



AN EXPERT PERSPECTIVE

Data Governance in AWS

Establishing a Data Perimeter Across a
Multi-Account Organization

Ivaylo Bahchevanov, Principal Cloud Architect

Phi Partners Cloud Centre-of-Excellence, Sofia, Bulgaria

September 2026, Whitepaper

Contents

1. Executive summary	4
2. Introduction	5
2.1 Context	5
2.2 The governance challenge on AWS	5
2.3 Approach and objectives	6
2.4 Scope	6
3. Data governance model	7
3.1 Guiding principles	7
3.2 Reference frameworks and standards	7
3.3 Core capability domains	7
3.4 Governance operating model	7
3.5 Roles and decision rights	8
4. AWS governance architecture	8
4.1 Multi-account foundation and guardrails	8
4.2 Where data resides	9
4.3 Governing data in place	9
4.4 Access model	10
4.5 Governance lifecycle	10
4.6 External integrations	10
5. The data perimeter in an AWS Organization	11
5.1 The three dimensions of the perimeter	11
5.2 Enforcement follows visibility: six control objectives	11
5.3 Policy mechanisms	12
5.4 Trust conditions	12
5.5 Exceptions	12
5.6 Progressive enforcement	13
6. Applying the data perimeter	13
6.1 How policies attach and inherit	13
6.2 Nested perimeters: One enterprise boundary, many secure zones	13
6.3 The enterprise perimeter	14
6.4 Secure zones: Org-internal perimeters	14
6.5 Shared services and cross-zone dependencies	15
6.6 Enterprise perimeter and secure zones compared	15
6.7 Design considerations	16
7. Implementing the data perimeter: A worked example	16
7.1 Scenario and organization layout	16
7.2 Control allocation	17
7.3 Layer 1 - The enterprise baseline	17
7.4 Layer 2 - The EU secure zone	18
7.5 Narrowing the shared-account exceptions	20
7.6 Network segregation and the dedicated endpoint account	21
7.7 Regional restriction and data residency	23

7.8 Exceptions for shared resources	25
7.9 Every request must pass both layers: how the two combine	26
8. Operating the data perimeter	27
8.1 Ownership and decision rights	27
8.2 Every exception needs a governed lifecycle	27
8.3 Change control	28
8.4 Conflict resolution	28
8.5 Monitoring and assurance	28
8.6 Drift erodes the perimeter without any policy changing	28
8.7 Evidence for audit	28
8.8 Automation and Policy as Code	29
9. Extending beyond the Organization: External and hybrid trust	30
9.1 External partners	30
9.2 Hybrid and on-premises access	30
10. Conclusion: From blueprint to operation	31
Appendix A - Policy reference	34
A.1 Data-perimeter policies	34
A.2 Encryption controls	34
A.3 Key-management lifecycle	37
A.4 Baseline hygiene	38
Appendix B - Governance model detail	40
B.1 Guiding principles	40
B.2 Core capability domains	41

1. Executive summary

An analyst in the EU secure zone is asked to share a quarterly customer extract with a colleague in the group finance team. The quickest route is to copy the objects from the EU Data account's bucket into a bucket the finance team owns in an account under the General Workloads OU. That bucket is also in eu-west-1, so the data never leaves an approved region. Every IAM policy allowed the request: the analyst's role could read the source bucket and write to any bucket in the organization, and the destination bucket's policy accepted writes from organizational principals. The enterprise perimeter allowed it too. The destination belongs to the organization, so the `aws:ResourceOrgID` check matched. Nothing was misconfigured. What was missing was a rule that says data in the secure zone may only move to resources inside the secure zone.

That rule is the data perimeter. It is a preventive, coarse-grained boundary across three dimensions: identity (only trusted identities reach organizational resources), resource (organizational identities reach only trusted resources), and network (data is reached only through expected networks). It adds to least-privilege permissions; it does not replace them. This whitepaper shows how to build one across a multi-account AWS Organization and how to run it. The central idea is a nested model: one enterprise perimeter that every account inherits, with secure zones inside it for the workloads that need stronger boundaries for residency, regulatory, legal-entity or sensitivity reasons (Figure 6.1). Section 7 builds one such zone, for EU-resident data, using real service control policies and resource control policies.

Key findings for decision-makers:

- **IAM alone cannot stop a valid credential from moving data out.** Least privilege limits what a principal may do. Only a perimeter limits where the data can go.
- **One baseline, many zones.** A small, stable enterprise perimeter at the organization root, with secure zones for data residence, regulated entities or sensitive data, scales without fragmenting the model.
- **Exceptions are where perimeters fail.** A single account-level exception for a shared service can quietly re-admit an entire account. Narrow every exception to the actions the service needs and give it an owner and a review date.
- **Controls drift even when nobody changes them.** New services, condition keys and accounts change what an existing policy means. The perimeter needs an owner, monitoring and audit evidence, not a one-off deployment.
- **Adopt it in phases.** Start in monitoring mode, enforce the enterprise baseline once trusted identities, resources and networks are known, then add secure zones where the risk justifies them (Section 10).

How to read this paper.

- **Executives and risk owners:** Sections 1, 9 and 10.
- **Security and cloud architects:** Sections 5 to 7, with the policies in Appendix A.
- **Platform and operations teams:** Section 8.

Sections 3 and 4 set out the governance model and AWS architecture the perimeter builds on; readers familiar with DAMA-style governance can skim them.

2. Introduction

Data has always mattered. What has changed is how widely it is spread and how many paths lead out of it. A single enterprise on AWS can hold data in many accounts, services and regions, and each of them has its own access controls. This section explains why that makes consistent governance hard and sets out the model's answer: govern data where it resides, not only after it has been consolidated.

2.1 Context

On AWS, data may reside across multiple accounts, organizational units, regions, applications, databases, object stores, analytics platforms, streaming services, backups, and third-party integrations. It is not only business datasets but also configuration, logs, audit files and scripts that can carry business data. Different business domains own different datasets, while shared services and centralized platforms provide common capabilities across the organization. This distribution creates a fundamental challenge: **how can an organization maintain consistent governance and assurance when data is intentionally distributed rather than centralized?**

Traditional approaches often assume that governance can be concentrated around a central data lake, warehouse, or catalog. Such platforms play an important role, but they do not represent the entire estate - transactional databases, application stores, backups, operational systems, files, and event streams continue to hold data outside them. Effective governance must therefore establish common policies and control objectives that apply wherever organizational data resides, while allowing individual business and technical domains to operate the services appropriate to their needs.

2.2 The governance challenge on AWS

AWS provides a broad set of services for storing, processing, sharing, and analysing data. This flexibility enables highly scalable and specialized solutions, but it also enlarges the governance surface. Several challenges recur as an AWS estate grows:

- **Distributed ownership** - data is owned and managed by different business domains and technical teams.
- **Multiple accounts and organizational boundaries** - workloads are separated for security, operational, regulatory, or business reasons.
- **Different data services** - each service provides its own security, access-control, and configuration mechanisms.
- **Cross-account and cross-domain access** - legitimate business relationships require data to move between otherwise separate trust domains.
- **External integrations** - users, applications, partners, and enterprise platforms may operate outside the immediate AWS environment.
- **Regulatory and residency requirements** - some data requires stronger controls over where it is stored, who can access it, and through which networks it can be reached.
- **Configuration drift** - controls that are correct at deployment can become ineffective as resources, policies, and integrations change.

Because of this, data governance cannot be reduced to a catalog, an access-control policy, or a collection of security tools; it requires a model that connects organizational accountability, governance policy, architecture, preventive controls, and continuous operational assurance.

2.3 Approach and objectives

The model combines two perspectives. The first is data governance as an organizational capability: ownership, stewardship, policies, standards, decision rights, quality expectations, lifecycle requirements, and compliance responsibilities. The second is AWS governance as an enforceable technical architecture: using AWS Organizations, identity and network controls, service-native protections, monitoring, and audit mechanisms to implement and continuously enforce those requirements. The data perimeter connects the two, translating the requirement to retain control over organizational data into an enforceable trust boundary around identities, resources, and networks.

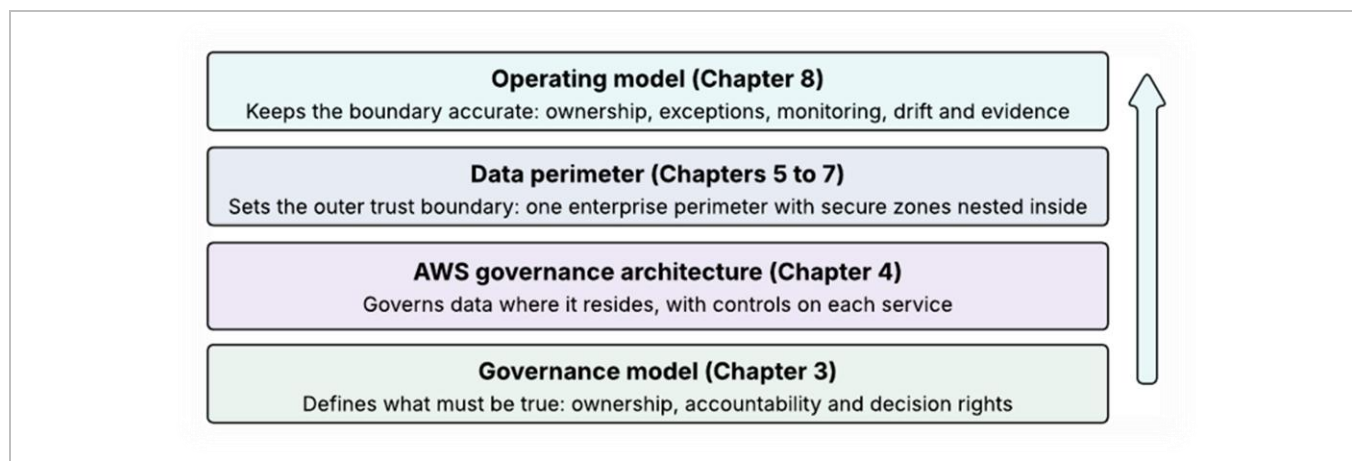


Figure 2.1 - The layered model. Each layer builds on the one below it; the data perimeter adds the trust boundary that service-level controls alone cannot provide.

Within that approach, the objectives are to:

- establish a consistent enterprise data governance model with clear ownership, accountability, and decision rights;
- govern data across distributed AWS services rather than relying on centralization;
- establish preventive guardrails that reduce the risk of unintended data exposure;
- enforce least-privilege access for human and machine identities;
- define trusted identities, trusted resources, and expected networks;
- support stronger isolation for sensitive, regulated, or geographically constrained workloads;
- make legitimate exceptions explicit and governable;
- provide continuous monitoring and evidence of control effectiveness through a repeatable pattern that can evolve as the estate grows;

The intent is not to prescribe a single AWS architecture for every workload, but to define a governance pattern that adapts to different organizational structures, data domains, regulatory requirements, and service choices.

2.4 Scope

The whitepaper covers data and data-holding workloads within an AWS Organization, from the governance model through to the perimeter and how it is operated. It does not provide detailed implementation guidance for every AWS service or replace service-specific security documentation. The controls described establish an architectural and governance pattern intended to be translated into service-specific implementation standards.

The paper is written for security and cloud architects, platform and operations teams, and the risk, compliance and data leaders who own the outcomes. It assumes working familiarity with AWS Organizations and IAM.

3. Data governance model

This section sets out the governance foundation in brief: the outcomes effective data governance must achieve, independent of AWS. They are the criteria against which the AWS architecture and the data perimeter are assessed in later sections. Full definitions are in Appendix B.

3.1 Guiding principles

Seven principles guide the design decisions that follow. **Data is an accountable asset** with named owners. **Standards are set centrally, while stewardship is federated** to the domains that know the data best. **Controls are built in by design** rather than added later. Data is **transparent and traceable**, **quality is fit for purpose**, protection is **proportionate to sensitivity**, and data is governed across its **whole lifecycle**. Appendix B.1 defines each principle.

