WHITE PAPER

Architecting **cloud value**

A FinOps approach to driving bottom-line performance

bbd
software development

Contents

01 From cloud-first to cloud-correct

When companies initially rushed to the cloud, agility was the ultimate currency. Speed to market trumped efficiency, and provisioning was treated as an infinite resource pool. Today, software is no longer just a tool for business; it is the business.

However, market maturity demands a structural pivot. The early scramble to the cloud has left behind a legacy of financial fragility, where uncontrolled consumption quietly erodes margins. **The industry went fast. Now it needs to go right.** What should be a strategic accelerator is too often treated like an unpredictable utility bill to be paid and occasionally questioned.

The reality is that cloud spend directly impacts gross margin. If your architecture is inefficient, your business is inefficient. Every unoptimised line of code or poorly routed packet is a direct hit to your bottom line.

Reframing the cost conversation

True cost optimisation is not a periodic cost-cutting exercise or a reactive finance mandate. It is an engineering discipline that must be architected into workloads, pipelines, and governance structures from day one.

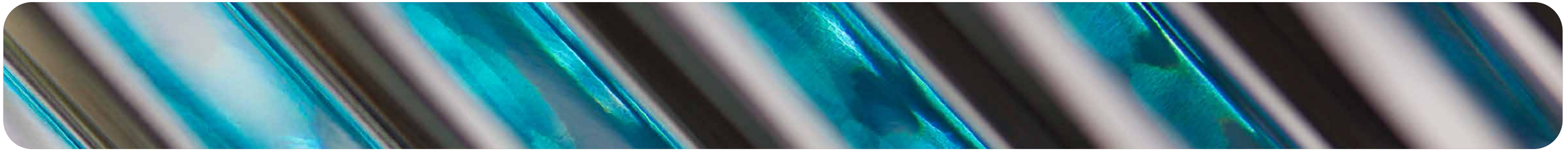
This white paper maps a practical path from financial fragility to fiscal precision. We examine where cloud spend silently leaks and how to eliminate it through deliberate architecture, layered commitment strategies, and security-enforced guardrails.

The goal is not cheaper cloud. It is a cloud that is as high-performing as it is cost-efficient: a direct, predictable driver of business value.

02 The path of least resistance (and cost)

As cloud environments mature, organisations face a jarring reality: data transfer costs routinely outpace compute expenditure. In a poorly architected network, routing data between your own services mirrors paying international roaming charges on your own infrastructure.

If data movement is not intentional, it is expensive. Placement, inspection, and routing decisions directly dictate your long-term operational expenditure.



The hidden cost drivers

Data movement inside AWS is rarely free. Unmanaged financial leakage typically originates from seven specific vectors:

01 **Cross-AZ traffic:** Inter-availability Zone data replication

02 **Transit gateway:** VPC-to-VPC data processing

03 **Internet egress:** Outbound data transfer to the public internet

04 **NAT gateways:** Per-GB processing fees on internet-bound or internal traffic

05 **Load balancers:** ALB/NLB egress and processing surcharges

06 **Hybrid connectivity:** Bulk data movement over Direct Connect or VPNs

07 **Centralised Inspection:** Forcing internal data through a single security checkpoint, creating costly U-turns

Strategic architecture & execution mandates

To optimise the network, architects must evaluate the specific financial implications of their technology choices. The table below outlines the primary optimisation vectors alongside mandatory design actions.

