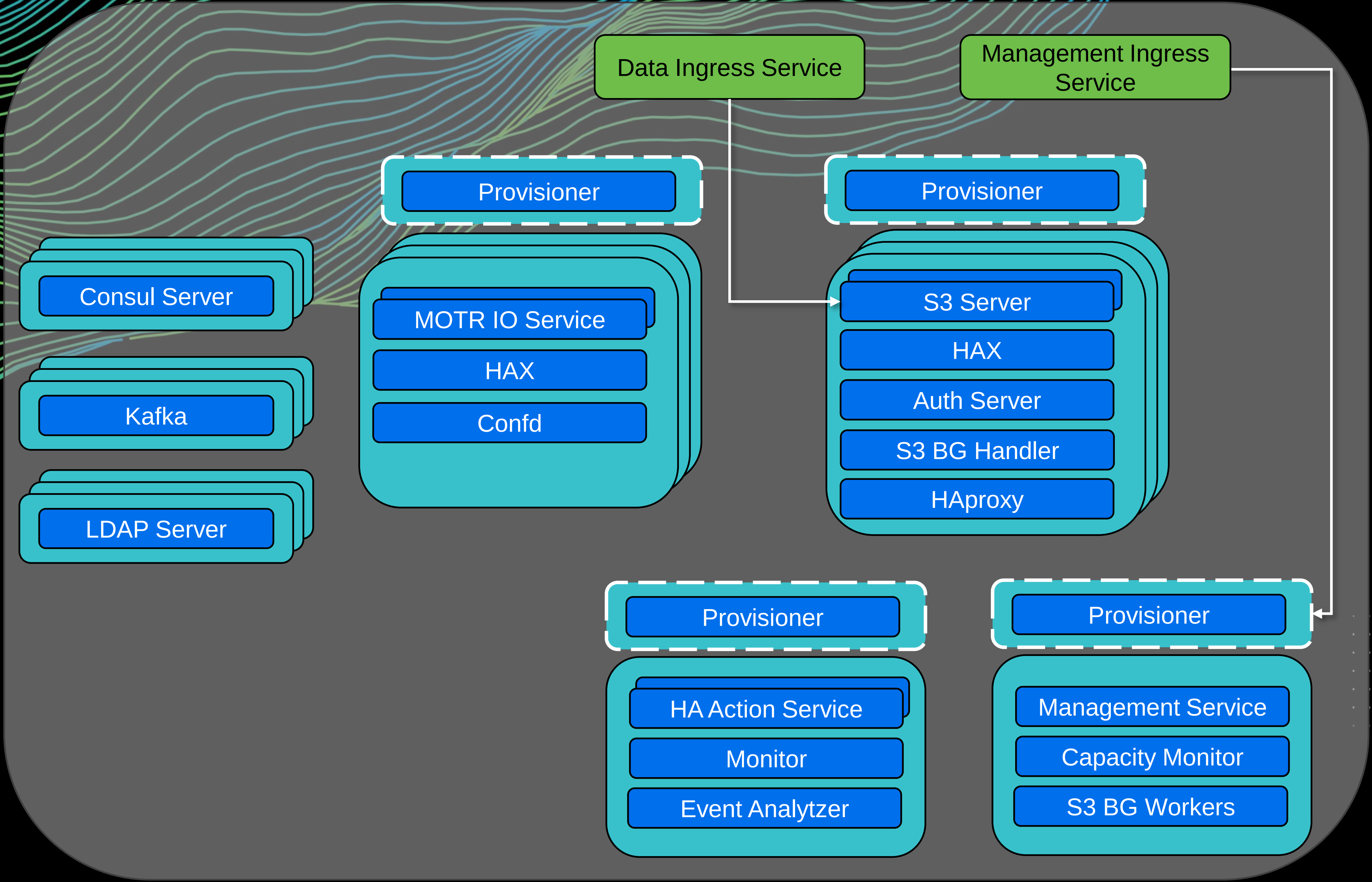
Key k8s concepts leveraged:

- Pods
- Services
- PVs

SEAGATE

Gen2

Kubernetes-enabled



🟩 Kubernetes Svc

🟦 Kubernetes Pod

🟦 Container

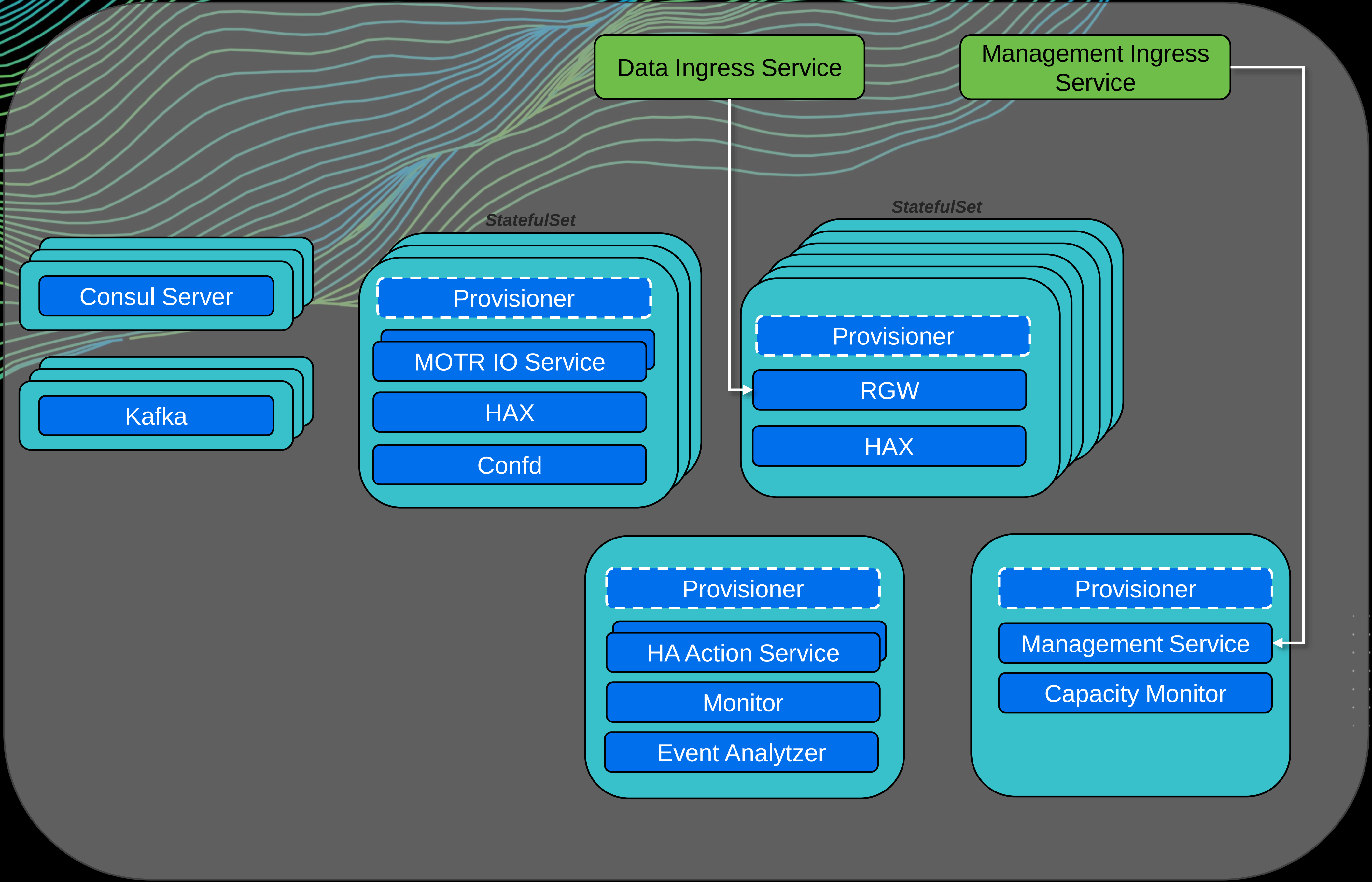
Key k8s concepts leveraged:

- Deployments
- Ephemeral Pods
- Fewer containers per Pod

SEAGATE

Gen3

Kubernetes-informed



Key k8s concepts leveraged:

- StatefulSets
- InitContainers
- Fewer containers per Pod

To Infinity… And Beyond!

Kubernetes-native

HELM CHARTS
Package Management for Kubernetes

OPERATORS
Application Management at Runtime

PROMETHEUS
Application Observability

ISTIO
Service Mesh

KNATIVE
Serverless Enhancements

CSI
Container Storage Interface innovations

COSI
Container Object Storage Interface innovations

Thank You!

Questions? Comments?

Join us on Slack at

https://cortx.link/slack_invite