3.2 Reference frameworks and standards

The model does not introduce a proprietary methodology. It draws on established bodies of knowledge - DAMA-DMBOK [1], the EDM Council's DCAM [2], ISO/IEC 38505-1:2017 [3], and COBIT [4] - for data-management practice, capability-maturity assessment, data-specific governance principles, and enterprise technology-risk objectives, respectively.

Mandatory outcomes are set by regulatory and legal requirements such as GDPR and applicable data-residency and sector-specific obligations [5]. These are reference points rather than implementation prescriptions; the objective is to translate their relevant principles into practical capabilities and controls.

3.3 Core capability domains

The model covers eight interdependent capability domains: ownership and stewardship, metadata, data quality, security and privacy, master and reference data, lifecycle, regulatory compliance, and operational governance. Appendix B.2 defines each one.

The domains reinforce one another: classification informs security controls, ownership determines accountability for quality, and lifecycle requirements drive retention and disposal. Three of them (classification, security and privacy, and lifecycle management) are implemented directly on AWS through the "governing data in place" control families in Section 4.

3.4 Governance operating model

The model follows a federated approach. Enterprise-level governance defines common policies, standards, control objectives, and decision frameworks, while individual business and data domains retain responsibility for applying those requirements to the data they own. In short, the central function defines *what must be true*, and domain teams determine *how* it is achieved within their business context, subject to enterprise controls. This balances enterprise consistency with the local knowledge needed to govern data well.

3.5 Roles and decision rights

A federated model typically distributes accountability across the following roles:

- **Data Governance Board** - approves enterprise policies and standards and resolves issues spanning multiple domains.
- **Chief Data Officer** - holds executive accountability for the data strategy and the effectiveness of the governance program.
- **Data Owners** - accountable for defined data domains, including classification, quality, appropriate use, and access requirements.
- **Data Stewards** - manage definitions, metadata, quality issues, and day-to-day governance within a domain.
- **Data Custodians** - operate the technical platforms and implement controls on behalf of owners and stewards.
- **Data Consumers** - people and systems that use data within the boundaries established by policy.

Decision rights should be made explicit through a responsibility model such as RACI (responsible, accountable, consulted, informed). These roles recur in the operating model of Section 8, which applies them to the data perimeter rather than redefining them.

Key takeaway: Governance defines what must be true (ownership, accountability, and decision rights) before any AWS control is chosen. Every control in the sections that follow traces back to these outcomes.

4. AWS governance architecture

This section translates the governance model into a high-level AWS architecture built on a single principle: data is governed in place. Rather than consolidating all enterprise data into a central lake or platform, governance controls are applied to the AWS services in which data legitimately resides - transactional databases, object stores, analytics platforms, application services, backups, and other purpose-built systems. The architecture rests on two foundations: a multi-account AWS environment that provides organizational boundaries and enterprise guardrails, and governance controls applied directly to data-holding services.

One example organization runs through the rest of the paper. It keeps security, shared services and general workloads in their own organizational units (OUs), and has one regulated branch whose EU workloads must stay in the EU and be more tightly isolated.

4.1 Multi-account foundation and guardrails

The environment is organized using AWS Organizations and, where appropriate, AWS Control Tower. Accounts are grouped into OUs by functional, business, environmental, or geographic boundary; where residency or sovereignty requirements apply, that structure provides an additional boundary for the affected workloads. This multi-account structure is the primary mechanism for blast-radius containment and organization-wide governance. Its key controls are:

- **Service control policies (SCPs) and resource control policies (RCPs)** - organization-level guardrails that establish, respectively, the maximum permissions for principals and resource-side restrictions.
- **AWS Control Tower controls** - preventive, detective, and proactive controls that establish a common account baseline.
- **AWS IAM Identity Center** - the workforce identity layer, federated with the corporate identity provider.

- **Account baselines and tagging standards** - common metadata such as owner, environment, and data classification, which underpin governance and access decisions.

These controls establish the environment within which individual data services operate.

4.2 Where data resides

Enterprise data is distributed across many AWS services. The architectural point is not that these services need identical controls, but that they need *consistent governance objectives* implemented through service-appropriate mechanisms.

Data category	Representative services	Typical content
Object storage	Amazon S3, S3 Glacier	Files, data-lake objects, backups, logs, exports
Relational databases	Amazon RDS, Amazon Aurora	Transactional and application data
NoSQL databases	DynamoDB, DocumentDB, Keyspaces, Neptune	Key-value, document, wide-column, graph data
Data warehousing and analytics	Amazon Redshift, Amazon EMR, Amazon Athena	Analytical datasets and query workloads
Streaming and messaging	Amazon Kinesis, Amazon MSK	Events and streaming data
File and block storage	Amazon EFS, Amazon FSx, Amazon EBS	Shared file systems and attached storage
Search and caching	Amazon OpenSearch Service, Amazon ElastiCache	Search indexes and cached data
Backup and archive	AWS Backup, S3 Glacier	Backups and long-term retention
Metadata and catalogs	AWS Glue Data Catalog, Amazon DataZone	Technical and business metadata

Table 4.1 - Representative data-holding AWS services by category.

4.3 Governing data in place

Governance is applied to each data-holding service through a common set of control families:

- **Classification and tagging** - resources are assigned ownership and classification metadata; services such as Amazon Macie can discover sensitive information and support classification of Amazon S3 data.
- **Encryption** - data is encrypted at rest using AWS Key Management Service (KMS) and in transit using Transport Layer Security (TLS), with customer-managed keys and key policies providing an additional authorization layer for sensitive workloads.
- **Preventive guardrails** - the organization-level SCPs and RCPs restrict what may be done and by whom, while service-specific controls, AWS Config, and AWS Control Tower enforce and validate configuration requirements.
- **Service-level access control** - each service enforces least privilege through its native authorization mechanisms, including Amazon S3 bucket policies and Access Points, AWS Lake Formation permissions, IAM authentication for databases, and Amazon Redshift role-based and fine-grained access controls.

- **Network isolation** - data services are placed behind approved network paths and, where appropriate, accessed through virtual private cloud (VPC) endpoints and AWS PrivateLink; these controls complement rather than replace identity and resource authorization.
- **Detection and audit** - AWS CloudTrail, AWS Config, Amazon GuardDuty, AWS Security Hub, and AWS Audit Manager provide activity logging, threat detection, configuration assurance, and evidence.

4.4 Access model

Access to governed data is controlled through AWS IAM and evaluated across multiple policy layers, so that a request must satisfy all applicable organizational, identity, resource, and session-level controls. This fine-grained authorization operates *inside* a broader organizational trust boundary - the data perimeter; the two are complementary layers rather than alternatives.

Human access. Workforce users authenticate through AWS IAM Identity Center and receive centrally governed permission sets. Where appropriate, attribute-based access control uses principal and resource attributes such as ownership and classification to reduce the need for resource-by-resource authorization.

Workload access. Applications and AWS services use IAM roles and short-lived credentials obtained through AWS STS rather than long-lived access keys, with resource-based policies and condition keys constraining access to intended organizations, accounts, services, and resources.

Cross-account access. Cross-account access is explicitly governed through role assumption, resource-based policies, AWS RAM, Lake Formation grants, or other appropriate mechanisms. Least-privilege authorization determines what an approved principal can do; the outer trust boundary determines whether the principal is trusted at all.

4.5 Governance lifecycle

Because governance is applied where data resides, the lifecycle is embedded into service provisioning and operation: provision the service under baseline guardrails within its account and OU; classify it and assign an accountable owner; protect it with appropriate encryption, network, and service-level controls; grant least-privilege access to approved users and workloads; monitor configuration, access, and security activity continuously; and remediate violations while managing retention, archival, and disposal according to policy.

4.6 External integrations

The architecture must integrate with the surrounding enterprise ecosystem - corporate identity providers, on-premises and hybrid environments, enterprise data catalogs, business intelligence platforms, external data providers, and enterprise security information and event management (SIEM) and observability platforms. These integrations should preserve governance metadata, identity context, and control requirements rather than creating unmanaged paths around the governance model.

Key takeaway: Govern data where it resides, with fine-grained controls on each service. Those controls are necessary, but none of them sets an outer limit on trust that holds when an individual policy or configuration is wrong.

5. The data perimeter in an AWS Organization

Previous section governs individual data services through layered controls over access, encryption, network, and monitoring. The data perimeter adds one further layer: an organization-wide trust boundary around data that establishes the outer limits within which data may be accessed [6]. It complements least-privilege authorization rather than replacing it. The distinction is the foundation of the whole architecture - least privilege answers *what may this principal do?*, while the data perimeter answers *which identities, resources, and networks are trusted at all?* This section defines its dimensions, control objectives, and policy mechanisms.

5.1 The three dimensions of the perimeter

A data perimeter is a set of preventive, coarse-grained controls ensuring that only *trusted identities* access *trusted resources* from *expected networks* [7]. Each term has a precise meaning:

- **Identity perimeter** - only trusted identities can access organizational resources. Trusted identities are the principals (IAM roles and users) within the organization's AWS accounts, together with AWS services acting on the organization's behalf.
- **Resource perimeter** - organizational identities can access only trusted resources. Trusted resources are those owned by the organization's AWS accounts, together with resources owned by AWS services acting on its behalf.
- **Network perimeter** - data is reached only through expected networks. Expected networks are the organization's on-premises data centers and VPCs, together with the networks of AWS services acting on its behalf.

5.2 Enforcement follows visibility: six control objectives

The three dimensions resolve into six complementary control objectives. Each is enforced by whichever mechanism can inspect the relevant request, which is why the enforcing mechanism sometimes sits on the opposite side from the dimension's name - RCPs filter by the calling principal and network, VPC endpoint policies inspect both the principal and the target resource, and SCPs restrict the organization's own identities [7].

Dimension	Control objective	Primary mechanism
Identity	Only trusted identities can access organizational resources	RCPs
	Only trusted identities are allowed from expected networks	VPC endpoint policies
Resource	Organizational identities can access only trusted resources	SCPs
	Only trusted resources can be accessed from expected networks	VPC endpoint policies
Network	Organizational identities can access resources only from expected networks	SCPs
	Organizational resources can be accessed only from expected networks	RCPs

Table 5.1 - The six data perimeter control objectives and their primary enforcement mechanisms [7].

The exact implementation still varies by service, because policy support and condition-key availability differ across AWS services.

5.3 Policy mechanisms

The perimeter is implemented through three primary capabilities rather than a single policy [7]:

- **Service control policies (SCPs)** - AWS Organizations policies that set the maximum available permissions for the principals (IAM roles and users) in the organization. In the perimeter, they keep organizational identities accessing only trusted resources and operating only from expected networks.
- **Resource control policies (RCPs)** - AWS Organizations policies that set the maximum available permissions for the resources in the organization. In the perimeter, they keep organizational resources accessible only to trusted identities and only from expected networks. RCP support is service-dependent and must be assessed per service.
- **VPC endpoint policies** - policies attached to VPC endpoints that control which principals, actions, and resources can be reached through the endpoint. In the perimeter, they ensure a network path is used only by trusted identities to reach only trusted resources.

One distinction matters in practice: SCPs apply only to principals within the organization, whereas RCPs and VPC endpoint policies can also constrain principals outside it [7]. Where a service does not yet support RCPs, resource-based policies attached directly to the resource can enforce the equivalent resource-side controls.

5.4 Trust conditions

The policies express the trust model through AWS global condition keys, grouped by perimeter dimension [7]:

- **Identity** - `aws:PrincipalOrgID` restricts access to principals in the organization; `aws:PrincipalIsAWSService` and `aws:SourceOrgID` allow AWS services to act on the organization's behalf while guarding against the cross-service confused-deputy problem.
- **Resource** - `aws:ResourceOrgID` restricts access to resources that belong to the organization.
- **Network** - `aws:SourceIp`, `aws:SourceVpc`, `aws:SourceVpce`, and `aws:VpceOrgID` restrict access to expected network locations; `aws:ViaAWSService` and `aws:PrincipalIsAWSService` permit legitimate AWS service paths that originate outside those locations.