Architectural vector	Cost trigger	Structural remedy
Security inspection	AWS Network Firewall charges an aggressive per-GB processing fee	Project monthly GB throughput before selecting inspection technology. Use third-party appliances (e.g., FortiGate) to avoid AWS per-GB fees; you pay only for base EC2 infrastructure and licensing
Workload placement	Cross-AZ traffic introduces per-GB transfer charges and latency	Audit VPC topology for cross-AZ dependencies. Pin tightly coupled, high frequency microservices and databases to the same AZ
Internet egress	Pushing massive datasets or media files out of AWS to external environments	Relocate dependent downstream systems or compute nodes directly into AWS closer to the data source
NAT gateways	Standard NAT architecture taxes all internal AWS-to-AWS traffic per GB	Mandate VPC Endpoints for S3, DynamoDB, and ECR as a non-negotiable baseline. Limited internal AWS-to-AWS traffic should be permitted to traverse a NAT Gateway
Load balancing	High-volume ALB egress to the public internet	Edge Caching: Deploy Amazon CloudFront to absorb traffic at the edge and slash origin fetch requirements
Network Core (TGW)	Utilising Transit Gateway as a bulk transport mechanism between VPCs	Restrict TGW strictly to routing segmentation and hybrid connectivity. Keep high-bandwidth cluster traffic within the bounds of the VPCs

Cost-efficient architecture does not sacrifice security or scalability; instead, it designs for traffic awareness from day one.
If you are not architecting for proximity, you are paying for distance.

IN PRACTICE | TELECOMMUNICATIONS SECTOR

BBD assisted a major telecommunications client in migrating legacy back-end systems to AWS.

The migration enabled the adoption of Amazon Elastic Kubernetes Service (EKS), leveraging elastic scaling through Karpenter and EKS to optimise infrastructure costs.

Third parties were able to connect directly from their own AWS environments using VPC interface endpoints (AWS PrivateLink), providing low-latency, secure private connectivity. This eliminated the need for traffic to traverse NAT Gateways or external connectivity paths such as AWS Direct Connect, significantly reducing data transfer and networking costs.

The result: A highly predictable, governed network topology that actively prevents cost compounding as new workloads are onboarded.

READ THE FULL CASE STUDY →

03 Plan ahead, save behind

Most organisations treat cloud pricing as a reactive, post-billing activity. This passive posture drives a highly avoidable form of cloud waste: paying premium on-demand rates for predictable, steady-state production workloads.

Shifting from buying elasticity to buying predictability is an architectural decision, not a financial one. Understanding a workload's consumption profile well enough to commit to it reflects engineering maturity. When executed correctly, commitment-based pricing does not constrain agility. It funds it.

The AWS commitment toolkit

AWS Savings Plans trade a dollar-per-hour spend pledge (e.g., $10/hour for 1 or 3 years) for steep, automated compute discounts.
Two primary archetypes dictate design strategy:

COMPUTE SAVINGS PLANS (MAXIMUM FLEXIBILITY)

Automatically apply across all EC2 instance families, regions, operating systems, AWS Fargate, and AWS Lambda. They deliver up to a **66% discount** and are ideal for dynamic, containerised architectures

EC2 INSTANCE SAVINGS PLANS (MAXIMUM DISCOUNT)

Restricted to a specific instance family within a specific region. They yield higher discount percentages but require a fixed, stable infrastructure architecture

DATABASE SAVINGS PLANS

Applies to most managed and serverless database instance types potentially saving 52% on 3-year leases

Governance, baseline analysis & risk mitigation

Commitment without ongoing structural control erodes its own value. To prevent the financial trap of over-committing to workloads that may later shrink or modernise, engineering teams must adhere to three core operational mandates:

RIGHTSIZING MUST PRECEDE COMMITMENT

Rightsizing initiatives (more on this in Chapter 4) and continuous consumption monitoring (Chapter 7) must happen before a financial pledge. Committing to inflated infrastructure permanently locks in operational inefficiency

COMMIT TO THE FLOOR, NOT THE CEILING

Analyse historical compute data to isolate the absolute minimum steady-state usage. Exclude seasonal peaks and variable environments to incrementally build a baseline incrementally

CENTRALISE THE PURCHASE HIERARCHY

Purchase all Savings Plans strictly at the master payer account level. This ensures that unused commitment credits automatically float and share across linked sub-accounts, maximizing utilisation efficiency across the organisation. Review coverage metrics monthly against infrastructure lifecycles

IN PRACTICE | SHYFT

BBD designed and manage a cloud-native Forex trading platform for Shyft, built on a serverless-first architecture. The operational cost model demonstrates the efficiency of the layered commitment approach:

The elastic edge: Event-driven trading bursts are handled by serverless AWS Lambda functions on a pure consumption basis

The stable core: Persistent compute assets and database workloads leverage historical Amazon Cost Explorer usage patterns to secure predictable baseline costs via Reserved Instances and EC2 Savings Plans

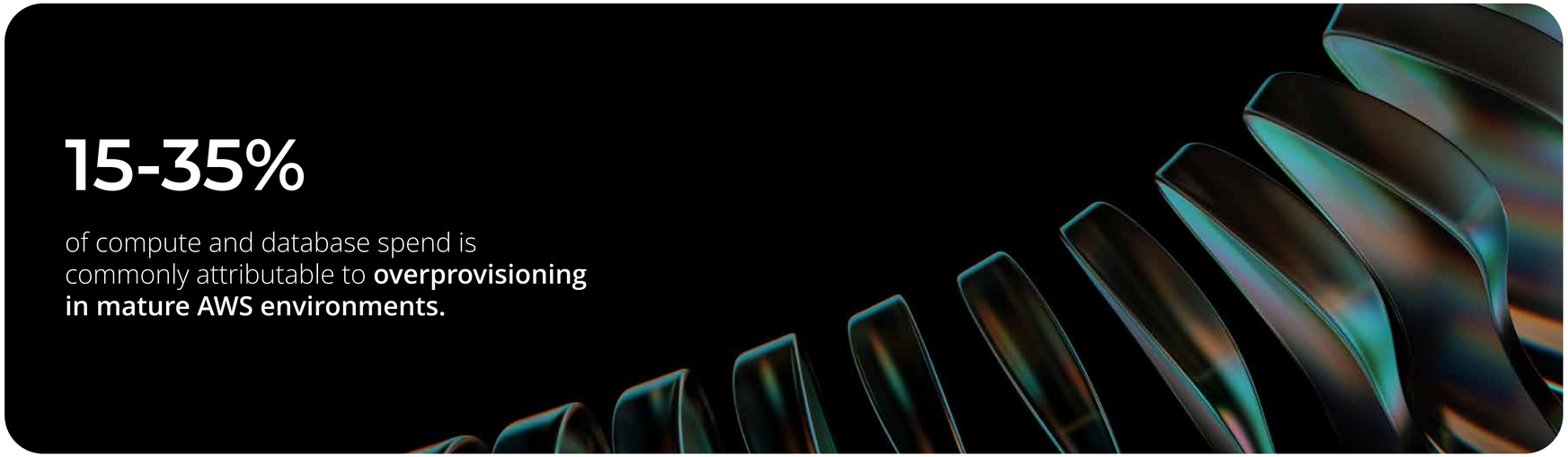
The result: Monthly SLA and cost trend reviews ensure that infrastructure commitments scale cleanly alongside business trajectory, delivering massive cost efficiency at scale without sacrificing the platform's ability to absorb sudden trading volume spikes.

READ THE FULL CASE STUDY →

04 Rightsizing done just right

Companies that have migrated their cloud but not their mindset often provision for the worst case and operate at average load. This leaves them with a heap of oversized instances running at a fraction of their allocated capacity, paying for compute resources that they don't ever use.

Rightsizing is not a one-time cost-cutting exercise; it is a continuous operational discipline that keeps infrastructure aligned with actual demand. Unlike commitment-based instruments such as Savings Plans, rightsizing focuses on eliminating waste before locking in the price.



15-35%

of compute and database spend is commonly attributable to **overprovisioning in mature AWS environments.**

PRO TIP

Sequence matters!

Optimise before you commit. Committing to an unoptimised environment locks you into financial inefficiency for the duration of the contract.