The organization must maintain the underlying trust definitions - organization identifiers, trusted accounts, VPCs, endpoints, and network ranges - as part of normal governance, since the perimeter is only as accurate as the definitions behind it.

5.5 Exceptions

A perimeter must accommodate legitimate exceptions, such as AWS service-to-service interactions, centralized enterprise services, approved external partners, shared data products, and cross-domain integration services. Each exception should be explicit and narrowly scoped so that accommodating one legitimate dependency does not weaken the perimeter globally. The full exception lifecycle is defined in Section 8.

In our EU zone, we define two exceptions for example: EU workloads must reach the shared networking account and the central image factory.

5.6 Progressive enforcement

The perimeter should be introduced progressively rather than switched on at once. A typical sequence establishes the organization and required prerequisites; identifies trusted identities, resources, and networks; analyses existing access patterns; deploys controls in monitoring or non-blocking modes where appropriate; identifies legitimate dependencies and introduces narrowly scoped exceptions for them; enforces the controls progressively; and then monitors and refines them continuously. Approached this way, the perimeter becomes an operating control rather than a one-time policy deployment.

Key takeaway: Least privilege decides *what* a trusted principal may do; the perimeter decides which identities, resources, and networks are trusted at all. Exceptions are expected, but each must be explicit and narrow, and enforcement should be phased in from monitoring mode.

6. Applying the data perimeter

Previous section has defined the perimeter's dimensions, control objectives, and policy mechanisms. This section addresses where those controls apply within the AWS Organizations hierarchy - and this is where the model earns its value in a large estate. The central idea is not that an organization chooses between a broad perimeter and a narrow one, but that it operates one enterprise perimeter and, nested inside it, one or more org-internal perimeters that ring-fence parts of the estate with stronger requirements. The rest of the section develops that nested model.

6.1 How policies attach and inherit

SCPs and RCPs can be attached to the organization root, to an OU, or to an individual account, and they are inherited by everything beneath the attachment point. Where policies attach at more than one level, their restrictions combine: a request must satisfy every applicable control on the path from the root down to the account. This inheritance behavior is what makes nesting possible, and it yields the governing principle for the whole section - attachment defines a perimeter's scope, while the policy's conditions define its trust boundary. The same type of control therefore means different things depending on where it is attached, which is precisely what lets one mechanism serve as both the outer enterprise boundary and an inner zone.

6.2 Nested perimeters: One enterprise boundary, many secure zones

Because perimeters inherit and combine, they nest. The outermost boundary - the enterprise perimeter - is attached at the organization root and establishes a single trust domain that every account and workload sits within. Inside it, an organization can define one or more secure zones: org-internal data perimeters, each attached to a branch of the OU hierarchy, that impose a narrower trust boundary on a subset of the estate to satisfy a specific requirement such as data residency, a regulatory regime, a legal-entity boundary, or a heightened sensitivity level.

The relationship is deliberately one-to-many. The enterprise perimeter is singular - there is exactly one outer boundary, and no account escapes it. Secure zones are plural: a large organization may run several at once, each with its own trust definition and its own set of exceptions, and each generally corresponding to a distinct compliance driver. A residency zone for EU-regulated workloads, a cardholder-data zone subject to payment-industry controls, and a zone for highly privileged workloads can all coexist inside the same enterprise perimeter without interfering with one another, as illustrated in Figure 6.1.

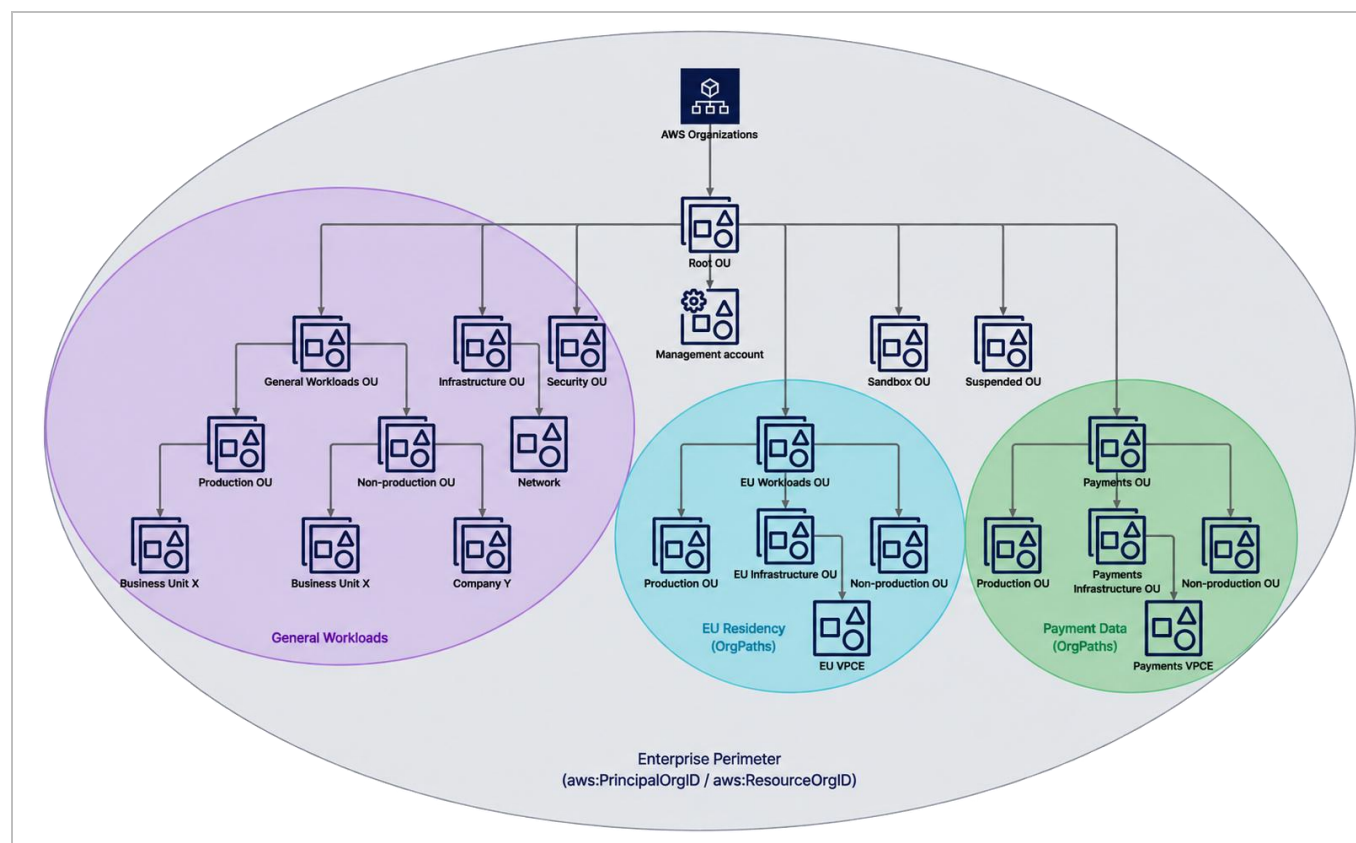


Figure 6.1 - One enterprise perimeter containing several independent secure zones. Every account is inside the enterprise boundary; only accounts within a zone's OU branch are additionally subject to that zone's narrower trust domain.

Zones are normally non-overlapping, each aligned to one branch of the hierarchy. Where a single requirement genuinely spans more than one zone, it is not expressed as a fourth, cross-cutting zone; instead, the same control is applied within each perimeter that must satisfy it - for example, enforced once at the enterprise layer so that it holds everywhere, or replicated in each zone that shares the requirement. This keeps every perimeter aligned to a clean branch of the OU tree while still enforcing common requirements across zones.

6.3 The enterprise perimeter

The enterprise perimeter is the outer boundary and the minimum governance baseline. Its controls are attached at the organization root, and its trust conditions are organization-wide: keys such as `aws:PrincipalOrgID` and `aws:ResourceOrgID` establish trust in any identity or resource belonging to the organization, while organization-owned VPC endpoints and approved corporate network ranges define the expected-network domain [7]. Its defining property is coverage - because the controls attach at the root and inherit downward, new accounts and OUs fall inside the baseline automatically, without depending on individual account teams to apply them. For that reason, the enterprise perimeter should be kept relatively small, stable, and technology-independent: it exists to guarantee that no account operates outside the enterprise trust domain, not to encode workload-specific rules, which belong in the zones.

6.4 Secure zones: Org-internal perimeters

A secure zone narrows trust from “the whole organization” to “this branch of it.” The mechanism is a shift in condition keys: where the enterprise perimeter uses organization-wide keys, a zone uses organizational-path keys

- `aws:PrincipalOrgPaths` and `aws:ResourceOrgPaths` - which match only the principals and resources within a specified organization root, OU, or its children [8]. An identity trusted by the enterprise baseline may therefore sit outside a given zone's trust domain, even though both controls are of the same type; only the attachment point and the path condition differ. This is the technical heart of the nested model: the same perimeter machinery, scoped by path, produces a stronger inner boundary. Typical drivers for standing up a zone are regulated workloads, sensitive data, legal-entity separation, geographic or sovereignty requirements, and phased rollout of stricter controls.

In our EU zone, the EU OU is the branch. The zone's policies attach to it and match on its organization path, so a principal in the general-workloads OU is trusted by the enterprise baseline but not by the zone.

6.5 Shared services and cross-zone dependencies

Narrowing trust to a branch has an immediate operational consequence: shared services that sit outside a zone - centralized identity, logging, security monitoring, shared networking, enterprise data products, integration platforms - are no longer automatically inside its trust boundary, even though they remain inside the enterprise perimeter. Any dependency a zone has on those services must therefore be modelled explicitly as an exception. This is not a weakness of the design but one of its primary benefits: cross-boundary trust becomes visible and governable rather than being silently inherited from broad organizational membership. The mechanics of defining and reviewing those exceptions belong to the operating model in Section 8; here the point is that a zone makes its external dependencies explicit by construction.

6.6 Enterprise perimeter and secure zones compared

The two are not alternatives to choose between; they are the outer and inner layers of one nested model, and they coexist. The following comparison sets out how the same controls differ in scope and trust definition at each layer.

Aspect	Enterprise perimeter (outer)	Secure zone (inner)
Attachment	Organization root / top-level scope	Target OU branch
Coverage	Entire organization	Zone OU and its descendants
Identity trust	Organization-wide (<code>aws:PrincipalOrgID</code>)	Organization path (<code>aws:PrincipalOrgPaths</code>)
Resource trust	Organization-wide (<code>aws:ResourceOrgID</code>)	Organization path (<code>aws:ResourceOrgPaths</code>)
Network trust	Organization-wide	Zone-specific
Other zones	Inside the baseline trust domain	Outside this zone's trust domain
Exceptions	Relatively few	Potentially more
Purpose	Enterprise baseline	Stronger isolation for a specific requirement
Administration	Central	Central or delegated to the zone owner

Table 6.1 - The enterprise perimeter and a secure zone compared as outer and inner layers of one nested model.

An account inside a zone is subject to both layers at once: the enterprise controls keep it inside the organization's trust domain, while the zone's controls further restrict it to the narrower path-scoped domain. A request that passes the outer layer but fails the inner one is denied.

6.7 Design considerations

A zone's boundary should reflect an actual trust relationship, not simply mirror the organizational chart. Before creating one, the organization should be able to name the identities, resources, and network paths that belong inside it, the shared-service dependencies that will cross its boundary, and the requirement that justifies it in the first place. The guiding rule is to make each zone the smallest practical trust domain that still supports the legitimate business and technical relationships the workloads depend on - small enough to be meaningful isolation, but not so tight that it forces a proliferation of exceptions that erode the boundary it was meant to create.

Key takeaway: Keep one small, stable enterprise perimeter at the organization root, and nest secure zones inside it only where a real residency, regulatory, or sensitivity requirement justifies stronger isolation.

7. Implementing the data perimeter: A worked example

The previous sections established the nested model - an enterprise perimeter with one or more secure zones inside it - and the mechanisms that create it. This section works a single organization through that model end to end, using one secure zone that protects EU workloads subject to stronger isolation and data-residency requirements. The policies shown are real, reduced to the perimeter essentials, and each demonstrates one of the control objectives defined in Table 5.1. Placeholders written as `{...}` are substituted with organization-specific values at deployment.

7.1 Scenario and organization layout

The example organization runs centralized functions alongside general workloads and a single regulated branch. Security, shared services, and general workloads sit in their own OUs, and a dedicated EU OU contains the workloads that require the secure zone (Figure 7.1).

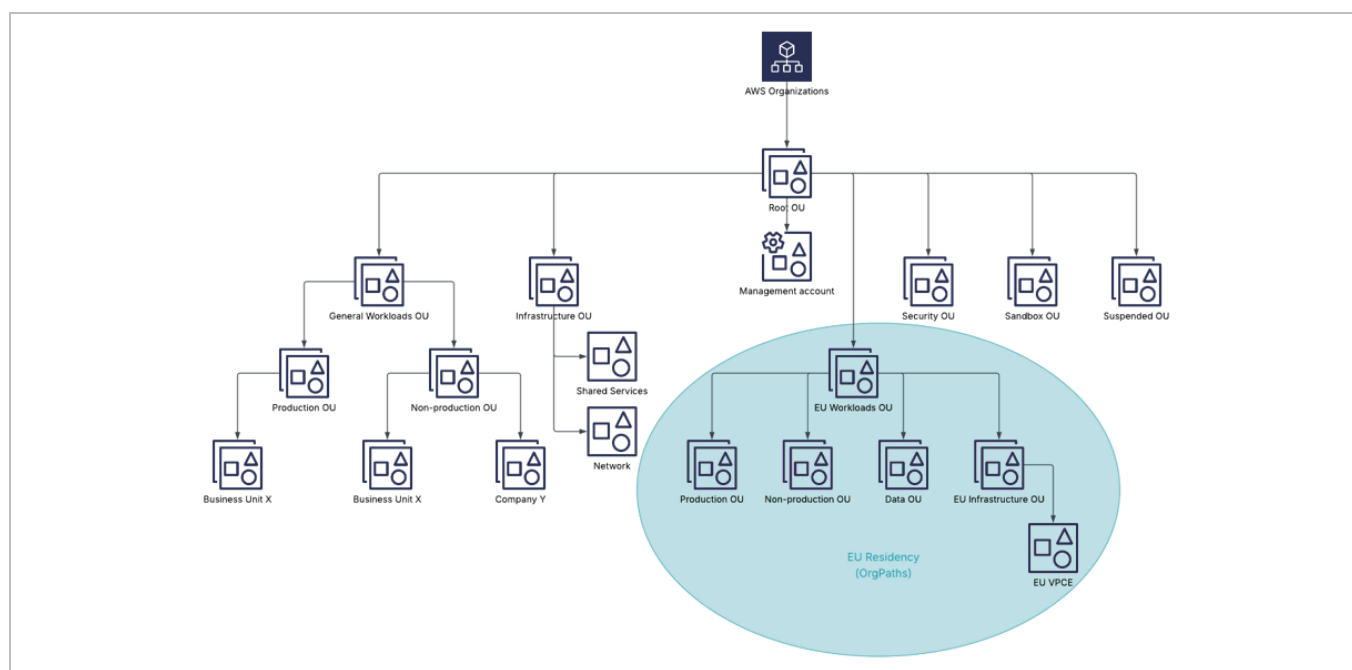


Figure 7.1 - Example organization layout. Centralized identity, logging, security monitoring, and networking remain outside the EU OU.

Because those centralized services sit outside the EU OU, they fall outside the zone's trust boundary while remaining inside the enterprise perimeter. This is the design challenge the rest of the section resolves: the EU workloads need stronger isolation and, at the same time, legitimate access to services that live outside their zone.

7.2 Control allocation

Following the nested model, the enterprise baseline and the EU zone do different jobs, and the zone strengthens rather than replaces the baseline. Two featured controls are enterprise-wide, both attached at the organization root so that they protect every account and every zone: the confused-deputy safeguard and the region restriction. The remaining controls are scoped to the EU branch. Table 7.1 maps each policy to its layer and attachment point, the perimeter dimension it enforces, and the condition key that does the work; the sections that follow take them layer by layer.

Listing	Policy (type)	Layer / attachment	Dimension	Key doing the work
7.1	rcp_PreventConfusedDeputyProblem (RCP)	Baseline / org root	Identity (AWS-service safeguard)	aws:SourceOrgID vs aws:ResourceOrgID
7.2	rcp_IdentityDataPerimeter (RCP)	Zone / EU OU	Identity	aws:PrincipalOrgPaths
7.3	scp_ResourceDataPerimeter (SCP)	Zone / EU OU	Resource	aws:ResourceOrgPaths
7.4	scp_ResourceDataPerimeterNetwork (SCP)	Zone / EU OU	Resource (action-scoped)	aws:ResourceAccount + NotAction
7.5	scp_ResourceDataPerimeterImages (SCP)	Zone / EU OU	Resource (action-scoped)	aws:ResourceAccount + NotAction
7.6	rcp_NetworkDataPerimeter (RCP)	Zone / EU OU	Network	aws:VpceAccount
7.7	scp_RestrictRegions (SCP)	Baseline / org root	Residency (operating footprint)	aws:RequestedRegion
7.8	scp_RestrictRegionsEU (SCP)	Zone / EU OU	Residency (EU only)	aws:RequestedRegion

Table 7.1 - The featured policies mapped to layer and attachment, perimeter dimension, and condition key. Listings 7.1 and 7.7 attach at the organization root and cover every account in the organization; the remaining policies are specific to the EU zone. The residency rows are grouped separately because region restriction reinforces the perimeter rather than forming a fourth dimension of it.

7.3 Layer 1 - The enterprise baseline

The enterprise perimeter is the outer boundary that every account inherits. Its organization-wide identity and resource controls follow the same shape as the zone policies below but use the organization-wide keys `aws:PrincipalOrgID` and `aws:ResourceOrgID` in place of the path keys, establishing that only organizational principals reach organizational resources [7]. Those two policies are not reproduced here: they differ from Listings 7.2 and 7.3 only in the condition key, and AWS publishes reference versions alongside its data-perimeter guidance [7]. The baseline also guards against AWS services being used as a *confused deputy* - tricked into using their privileges on behalf of a caller in another organization. Listing 7.1 denies any AWS service principal carrying a

request from a source in a *different* organization from reaching the organization's KMS keys or secrets, by requiring the source and resource organizations to match.

```
{
  "VERSION": "2012-10-17",
  "STATEMENT": [
    {
      "SID": "PreventConfusedDeputyProblem",
      "EFFECT": "Deny",
      "ACTION": ["kms:*", "secretsmanager:*"],
      "PRINCIPAL": "*",
      "RESOURCE": "*",
      "CONDITION": {
        "STRINGNOTEQUALSIFEXISTS": {
          "AWS:SOURCEORGID": "${aws:ResourceOrgID}"
        },
        "NULL": { "AWS:SOURCEACCOUNT": "false" },
        "BOOL": { "AWS:PRINCIPALISAWSSERVICE": "true" }
      }
    }
  ]
}
```

Listing 7.1 - rcp_PreventConfusedDeputyProblem.json, attached to the organization root. `${aws:ResourceOrgID}` is an AWS policy variable resolved at evaluation time to the organization that owns the resource, so the deny fires whenever an AWS service carries a request from an out-of-organization source. Because it attaches at the root, it protects every account and every zone, not only the EU zone. It safeguards the identity dimension for the service-to-service case that the human and role controls in Section 7.4 do not cover.

The baseline is deliberately zone-agnostic: it should stay stable and contain no assumptions specific to the EU zone, so that it continues to hold as the estate grows.

7.4 Layer 2 - The EU secure zone

The zone narrows trust from the whole organization to the EU branch, using the path-scoped keys. Two policies establish the core boundary. The identity perimeter (Listing 7.2, an RCP attached to the EU OU) protects the zone's S3, KMS, and Secrets Manager resources by denying any principal whose organization path is *not* within the EU branch - its first statement covering human and role principals via `aws:PrincipalOrgPaths`, its second covering AWS services acting on behalf of a source via `aws:SourceOrgPaths` [8]. The resource perimeter (Listing 7.3, an SCP on the same OU) works in the opposite direction, denying EU principals from reaching resources *outside* the EU path. Together they realize the objectives *only trusted identities can access organizational resources* and *organizational identities can access only trusted resources*.

```
{
  "VERSION": "2012-10-17",
  "STATEMENT": [
    {
      "SID": "IdentityDataPerimeter1",
      "EFFECT": "Deny",
      "ACTION": ["s3:*", "kms:*", "secretsmanager:*"],
      "PRINCIPAL": "*",
      "RESOURCE": "*",
      "CONDITION": {
        "FORANYVALUE:STRINGNOTLIKE": {
          "AWS:PRINCIPALORGPATHS": ["${ou_org_path}"]
        },

```

```

    "BOOLIFEXISTS": { "AWS:PRINCIPALISAWSSERVICE": "false" },
    "ARNNOTLIKE": {
      "AWS:PRINCIPALARN": [
        "arn:aws:iam::*:role/AWSControlTowerExecution",
        "arn:aws:iam::*:role/SomeCentralisedDeploymentRole",
        "arn:aws:iam::*:role/SomeSupportRoles-*"
      ]
    }
  },
  {
    "SID": "IdentityDataPerimeter2",
    "EFFECT": "Deny",
    "ACTION": ["s3:*", "kms:*", "secretsmanager:*"],
    "PRINCIPAL": "*",
    "RESOURCE": "*",
    "CONDITION": {
      "FORANYVALUE:STRINGNOTLIKEIFEXISTS": {
        "AWS:SOURCEORGPATHS": ["${ou_org_path}"]
      },
      "BOOLIFEXISTS": { "AWS:PRINCIPALISAWSSERVICE": "true" },
      "NULL": { "AWS:SOURCEACCOUNT": "false" }
    }
  }
]
}

```

Listing 7.2 - `rcp_IdentityDataPerimeter.json`, attached to the EU OU. Statement 1 handles human and role principals; statement 2 handles AWS services acting on behalf of a source. The `aws:PrincipalArn` exceptions admit named central automation and support roles that legitimately operate across the boundary (see Section 7.8).

```

{
  "VERSION": "2012-10-17",
  "STATEMENT": [
    {
      "SID": "ResourceDataPerimeter",
      "EFFECT": "Deny",
      "NOTACTION": ["organizations:*"],
      "RESOURCE": "*",
      "CONDITION": {
        "FORANYVALUE:STRINGNOTLIKE": {
          "AWS:RESOURCEORGPATHS": ["${ou_org_path}"]
        },
        "BOOL": { "AWS:PRINCIPALISAWSSERVICE": "false" },
        "STRINGNOTEQUALS": {
          "AWS:RESOURCEACCOUNT": [
            "${network_account_id}",
            "${image_factory_account_id}"
          ]
        }
      }
    }
  ]
}

```

Listing 7.3 - `scp_ResourceDataPerimeter.json`, attached to the EU OU. EU principals are denied access to resources whose organization path is outside the EU branch, except resources in the two explicitly listed shared-service accounts. An identity trusted by the enterprise baseline can therefore still sit outside this zone's trust domain - the same control type, scoped by path, produces the stronger inner boundary.

7.5 Narrowing the shared-account exceptions

An account-level exception is never narrow. Admitting one account so that a single integration works admits everything in that account.