Platform optimisation strategies:

STANDALONE EC2 (SAVE 10 – 40%)

Stop paying for massive servers running at less than 20% capacity. Look at 1 to 3 months of data, shrink the server sizes to fit actual daily usage, and upgrade to newer, cheaper processor types (like Graviton)

EKS ON EC2 NODE GROUPS (SAVE 15 – 35%)

Stop letting applications claim huge blocks of server space they don't use. Force them to pack tightly onto fewer servers and use a system that only adds a new server when the existing ones are completely full

EKS WITH KARPENTER (AUTOMATED)

Ditch rigid, pre-sized server groups entirely. Karpenter acts like an automated dispatcher. The second an application needs to run, it instantly fires up a perfectly sized server, and deletes it the moment the job is done

ECS ON EC2 (SAVE 10 – 30%)

Similar to EKS, monitor how much memory and compute power your container apps actually use, trim their reservations down to size, and automate how your background servers scale

AMAZON AURORA DATABASES (SAVE 15 – 40%)

Remove duplicate backup database instances that aren't being read. Use a serverless setting that automatically stretches and based on real-time traffic so you never pay a high flat rate

AWS FARGATE (SAVE 5 – 25%)

Fargate charges a premium price for the exact size you request. Stop overestimating your application needs and rounding up your numbers, or you'll for idle space

AMAZON RDS DATABASES (SAVE 20 – 50% IN NON-PROD)

Shrink oversized databases to smaller classes. For testing and development environments, use cheaper types and set an automated timer to shut them off completely at night and on weekends

Quarterly reviews are too slow

It’s important to note that instance families change, workloads evolve, and Spot availability fluctuates on timescales that static reviews miss. Organisations must treat rightsizing as continuous infrastructure behaviour, not a reactive project.

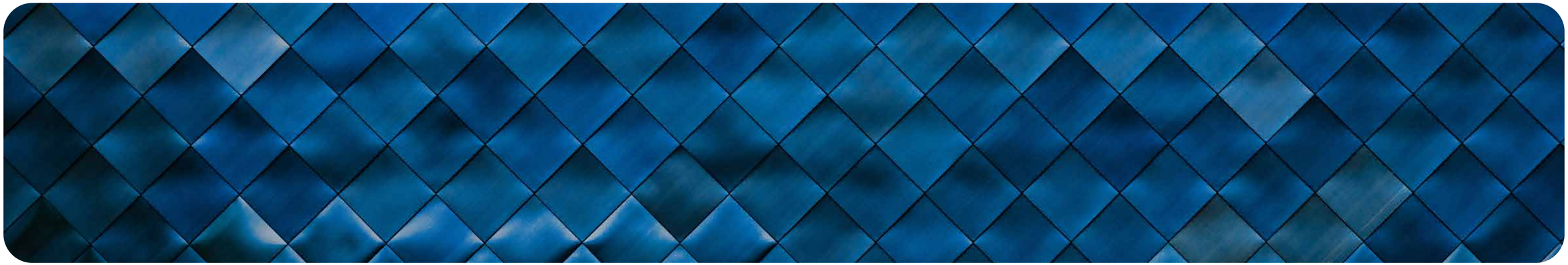
The objective is to establish clear rightsizing governance, prioritise the largest cost contributors first (EKS clusters, RDS fleets, EC2 fleets), automate via tools like Karpenter, and embed rightsizing as a foundational FinOps pillar alongside data transfer and Savings Plans governance.

IN PRACTICE | CELL C — MVNO PLATFORM SCALE

Cell C's migration of CDRator, the core system responsible for managing all MVNO subscribers to AWS, required rigorous capacity modelling. Handling 6% of South Africa's mobile subscribers, the platform demanded scale without the financial penalty of chronic overprovisioning.

BBD's architecture enabled Cell C to scale into managing 8 of 13 South African MVNOs while reducing new MVNO onboarding time from years to just 6 months. This agility was only achievable because capacity fit kept pace with growth. Ongoing managed services include capacity reviews and rightsizing as standard operational practice, ensuring the environment remains lean as it scales.