The `aws:ResourceAccount` exceptions in Listing 7.3 have exactly this side effect, and it must be corrected. Excluding an account from the resource perimeter re-admits that account *in full* - every service and every action in it - when the zone's genuine need is far narrower. For the shared networking account, EU workloads legitimately require only a specific set of Transit Gateway attachment and route-table operations; for the image-factory account, only the ability to launch instances from shared golden Amazon Machine Images (AMIs). A bare account-level exception grants far more than intended.

The fix is a complementary SCP for each exception. Where Listing 7.3 decides *which accounts* an EU principal may reach, Listings 7.4 and 7.5 constrain *what may be done* once each account is reached, denying every action except the mandatory ones.

```
{
  "VERSION": "2012-10-17",
  "STATEMENT": [
    {
      "SID": "ResourceDataPerimeterNetwork",
      "EFFECT": "Deny",
      "NOTACTION": [
        "ec2:AcceptTransitGatewayVpcAttachment",
        "ec2:AssociateTransitGatewayRouteTable",
        "ec2:CreateTags",
        "ec2:CreateTransitGatewayVpcAttachment",
        "ec2>DeleteTags",
        "ec2>DeleteTransitGatewayVpcAttachment",
        "ec2:DescribeTags",
        "ec2:DescribeTransitGateways",
        "ec2:DescribeTransitGatewayVpcAttachments",
        "ec2:DisableTransitGatewayRouteTablePropagation",
        "ec2:DisassociateTransitGatewayRouteTable",
        "ec2:EnableTransitGatewayRouteTablePropagation",
        "ec2:GetTransitGatewayRouteTableAssociations",
        "ec2:GetTransitGatewayRouteTablePropagations"
      ],
      "RESOURCE": "*",
      "CONDITION": {
        "STRINGEQUALS": {
          "AWS:RESOURCEACCOUNT": ["${network_account_id}"]
        }
      }
    }
  ]
}
```

Listing 7.4 - `scp_ResourceDataPerimeterNetwork.json`, attached to the EU OU. Denies EU principals every action against the networking account except the Transit Gateway attachment and route-table operations they need to connect to shared networking.

```
{
  "VERSION": "2012-10-17",
  "STATEMENT": [
    {
      "SID": "ResourceDataPerimeterImages",
      "EFFECT": "Deny",
```

```
"NotAction": ["ec2:RunInstances"],
"Resource": "*",
"Condition": {
  "StringEquals": {
    "aws:ResourceAccount": ["${image_factory_account_id}"]
  }
}
}
```

Listing 7.5 - scp_ResourceDataPerimeterImages.json, attached to the EU OU. Denies EU principals every action against the image-factory account except ec2:RunInstances, allowing EU workloads to launch instances from the shared golden AMIs while blocking any other cross-account operation.

Because these policies constrain EU principals reaching outward, they are SCPs on the EU OU - the same scope as Listing 7.3. The shared accounts sitting outside the EU OU does not matter here: the control governs the EU side of the interaction, not the shared account's own principals. The pattern generalizes to any account admitted by an `aws:ResourceAccount` exception - the exception opens the account, and a paired action-scoped SCP on the same OU narrows it to the mandatory operations. Because these are shared-services accounts that every zone may consume, the same pair is replicated in each secure zone's SCP set, so every zone admits the shared accounts on identical, minimal terms.

7.6 Network segregation and the dedicated endpoint account

Identity and resource controls decide who may reach what; the network dimension decides by which path. The secure zone routes its access to sensitive data services through a dedicated VPC endpoint account - an account whose sole purpose is to host the VPC endpoints through which the zone reaches Amazon S3 and AWS KMS (Figure 7.2). Concentrating the zone's endpoints in one governed account gives the network dimension a single, auditable choke point: rather than approving endpoints scattered across many accounts, the zone declares one account as its expected network path and pins access to it.

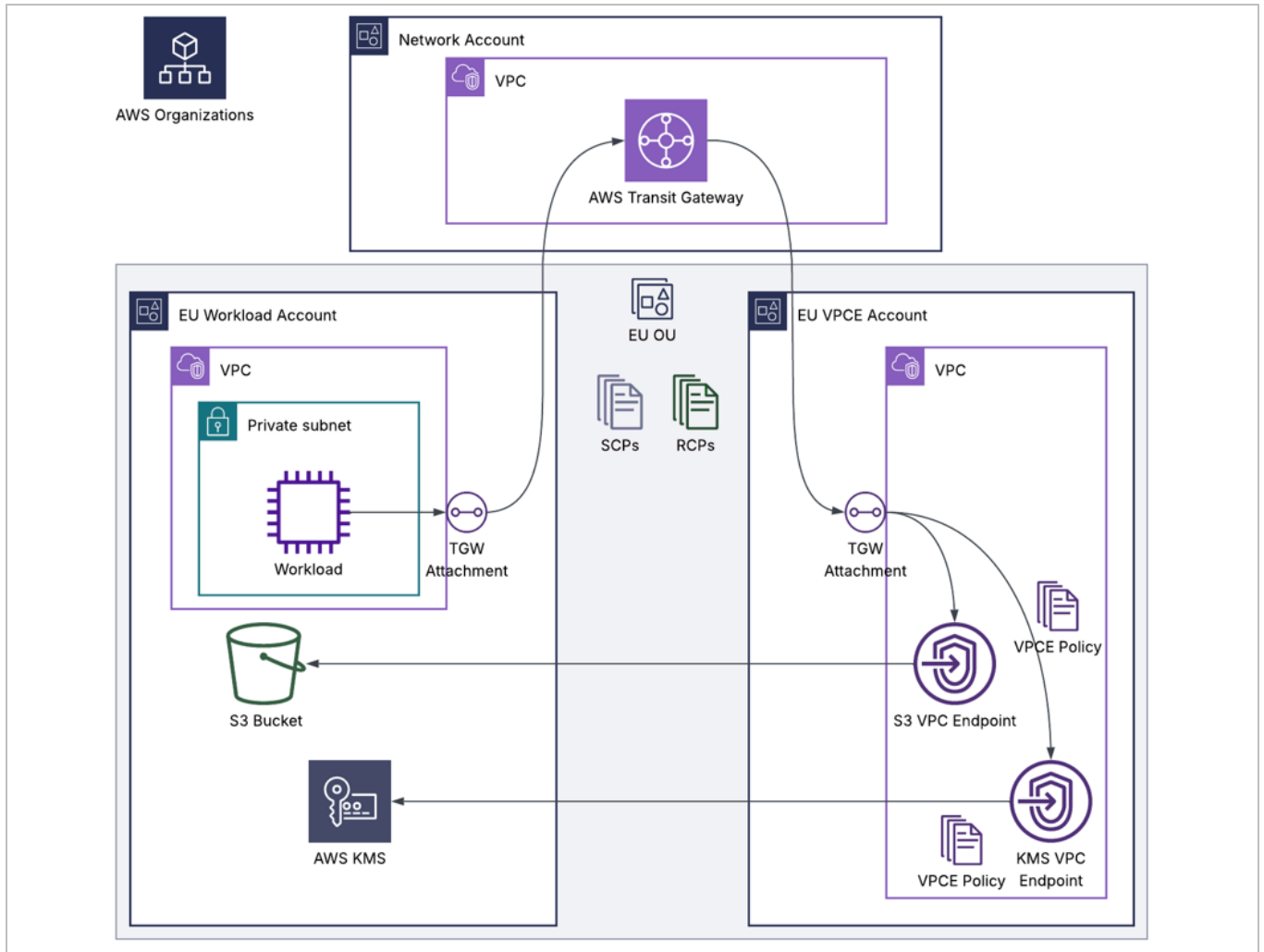


Figure 7.2 - The secure zone's approved network path. All access to the zone's sensitive services traverses VPC endpoints hosted in the dedicated endpoint account.

The RCP in Listing 7.6 enforces this design. It denies access to the zone's S3 and KMS resources whenever a request arrives through a VPC endpoint whose account is *not* the dedicated endpoint account, while leaving genuine AWS service paths untouched. This realizes the Table 5.1 objective *organizational resources are accessed only from expected networks*, where the expected network is defined as the dedicated account's endpoints.

```
{
  "VERSION": "2012-10-17",
  "STATEMENT": [
    {
      "SID": "NetworkDataPerimeter",
      "EFFECT": "Deny",
      "ACTION": ["s3:*", "kms:*"],
      "PRINCIPAL": "*",
      "RESOURCE": "*",
      "CONDITION": {
        "STRINGNOTEQUALSIFEXISTS": {
          "AWS:VPCEACCOUNT": "${vpce_account_id}"
        },
        "BOOLIFEXISTS": {
          "AWS:VIAAWSSERVICE": "false",
          "AWS:PRINCIPALISAWSSERVICE": "false"
        },
        "NULL": { "AWS:VPCEACCOUNT": "false" }
      }
    }
  ]
}
```

Listing 7.6 - `rcp_NetworkDataPerimeter.json`, attached to the EU OU. The `NULL` check restricts the statement to requests that actually traverse an endpoint; `aws:VpceAccount` must then match the dedicated endpoint account (`${vpce_account_id}`). The control blocks access through any other account's endpoints; combined with the routing design, it makes the dedicated account's endpoints the zone's single approved path to its data. Network isolation is one dimension of the perimeter, not an authorization mechanism in itself - identity and resource policies still decide whether the operation is allowed.

7.7 Regional restriction and data residency

The controls shown so far establish *who* is trusted, *what* they may reach, and *through which network*. Residency adds the remaining question - *in which regions operations may occur* - which is often the specific regulatory obligation a secure zone exists to satisfy. Region restriction is best understood as a control that reinforces the perimeter rather than a fourth perimeter dimension: AWS's data-perimeter model treats identity, resource, and network as the perimeter proper, and region control as a complementary residency guardrail. It demonstrates the nested model especially clearly, because the relationship between the two layers is a strict subset.

At the enterprise baseline, an SCP fixes the organization's operational footprint: operations are denied in any region outside the approved set, so the estate never spreads across all of AWS (Listing 7.7). Only genuinely global services - those with a single endpoint physically located in `us-east-1`, such as IAM, CloudFront, Route 53, and Support - are exempted, because they cannot be region-bound without breaking [9].

```
{
  "VERSION": "2012-10-17",
  "STATEMENT": [
    {
      "SID": "RestrictRegions",
      "EFFECT": "Deny",
      "NOTACTION": [
        "cloudfront:*",
        "globalaccelerator:*",
        "iam:*",
        "kms:*",

```

```

    "organizations:*",
    "route53:*",
    "sts:*",
    "support:*",
    "waf:*",
    "wafv2:*",
    "networkmanager:*",
    "account:*",
    "aws-portal:*"
  ],
  "RESOURCE": "*",
  "CONDITION": {
    "STRINGNOTEQUALS": {
      "AWS:REQUESTEDREGION": ["${aws_regions}"]
    },
    "BOOL": { "AWS:PRINCIPALISAWSSERVICE": "false" },
    "ARNNOTLIKE": {
      "AWS:PRINCIPALARN": [
        "arn:aws:iam::*:role/AWSControlTowerExecution",
        "arn:aws:iam::*:role/SomeCentralisedDeploymentRole",
        "arn:aws:iam::*:role/SomeSupportRoles-*"
      ]
    }
  }
}
]
}
}

```

Listing 7.7 - `scp_RestrictRegions.json`, attached at the enterprise baseline. `${aws_regions}` is the organization's approved operating set. The `NotAction` list is a representative set of global services, not an exhaustive one; the authoritative, current list should mirror AWS's published region deny control [10], since AWS updates it as services change.

At the zone, a second SCP narrows the same control to the residency regions - here the EU only - a strict subset of the enterprise set (Listing 7.8). The exemption list stays limited to true global services; every regional and data service (S3, RDS, EFS, DynamoDB, and the rest) is left subject to the restriction, which is what actually binds the zone's data to the EU.

```

{
  "VERSION": "2012-10-17",
  "STATEMENT": [
    {
      "SID": "RestrictRegionsEU",
      "EFFECT": "Deny",
      "NOTACTION": [
        "cloudfront:*",
        "globalaccelerator:*",
        "iam:*",
        "kms:*",
        "organizations:*",
        "route53:*",
        "sts:*",
        "support:*",
        "waf:*",
        "wafv2:*",
        "networkmanager:*",
        "account:*",
        "aws-portal:*"
      ],
      "RESOURCE": "*"
    }
  ]
}

```

```
"CONDITION": {
  "STRINGNOTEQUALS": {
    "AWS:REQUESTEDREGION": [
      "eu-west-1",
      "eu-central-1"
    ]
  },
  "BOOL": { "AWS:PRINCIPALISAWSSERVICE": "false" },
  "ARNNOTLIKE": {
    "AWS:PRINCIPALARN": [
      "arn:aws:iam::*:role/AWSControlTowerExecution",
      "arn:aws:iam::*:role/SomeCentralisedDeploymentRole",
      "arn:aws:iam::*:role/SomeSupportRoles-*"
    ]
  }
}
```

Listing 7.8 - scp_RestrictRegionsEU.json, attached to the EU OU. The allowed set (eu-west-1, eu-central-1, illustrative) is a subset of `aws_regions`. An EU account inherits both SCPs, so its effective allowed regions are the intersection of the two - the EU set - while general workloads elsewhere remain governed only by the wider baseline. This is the same enterprise-baseline-plus-zone nesting as the identity and resource controls, applied to residency; AWS provides it directly as the landing-zone and OU region deny controls under its digital sovereignty control group [10].

KMS is included in the exemption list because the identity and network RCPs (Listings 7.2 and 7.6) already constrain KMS access to the zone's trusted principals and its dedicated endpoints, so region-pinning it as well would be redundant and could interfere with global services that integrate with KMS.

The catch is that exempting KMS leaves keys unpinned. The RCPs bound KMS by identity and network, not by region, so this control does not stop key operations outside the residency regions. Where a regulation mandates strict key-material residency specifically, KMS should instead be left subject to the region SCP.

7.8 Exceptions for shared resources

The EU workloads depend on services that sit outside the zone, and the policies above make those dependencies explicit rather than leaving them to implicit trust. Two kinds of exception appear directly in the policy conditions: `aws:ResourceAccount` exceptions in the resource SCP, which let EU principals reach specific shared-service accounts (then narrowed by Listings 7.4-7.5), and `aws:PrincipalArn` exceptions in the identity RCP, which let named central roles from outside the branch reach the zone’s resources. Table 7.2 records each one and its justification.

Exception (source policy)	What it permits	Why it exists
<code>\${network_account_id}</code> (Listing 7.3, narrowed by 7.4)	EU principals reach the shared networking account, for Transit Gateway operations only	Centralized VPC, Transit Gateway, and shared network resources
<code>\${image_factory_account_id}</code> (Listing 7.3, narrowed by 7.5)	EU principals reach the image-factory account, to launch from shared AMIs only	Golden AMI / image pipeline shared across the organization

Exception (source policy)	What it permits	Why it exists
AWSControlTowerExecution (Listing 7.2)	Control Tower's execution role reaches EU resources	Landing-zone automation and account provisioning
SomeCentralisedDeploymentRole (Listing 7.2)	Central deployment role reaches EU resources	Centralized CI/CD and infrastructure-as-code deployment
SomeSupportRoles-* (Listing 7.2)	Central support roles reach EU resources	Operational support and break-glass access

Table 7.2 - Shared-resource exceptions embedded in the zone policies. Each is narrowly scoped to a specific account or role, so it permits exactly one cross-boundary relationship and does not weaken the zone for anything else.

Expressing exceptions inline keeps them auditable in the policy itself, but each still needs the governance metadata - owner, justification, and review date - defined in the exception lifecycle; the policy records *what* is permitted, and the operating model records *why* and *for how long*.

7.9 Every request must pass both layers: how the two combine

Every request from an account inside the EU OU is evaluated against both layers. The enterprise layer asks whether the request stays inside the organization's trust domain and approved operating regions; the EU layer asks whether it also stays inside the EU path, its approved network estate, and the EU residency regions. Both must allow it - a request that passes the outer layer but fails the inner one is denied. This is the concrete payoff of the nested model: a common enterprise baseline and a stronger zone-specific boundary enforce together, without either compromising for the other. The worked example therefore demonstrates the central principle of the whole approach - strong trust boundaries do not require complete isolation, only explicit and governable trust relationships.

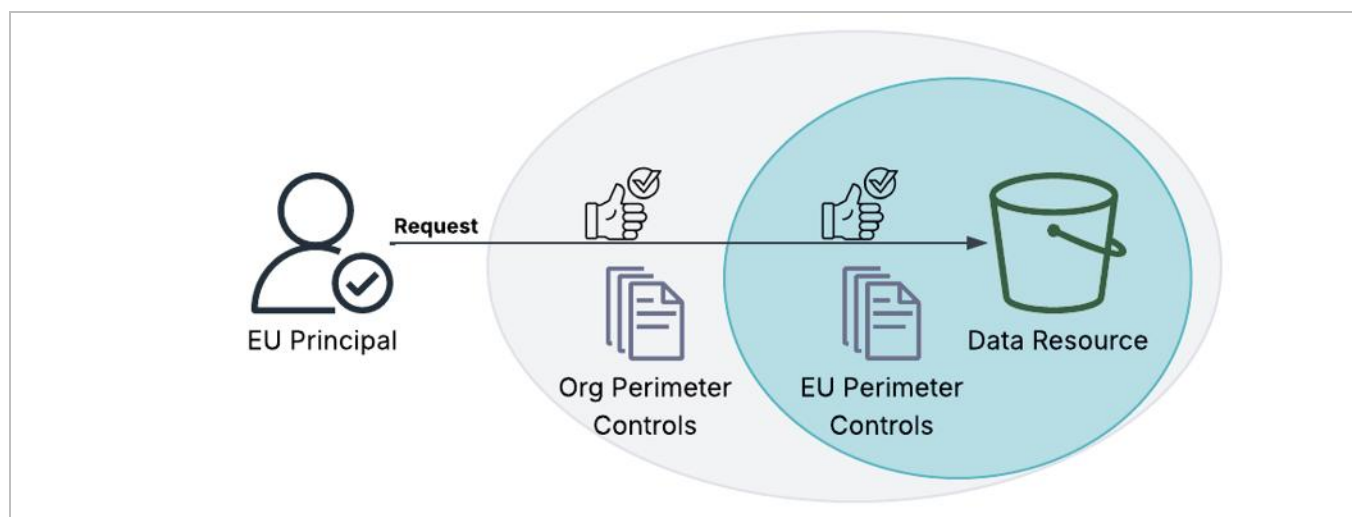


Figure 7.3 - How the two layers combine. Both layers are evaluated for every request; a deny in either is final.

Key takeaway: A request from the EU zone must pass both layers. Admit each shared-account exception at account level, then narrow it to the actions it needs, because an account-level exception on its own admits everything in that account.

8. Operating the data perimeter

A data perimeter is not a one-time security configuration but an operational capability that must stay aligned with the organization, its workloads, its networks, the AWS services in use, and its regulatory obligations as all these change. The controls in the preceding sections establish the boundary; the operating model described here keeps that boundary meaningful over time. Without it, a correctly built perimeter degrades silently as the estate around it moves.

8.1 Ownership and decision rights

Operating the perimeter is a shared responsibility, and the governance roles defined in Section 3 - the Data Governance Board, Data Owners, Stewards, and Custodians - carry directly into it rather than being redefined here.

What the perimeter adds is a set of specific operational responsibilities layered onto those roles. The Cloud Platform or AWS Governance function implements and maintains the organization-level SCPs and RCPs and their attachment points. The Security function defines control requirements and monitors their effectiveness. The Network function owns the approved network paths, VPC endpoints, and the dedicated endpoint accounts on which the network perimeter depends. Workload Owners identify the legitimate application dependencies that become exceptions, and each approved exception has a named Exception Owner accountable for it.

Decision rights across these functions should be made explicit through the same responsibility model (such as RACI) established in Section 3, so that authority over the perimeter is never ambiguous when a change or conflict arises.

8.2 Every exception needs a governed lifecycle

Exceptions are a designed-in feature of a federated perimeter, not a failure of it. To remain safe, each must be treated as a governed object with a defined lifecycle: it is requested, assessed, approved, implemented, monitored, reviewed, and eventually removed. An exception that is implemented but never reviewed is how a narrow, justified deviation slowly becomes a permanent hole.

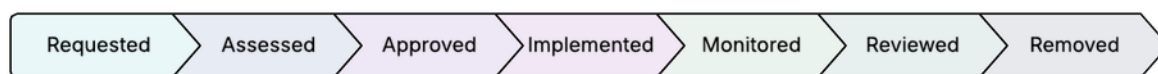


Figure 8.1 - The exception lifecycle. A review either renews the exception, returning it to monitoring, or removes it.

Every exception should carry a consistent record, and this is the authoritative definition that the inline policy exceptions instantiate. The record captures the source identity or workload, the destination resource, the required action, the expected network path, the business justification, the accountable owner, and a review or expiration date. The relationship between the record and the policy is deliberate: the policy condition - an `aws:ResourceAccount` or `aws:PrincipalArn` entry, as shown in Listings 7.3-7.5 - enforces what is permitted, while the exception record documents why, for whom, and for how long. Table 7.2 in the worked example is a populated instance of exactly this schema. Keeping the two linked means, every enforced exception is traceable to a governed decision, and every governed decision is visible in the policy that enforces it.

8.3 Change control

Changes to accounts, OUs, VPCs, endpoints, corporate network ranges, AWS services, or external integrations can all alter the trust model the perimeter assumes, often without any change to the perimeter policies themselves. Perimeter impact should therefore be a standing consideration in normal change management, assessed before a change lands rather than discovered afterward. Infrastructure-as-code and centralized policy deployment provide the controlled mechanisms for making and reviewing these changes, so that the trust model evolves deliberately and every adjustment leaves a reviewable trail.

8.4 Conflict resolution

When a legitimate business requirement conflicts with a perimeter control, the resolution should follow a fixed order of preference, from least to most disruptive to the boundary. First, determine whether the requirement can be met inside the existing boundary as it stands. If not, consider an architectural change - relocating a resource, adjusting a network path - that removes the conflict without touching the perimeter. Only then consider a narrowly scoped exception, and only as a last resort consider changing the broader perimeter itself. This ordering protects the boundary from being weakened wholesale to accommodate a single workload, which is the most common way a perimeter quietly loses its value.

8.5 Monitoring and assurance

Operating the perimeter requires continuous visibility into both the controls and the environment around them. Monitoring should cover policy and configuration changes, denied-access events, unusual access patterns, changes to the set of trusted accounts and resources, changes to approved network paths, the appearance of new external dependencies, and activity against existing exceptions. AWS provides complementary sources for this: AWS CloudTrail for activity, AWS Config for configuration state, IAM Access Analyzer for external and cross-account access findings, and Amazon GuardDuty and AWS Security Hub for threat detection and aggregated findings. These signals should feed the organization's broader security operations and incident-management processes rather than being monitored in isolation, so that a perimeter deviation is handled with the same rigor as any other security event.

8.6 Drift erodes the perimeter without any policy changing

Drift occurs when the actual environment no longer matches the assumptions encoded in the perimeter - the controls are unchanged, but the ground beneath them has moved. Common forms include new accounts created outside the intended OU structure, new VPC endpoints not reflected in the trust model, unapproved cross-boundary dependencies, manually modified policies, and changed corporate network ranges. Because drift accumulates quietly, it should be detected continuously where practical and corrected through controlled remediation rather than ad-hoc fixes. Left unmanaged, drift is what turns a perimeter that passed its last audit into one that no longer reflects reality.