READ THE FULL CASE STUDY →



05 Cutting deadweight and silent waste

The 2 kinds of silent waste:

ORPHANED ASSETS

Resources not attached to active workloads that continue to bill indefinitely

NON-PROD OVERENGINEERING

Environments build to production standards that run around the clock when they only need to be operational during business hours

20-40%

of recoverable spend is typically represented by these two categories combined across mature AWS environments.

Orphaned assets

Waste accumulates naturally. Short-term projects are never decommissioned. Migrations leave behind their predecessors. Elastic scaling scales up but never fully scales back. Staff turnover creates ownership gaps. The result is an environment populated with resources that have no connection to any active workload, and no one who can account for them.

Tracking the hidden leakage

ZOMBIE EBS VOLUMES

When you delete an EC2 server, AWS leaves its attached hard drive (EBS volume) behind by default. It sits in the dark, doing nothing, while continuing to bill you every second

THE FIX

Lovok for unattached volumes. At standard gp3 pricing, a single forgotten 500GB volume quietly burns around $40 every single month

SNAPSHOT BLOAT

Backup policies without expiration dates create an infinite archive of ancient snapshots you will never use and frequently represents 15 – 30% of a company's total storage spend

THE FIX

Enforce strict retention limits. If a backup is older than x days and has no regulatory requirement, delete it

GHOST NETWORKING (NAT GATEWAYS & ALBS)

Load Balancers and NAT Gateways built for retired projects will happily sit empty, handling zero traffic, while billing you a flat hourly rate just for existing

THE FIX

Audit your subnets quarterly. If a load balancer has zero traffic, tear it down immediately

ORPHANED DATABASE INFRASTRUCTURE

Idle database reader instances, forgotten setups, and abandoned replicas can easily rack up bills

THE FIX

Map every single database to a confirmed, active application. If it has no owner, it has no business existing

Non-production overengineering

Non-production environments (Dev, Test, QA, UAT, and Staging) are consistently among the most bloated components of enterprise cloud estates. In many organisations, they consume massive amounts of total cloud spend despite sitting completely idle outside business hours.

The objective of non-production is functional validation and testing. It is not business-critical runtime infrastructure. Production-grade resilience is a premium cost that non-production does not need to bear. A non-production environment running at 3am on a Saturday is not a safety net. It is an invoice. An orphaned EBS volume is not a forgotten resource. It is a governance failure. Both represent the same thing: paying for something that delivers nothing.

IN PRACTICE | RETAIL SECTOR

BBD's engagement with a leading retail client included implementing automated tag enforcement through AWS Config rules. The system was configured to automatically block the creation of any new resource lacking an Owner, Environment, Workload, or Cost Centre tag, effectively stopping orphan accumulation before it could even start.

Simultaneously, automated non-production shutdown schedules and Single-AZ deployment standards were baked directly into the client's landing zone governance model. This structural change delivered massive, immediate non-production cost reductions while tightening the client's overall security and compliance posture across their entire AWS estate.

READ THE FULL CASE STUDY →

06 Guardrails, governance and setting boundaries

Every chapter in this white paper describes a category of cloud waste. But waste does not accumulate randomly. It accumulates wherever governance is absent. Oversized instances, orphaned assets, and ballooning log files all share a single root cause: the absence of enforced boundaries at the exact moment an engineering decision is made.

The core principle is simple: if engineers can manually click around and build whatever they want in your production console, your cost control is already compromised. Security and cost management are not competing priorities; they are the exact same automation system viewed from different angles.

The rule of governance:
If you cannot govern it,
you cannot optimise it.