8.7 Evidence for audit

The organization should be able to demonstrate not only that the perimeter exists but that it operates effectively over time. The evidence base for this includes the SCP and RCP configurations, their attachment and inheritance, the trust-boundary definitions, VPC and endpoint inventories, the register of approved exceptions, CloudTrail records, AWS Config findings, IAM Access Analyzer results, remediation records, and the outcomes of periodic

control reviews. The objective is continuous assurance rather than audit-time documentation: evidence should show that the controls have operated throughout the period, not merely that a compliant configuration existed on the day of the audit.

8.8 Automation and Policy as Code

At scale, the perimeter cannot be maintained by hand. The same infrastructure-as-code and centralized deployment mechanisms introduced for change control (Section 8.3) should carry the routine work of perimeter management: deploying the baseline to new accounts and OUs as they are onboarded, validating policy changes before they land, inventorying VPC endpoints, detecting drift (Section 8.6), monitoring exception activity, generating audit evidence (Section 8.7), and surfacing stale exceptions for review. Automating these tasks does more than save effort - it keeps the enforced configuration continuously aligned with the intended trust model, so that human governance is freed to focus on the decisions that actually require judgment: risk acceptance, exceptions, and changes to the trust boundaries themselves.

Table 8.1 turns this section into a standing checklist. Cadences are indicative and should be set to the organization's risk appetite and rate of change.

Activity	Owner	Cadence
Review every exception against its record; remove expired ones	Exception Owner, with Security	Quarterly
Assess perimeter impact of account, OU, network, and service changes	Cloud Platform	Every change
Triage denied-access events and unusual access patterns	Security	Continuous
Check for drift: new accounts, endpoints, network ranges, manual policy edits	Cloud Platform and Network	Continuous, with monthly review
Confirm new accounts and OUs inherit the baseline	Cloud Platform	At onboarding
Reconcile VPC endpoint inventory with the trust model	Network	Monthly
Collect and retain evidence of control operation	Security, with Audit	Continuous; reviewed per audit period
Review the trust model and zone boundaries	Data Governance Board	Annually, or on major change

Table 8.1 - Perimeter operating checklist.

Key takeaway: A perimeter that is not operated degrades silently. Every exception needs an owner, a justification, and a review date, and automation should keep what is enforced aligned with what was intended.

9. Extending beyond the Organization: External and hybrid trust

The nested model of Sections 6 and 7 subdivides trust *inside* the organization. Every condition it relies on - `aws:PrincipalOrgID`, `aws:PrincipalOrgPaths`, `aws:ResourceOrgPaths` - describes principals and resources that already belong to the organization; a secure zone is a smaller trust domain carved out of a larger one. Two requirements push past that internal boundary and cannot be met by adding another zone: granting access to genuinely external parties and reaching workloads that run outside AWS. Both are extensions of the same pattern - define trust explicitly, then enforce it - applied at the outer edge of the organization rather than within it.

9.1 External partners

An external partner sits outside `aws:ResourceOrgID` entirely, so admitting one is a different act from creating a nested zone: it cannot be expressed by narrowing an organization-scoped condition, because the partner is not in the organization at all. Instead, it is admitted by explicit, bounded trust [7], and the direction of access determines the mechanism.

Where the zone's own principals need to consume a partner's resources, the mechanism is exactly the shared-account pattern from Section 7. The partner's account is added as an `aws:ResourceAccount` exception in the resource perimeter and then narrowed by a complementary action-scoped policy, precisely as the shared networking and image-factory accounts were in Listings 7.3-7.5.

What differs is the organization boundary, and it has a concrete consequence. Those shared accounts live inside the organization, so the enterprise baseline already trusts them and only the zone's path restriction has to be relieved; a partner account lies outside `aws:ResourceOrgID`, so it fails the enterprise resource perimeter as well and must be admitted at *both* layers, not just the zone.

The trust is also qualitatively different - a shared-services account is under the organization's own governance, whereas a partner account is controlled by a third party whose security posture the organization does not manage - which is why a partner exception should carry a strict review or expiration date rather than standing indefinitely.

Where the direction is reversed - a partner assuming a role in the organization's accounts to reach its resources - the safeguard is the **external ID**. The role's trust policy should require a partner-supplied external ID through the `sts:ExternalId` condition, so the role can be assumed only when that exact value is presented [11]. This matters because a partner - typically a SaaS provider serving many customers - often assumes a similarly named role in each customer's account; without a per-customer external ID, one customer could induce the partner to act against another customer's role, the third-party form of the confused-deputy problem that Section 7 addressed for AWS-service principals internally. This inbound direction has no equivalent among the shared-service accounts, which the zone consumes rather than the reverse.

In both directions the partner is defined with the same rigor as an internal exception - the external identity or organization, the specific resources and actions permitted, the expected network path, the accountable business owner, and a review date - and in both it remains visibly outside the organization's own trust domain.

9.2 Hybrid and on-premises access

The network perimeter already contemplates paths that originate outside AWS: expected networks include the organization's on-premises data centers and corporate address ranges, not only its VPCs and endpoints [7].

Extending the perimeter to a hybrid estate is therefore partly a matter of describing those paths precisely - using the source-network conditions introduced in Section 5.4 to admit corporate ranges and private connectivity while continuing to deny unexpected origins.

Identity is the harder half, and the mechanism for it is **IAM Roles Anywhere**: workloads running outside AWS present X.509 certificates issued by a certificate authority that the organization registers as a trust anchor, and exchange them for temporary AWS credentials, using the same IAM roles and policies as in-cloud workloads and removing the need for long-lived access keys [12].

An on-premises workload authenticated this way is a first-class trusted identity, subject to the same perimeter conditions as one running inside AWS, so the governance questions do not change across the boundary - which identities are trusted, which resources they may reach, through which paths, and under which exceptions - even though the credential mechanism differs.

The pattern is therefore reusable at the organization's outer edge as well as within it: the trust model is the durable part, and the specific AWS mechanisms - the `aws:ResourceAccount` exception for consuming a partner's resources, the external ID for partners reaching in, and IAM Roles Anywhere for hybrid workloads - are the implementation detail that varies by case.

Key takeaway: The same trust model extends past the organization's edge. Only the mechanism changes: resource-account exceptions for consuming partner resources, external IDs for partners reaching in, and IAM Roles Anywhere for hybrid workloads.

10. Conclusion: From blueprint to operation

This whitepaper set out to show how a distributed AWS data estate can be governed without being centralized. Its central proposition is that data can be governed where it resides, provided the organization establishes clear ownership, applies controls to the services that actually hold the data, and draws an enforceable trust boundary around them. The data perimeter is what makes that boundary real: a preventive, coarse-grained control over which identities, resources, and networks are trusted at all, sitting above the fine-grained, least-privilege permissions that decide what a trusted principal may then do.

Each layer does a different job, and none is sufficient on its own. The governance model defines what must be true. The AWS architecture governs data where it resides. The enterprise perimeter sets the outer trust boundary, and secure zones narrow it where residency, regulation, or sensitivity require. The operating model keeps all of it accurate as the estate changes. Three rules carry into any implementation: make every cross-boundary trust relationship explicit, keep the enterprise baseline small and stable, and treat every exception as a governed object with an owner and a review date.

The model does not need to be adopted all at once. A practical sequence establishes the foundation first - the organizational structure, governance roles, classification, and baseline security controls - then rolls out the enterprise perimeter progressively, moving from monitoring to enforcement once trusted identities, resources, and networks are understood. Stronger secure zones follow for the workloads that require them, each with its identity, resource, network, and residency controls and its narrowly scoped exceptions. Only then does the operating model become the ongoing work: exception management, monitoring, drift detection, evidence, and the automation that keeps enforcement aligned with intent. Finally, the same pattern extends to further zones, external partners, and hybrid environments as the estate and its obligations grow. Each phase delivers standalone value, so governance maturity can develop incrementally rather than through a single large transformation.

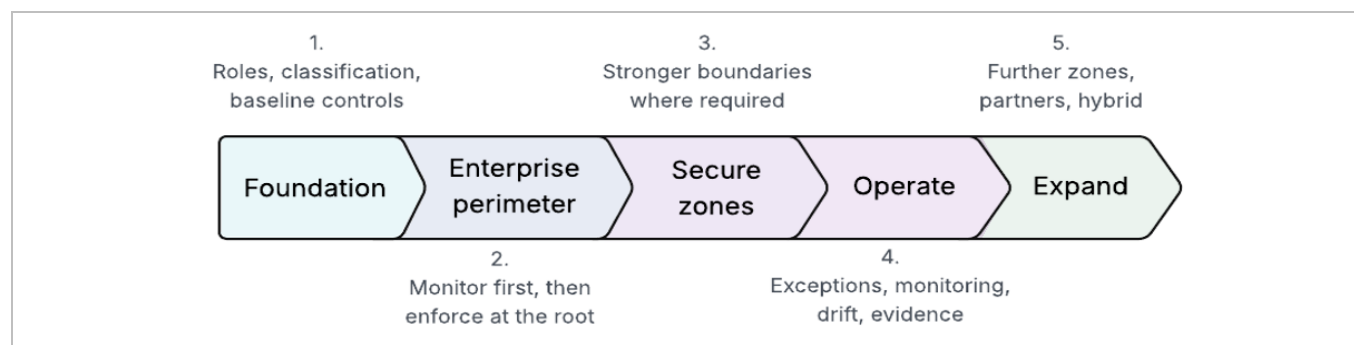


Figure 10.1 - Adoption roadmap. Each phase delivers value on its own; the first 90 days below cover phases 1 to 3 and begin phase 4.

The result is not another centralized security control or a mandate to consolidate a distributed estate, but a repeatable governance pattern in which organizational policy, AWS architecture, preventive controls, and operational assurance reinforce one another. Data remains where it delivers value; ownership and accountability stay explicit; an enterprise baseline guarantees a common floor; stronger boundaries are introduced where risk requires them; access remains least-privilege; exceptions remain governed; and the controls are continuously monitored and evidenced. The data perimeter is one component of that pattern, and its worth lies in how it combines with the rest - organizational structure defining scope, identity and resource controls defining authorization, network controls defining expected paths, and monitoring providing continuous assurance. Together they let the model move beyond a blueprint and become an operational capability that evolves with the organization, the AWS platform, and the data it exists to protect.

A realistic first 90 days looks like this:

- **Days 1 to 30:** confirm the governance roles and name an exception owner; inventory accounts, OUs, VPC endpoints, and cross-account dependencies; and deploy the enterprise baseline in monitoring mode.
- **Days 31 to 60:** turn the dependencies observed in monitoring into an exception register, then enforce the enterprise baseline at the organization root.
- **Days 61 to 90:** build the first secure zone for the workload with the clearest residency or regulatory driver, feed perimeter signals into security operations, and run the first exception review.

Five questions test whether an organization needs this work now:

1. Could a valid credential copy data to an account outside the organization today, and would anything stop it?
2. Which shared-services accounts are exempted from perimeter controls, and is each exemption scoped to actions or to the whole account?
3. Who owns each exception, and when was it last reviewed?
4. Are encryption keys held to the same residency controls as the data they protect?
5. Could the organization show an auditor that the perimeter operated throughout the last year, not only that it was configured today?

About Phi Partners. Phi Partners is a specialist capital markets consultancy focused on front office, pricing, risk, quantitative services, vendor platforms, and cloud engineering. Its Cloud Services practice designs and operates secure multi-account AWS architectures for financial institutions, working within client governance, information security, and audit processes.