The four pillars of preventative governance:

01 Infrastructure as Code (IAC)

Ensure that no resource enters your environment without a code review and a pipeline trail. IaC makes every infrastructure decision visible before it hits your production bill. A pull request that incorrectly provisions a massive instance or leaves off critical tags can be stopped and fixed before it ever costs you a cent

02 Enforced “Tag-or-Deny”

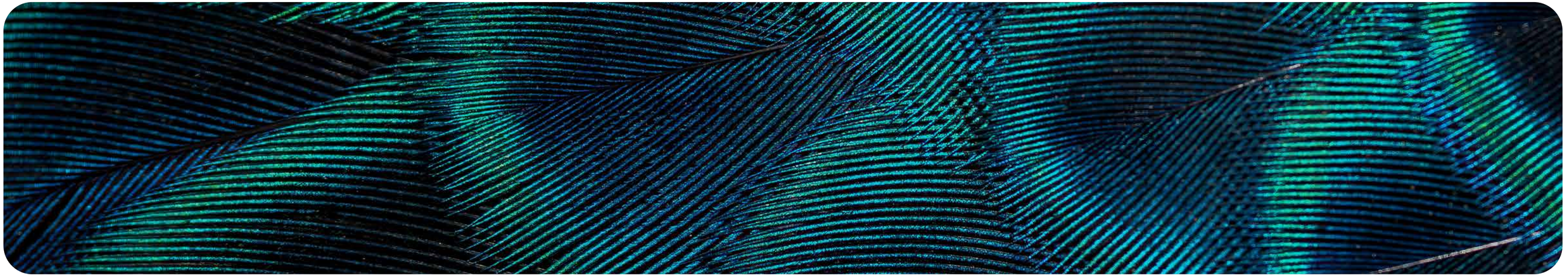
Untagged resources are unmanageable resources. If you don’t know who built a server or what application it belongs to, you can’t attribute its cost or safely clean it up. In a Tag-or-Deny setup, if a new resource attempts to launch without a demarcated foundational tag, environment outright blocks its creation

03 Service Control Policies (SCPS)

SCPs sit above everything, including your root administrators, and cannot be bypassed. If an SCP says “No,” no identity in that account can override it. This lets you lock down boundaries before bad spending habits can even start

04 Strict role separation

Long-lived administrator access in production accounts is a major financial risk. Keep your roles structured to ensure the right people have the right access to what they need



Curbing the “Set and Forget” consumption surcharges

A major category of AWS services operates on a pure consumption model – billing you strictly by the event, the byte ingested, or the API call made. They are cheap to start with, incredibly easy to scale, and entirely capable of triggering massive billing surprises if left unchecked.

CLOUDWATCH LOGS

The most frequent offender. When teams turn on aggressive trace logging, or let high-frequency Lambda functions run wild, log ingestion costs can quickly rival your actual compute spend

THE FIX

Implement strict retention caps, standardize on production INFO levels rather than verbose debugging, and scope VPC Flow Logs tightly at the subnet level rather than across the entire VPC

AWS CONFIG

Charges you for every single resource change it records. In high-churn environments like EKS clusters or aggressive CI/CD pipelines, recording every microscopic change can generate thousands of dollars in unnecessary metadata tracking

THE FIX

Limit the scope of recorded resource types in sandbox and testing accounts

S3 AND SNAPSHOTS

Unmanaged backups grow silently forever

THE FIX

Apply mandatory lifecycle rules that automatically expire old object versions and move older datasets to cheaper storage classes, while enforcing hard retention limits on all EBS and database snapshots

LOAD BALANCERS & KMS KEYS

High request volumes and excessive encryption calls beyond the free tier quietly add up at scale

THE FIX

Tear down load balancers with zero active backend traffic, use CloudFront edge caching to absorb public internet requests, and audit key management calls monthly

Governance is not a tax on engineering velocity. It is the engineering practice that makes velocity sustainable. Every guardrail that prevents an untagged resource from being created is simultaneously preventing a future orphan, a compliance finding, and an unexplained line item on next month's bill.

IN PRACTICE | SHYFT

For the Shyft Forex trading platform, BBD implemented a comprehensive observability stack using Amazon CloudWatch, Grafana, Loki, and Prometheus, with cost-containment guardrails baked in from day one.

Instead of funnelling raw, verbose data into high-cost storage tiers, Grafana Loki was configured to handle log aggregation across highly defined retention buckets, dramatically dropping baseline CloudWatch storage costs. Prometheus metrics are published selectively, intentionally filtering out high-cardinality data labels that traditionally trigger massive custom metric surcharges. Monthly reviews treat observability data ingestion as a governed, elastic variable, providing the engineering team with complete runtime visibility while maintaining monitoring costs as a tightly bounded, highly predictable line item.

READ THE FULL CASE STUDY →

07 Fast forward: Autonomous cloud and beyond

The chapters of this white paper have mapped the terrain of cloud waste. Each represents a domain where architectural discipline transforms cloud spend from an unpredictable overhead into a predictable margin driver. Taken together, they constitute not a temporary cost-cutting project, but a permanent corporate posture.

The trajectory of cloud engineering points toward autonomous cost governance: environments that detect, respond to, and correct cost inefficiencies entirely without human intervention. This is no longer a future aspiration; the building blocks are already live.

The disciplines that compound

Organisations that reach autonomous cost governance do so because they built the foundational layers first: their workloads are well-known, their infrastructure is clean, and their governance is tight. When these elements connect, optimisation shifts from manual intervention to algorithmic automation:

SELF-MANAGING COMPUTE	Systems like Karpenter already provision and deprovision compute dynamically based on real-time application demand signals, eliminating human guesswork
INTELLIGENT ANOMALY DETECTION	Machine learning models within AWS Cost Anomaly Detection actively flag spend deviations against learned historical baselines the moment a spike occurs
AUTOMATED POLICY-AS-CODE	Guardrails like AWS Config and Service Control Policies (SCPs) actively block non-compliant, over-budget, or out-of-region infrastructure from being created in the first place
THE NEXT HORIZON	The industry is moving toward self-healing loops where policy-as-code automatically cleans up anomalies, ML-driven commitment tools update Savings Plans dynamically, and observability pipelines feed financial telemetry directly back into deployment pipelines

The journey from cloud-first to cloud-correct is the absolute prerequisite for the journey to cloud-autonomous.

The continuous value engine: Managed Services

For organisations without the internal engineering capacity to construct and run this automated posture, BBD's Cloud Managed Services practice delivers it as a continuous, turn-key operational standard. We imbed cost control directly into the architecture layer rather than reporting on financial damage weeks after the invoice arrives.

IN PRACTICE | CELL C

Cell C's architectural journey is the definitive expression of the engineering-led FinOps philosophy. The partnership with BBD delivered far more than cloud infrastructure; it established the core capability to operate efficiently at massive scale.

By migrating CDRator, the system responsible for managing all MVNO subscribers, to AWS, BBD designed a highly optimised topology capable of supporting 8 of 13 South African MVNOs. This architecture radically streamlined operations, enabling Cell C to reduce new operator onboarding timelines, while equipping their internal teams to maintain a lean, highly agile cloud footprint.

"Our partnership with BBD has played a crucial role in empowering our teams to think differently and scale rapidly. Together, we are not just adopting technology, we are using it to improve the lives of our customers by delivering seamless and efficient services." – Schalk Visser, Chief Technology Officer, Cell C

READ THE FULL CASE STUDY →

08 The bottom line

The cloud's promise has always been correct. Elastic scale, global reach, and the ability to focus engineering effort on product rather than infrastructure are real and achievable. What the initial wave of cloud adoption missed was the discipline required to realise that promise economically.

Financial fragility is not a cloud problem. It is an engineering practice problem. **Organisations that treat cost as a non-functional requirement, alongside performance, reliability, and security, build cloud environments that behave like the strategic asset the cloud was always meant to be.**

The path from cloud-first to cloud-correct runs through the decisions described in this paper: none of which is individually complex, but together constitute an engineering culture that builds fast, wastes less, and thinks ahead.

Let’s build what’s next

Whether you’re migrating, modernising or scaling, we’ll help you build a secure, reliable cloud foundation that gives you clarity, control, and long-term value – or get in touch with our team to find out more.

CLICK HERE FOR MORE INFO →

CLICK HERE TO CONTACT US →

bbd
software development