References

- [1] DAMA International, *DAMA-DMBOK: Data Management Body of Knowledge* - <https://dama.org/learning-resources/dama-data-management-body-of-knowledge-dmbok>
- [2] EDM Council, *Data Management Capability Assessment Model (DCAM)* - <https://edmcouncil.org/frameworks/dcam/>
- [3] ISO/IEC 38505-1:2017, *Information technology - Governance of IT - Governance of data - Part 1: Application of ISO/IEC 38500 to the governance of data*. Geneva: ISO, 2017 (withdrawn 2026-08-20; superseded by ISO/IEC 38505-1:2026) - <https://www.iso.org/standard/56639.html> Current edition: <https://www.iso.org/standard/87195.html>
- [4] ISACA, *COBIT 2019 Framework: Governance and Management Objectives*. Schaumburg, IL: ISACA, 2019. - <https://www.isaca.org/resources/cobit>
- [5] Regulation (EU) 2016/679 (General Data Protection Regulation), *Official Journal of the European Union*, L 119, 2016. - <https://eur-lex.europa.eu/eli/reg/2016/679/oj>
- [6] Amazon Web Services, *Building a Data Perimeter on AWS*, AWS Whitepaper, Jun. 13, 2023. - <https://docs.aws.amazon.com/whitepapers/latest/building-a-data-perimeter-on-aws/building-a-data-perimeter-on-aws.html>
- [7] I. Epshteyn, "Establishing a data perimeter on AWS: Overview," *AWS Security Blog*, updated Aug. 28, 2025. - <https://aws.amazon.com/blogs/security/establishing-a-data-perimeter-on-aws/>
- [8] Amazon Web Services, "How to control access to AWS resources based on AWS account, OU, or organization," *AWS Security Blog*, Apr. 27, 2022 (updated 2025). - <https://aws.amazon.com/blogs/security/how-to-control-access-to-aws-resources-based-on-aws-account-ou-or-organization/>
- [9] Amazon Web Services, "Deny access to AWS based on the requested Region," *AWS IAM User Guide*. - https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_examples_aws_deny-requested-region.html
- [10] Amazon Web Services, "Region deny control applied to the OU" (control CT.MULTISERVICE.PV.1, Digital sovereignty group), *AWS Control Tower Controls Reference*. - <https://docs.aws.amazon.com/controltower/latest/controlreference/ou-region-deny.html>
- [11] Amazon Web Services, "The confused deputy problem," *AWS IAM User Guide*. - <https://docs.aws.amazon.com/IAM/latest/UserGuide/confused-deputy.html>
- [12] Amazon Web Services, "AWS IAM Roles Anywhere: User Guide" (extending IAM roles to workloads running outside AWS). - <https://docs.aws.amazon.com/rolesanywhere/latest/userguide/introduction.html>

Appendix A - Policy reference

This appendix collects the policies referenced in the whitepaper. Section A.1 indexes the data-perimeter policies shown in full in Section 7. This appendix provides the complementary “govern data in place” guardrails introduced in Section 4, which support the perimeter but are not themselves perimeter controls. All policies use `${...}` placeholders for organization-specific values.

A.1 Data-perimeter policies

(See Section 7)

Policy	Type	Section 7 listing	Attachment
rcp_PreventConfusedDeputyProblem	RCP	Listing 7.1	Org root
rcp_IdentityDataPerimeter	RCP	Listing 7.2	EU OU
scp_ResourceDataPerimeter	SCP	Listing 7.3	EU OU
scp_ResourceDataPerimeterNetwork	SCP	Listing 7.4	EU OU
scp_ResourceDataPerimeterImages	SCP	Listing 7.5	EU OU
rcp_NetworkDataPerimeter	RCP	Listing 7.6	EU OU
scp_RestrictRegions	SCP	Listing 7.7	Org root
scp_RestrictRegionsEU	SCP	Listing 7.8	EU OU

A.2 Encryption controls

scp_DataEncryptionAtRestS3.json - denies s3:PutObject unless the object is encrypted with a KMS key (excludes the central logging bucket).

```
{
  "VERSION": "2012-10-17",
  "STATEMENT": [
    {
      "SID": "DataEncryptionAtRestS3",
      "EFFECT": "Deny",
      "ACTION": ["s3:PutObject"],
      "NOTRESOURCE": "arn:aws:s3::central-logging-${account id}-${aws region}-*/*",
      "CONDITION": {
        "ARNNOTLIKE": {
          "s3:x-amz-server-side-encryption-aws-kms-key-id": "arn:aws:kms:*:*:key/*" },
        "BOOL": { "AWS:PRINCIPALISAWSSERVICE": "false" }
      }
    }
  ]
}
```

scp_DataEncryptionAtRestRDS.json - denies creation of unencrypted RDS instances and clusters.

```
{
  "VERSION": "2012-10-17",
```

```

"STATEMENT": [
  {
    "SID": "DataEncryptionAtRestRDS",
    "EFFECT": "Deny",
    "ACTION": [
      "rds:CreateDBInstance",
      "rds:CreateDBCluster"
    ],
    "RESOURCE": "*",
    "CONDITION": {
      "BOOL": {
        "RDS:STORAGEENCRYPTED": "false",
        "AWS:PRINCIPALISAWSSERVICE": "false"
      }
    }
  }
]
}

```

scp_DataEncryptionAtRestEBS.json - denies creation of unencrypted EBS volumes and instances.

```

{
  "VERSION": "2012-10-17",
  "STATEMENT": [
    {
      "SID": "DataEncryptionAtRestEBS",
      "EFFECT": "Deny",
      "ACTION": [
        "ec2:CreateVolume",
        "ec2:RunInstances"
      ],
      "RESOURCE": "*",
      "CONDITION": {
        "BOOL": {
          "EC2:ENCRYPTED": "false",
          "AWS:PRINCIPALISAWSSERVICE": "false"
        }
      }
    }
  ]
}

```

scp_DataEncryptionAtRestEFS.json - denies creation of unencrypted EFS file systems.

```

{
  "VERSION": "2012-10-17",
  "STATEMENT": [
    {
      "SID": "DataEncryptionAtRestEFS",
      "EFFECT": "Deny",
      "ACTION": ["elasticfilesystem:CreateFileSystem"],
      "RESOURCE": "*",
      "CONDITION": {
        "BOOL": {
          "ELASTICFILESYSTEM:ENCRYPTED": "false",
          "AWS:PRINCIPALISAWSSERVICE": "false"
        }
      }
    }
  ]
}

```

scp_DataEncryptionAtRestSecretsManager.json - denies creating or updating a secret without a KMS key.

```
{
  "VERSION": "2012-10-17",
  "STATEMENT": [
    {
      "SID": "DataEncryptionAtRestSecretsManager",
      "EFFECT": "Deny",
      "ACTION": [
        "secretsmanager:CreateSecret",
        "secretsmanager:UpdateSecret"
      ],
      "RESOURCE": "*",
      "CONDITION": {
        "NULL": { "SECRETSMANAGER:KMSKEYID": "true" },
        "BOOL": { "AWS:PRINCIPALISAWSSERVICE": "false" }
      }
    }
  ]
}
```

scp_DataEncryptionInTransit.json - denies non-TLS requests to the listed data services.

This policy looks complete but is not. For SNS, aws:SecureTransport governs only the publisher's connection to SNS, not SNS's delivery to subscribers, so a topic can still push messages to a plain-HTTP endpoint.

scp_PreventUnencryptedSnsSubscriptions.json (Appendix A.4) closes that gap.

```
{
  "VERSION": "2012-10-17",
  "STATEMENT": [
    {
      "SID": "DataEncryptionInTransit",
      "EFFECT": "Deny",
      "ACTION": [
        "backup:*",
        "ec2:*",
        "elasticfilesystem:*",
        "eks:*",
        "lambda:*",
        "kms:*",
        "rds:*",
        "rds-data:*",
        "rds-db:*", "s3:*",
        "secretsmanager:*",
        "sns:*",
        "states:*",
        "sqs:*"
      ],
      "RESOURCE": "*",
      "CONDITION": {
        "BOOL": {
          "AWS:SECURETRANSPORT": "false",
          "AWS:PRINCIPALISAWSSERVICE": "false"
        }
      }
    }
  ]
}
```

scp_PreventDisableDefaultEBSEncryption.json - prevents disabling account-default EBS encryption (excludes named platform roles).

```
{
  "VERSION": "2012-10-17",
  "STATEMENT": [
    {
      "SID": "PreventDisableDefaultEBSEncryption",
      "EFFECT": "Deny",
      "ACTION": ["ec2:DisableEbsEncryptionByDefault"],
      "RESOURCE": "*",
      "CONDITION": {
        "ARNNotLike": {
          "AWS:PrincipalArn": [
            "arn:aws:iam::*:role/AWSControlTowerExecution",
            "arn:aws:iam::*:role/SomeCentralisedDeploymentRole",
            "arn:aws:iam::*:role/SomeSupportRoles-*"
          ]
        }
      }
    }
  ]
}
```

scp_PreventDisableEbsPublicSharing.json - prevents disabling the EBS snapshot public-access block (excludes named platform roles).

```
{
  "VERSION": "2012-10-17",
  "STATEMENT": [
    {
      "SID": "PreventDisableEbsPublicSharing",
      "EFFECT": "Deny",
      "ACTION": ["ec2:DisableSnapshotBlockPublicAccess"],
      "RESOURCE": "*",
      "CONDITION": {
        "ARNNotLike": {
          "AWS:PrincipalArn": [
            "arn:aws:iam::*:role/AWSControlTowerExecution",
            "arn:aws:iam::*:role/SomeCentralisedDeploymentRole",
            "arn:aws:iam::*:role/SomeSupportRoles-*"
          ]
        }
      }
    }
  ]
}
```

A.3 Key-management lifecycle

rcp_PreventKmsKeyDeletionLessThan30Days.json - denies scheduling KMS key deletion with a pending window shorter than 30 days. Because AWS permits a window of 7-30 days, this forces the maximum 30-day recovery window.

```
{
  "VERSION": "2012-10-17",
  "STATEMENT": [
    {
      "SID": "KmsKeyDeletionWindow",
      "EFFECT": "Deny",
```

```

    "ACTION": "kms:ScheduleKeyDeletion",
    "PRINCIPAL": "*",
    "RESOURCE": "*",
    "CONDITION": {
      "NUMERICLESSTHAN": {
        "KMS:SCHEDULEKEYDELETIONPENDINGWINDOWINDAYS": "30"
      }
    }
  ]
}

```

rcp_PreventKmsLockoutOverride.json - denies creating a key or putting a key policy that bypasses the policy-lockout safety check.

```

{
  "VERSION": "2012-10-17",
  "STATEMENT": [
    {
      "SID": "PreventKmsLockoutOverride",
      "EFFECT": "Deny",
      "ACTION": ["kms:CreateKey", "kms:PutKeyPolicy"],
      "PRINCIPAL": "*",
      "RESOURCE": "*",
      "CONDITION": { "BOOL": { "KMS:BYPASSPOLICYLOCKOUTSAFETYCHECK": "true" } }
    }
  ]
}

```

A.4 Baseline hygiene

scp_AllowedAWSservices.json - denies use of any service outside the approved list (abbreviated; excludes named platform roles).

```

{
  "VERSION": "2012-10-17",
  "STATEMENT": [
    {
      "SID": "ApprovedAWSservices",
      "EFFECT": "Deny",
      "NOTACTION": [
        "account:*",
        "access-analyzer:*",
        "acm:*", "acm-pca:*",
        "apigateway:*",
        "artifact:*",
        "athena:*",
        "autoscaling:*",
        "backup:*",
        "batch:*",
        "billing:*",
        "budget:*",
        "...:*"
      ],
      "RESOURCE": "*",
      "CONDITION": {
        "ARNNOTLIKE": {
          "AWS:PRINCIPALARN": [
            "arn:aws:iam::*:role/AWSControlTowerExecution",
            "arn:aws:iam::*:role/SomeCentralisedDeploymentRole",

```

```

        "arn:aws:iam::*:role/SomeSupportRoles-*"
    ]
}
}
}
]
}

```

scp_EnforceEc2IMDSv2.json - denies launching or modifying instances that do not require version 2 of the Instance Metadata Service (IMDSv2).

```

{
  "VERSION": "2012-10-17",
  "STATEMENT": [
    {
      "SID": "EnforceEc2IMDSv2",
      "EFFECT": "Deny",
      "ACTION": [
        "ec2:RunInstances",
        "ec2:ModifyInstanceMetadataOptions"
      ],
      "RESOURCE": "arn:aws:ec2:*:*:instance/*",
      "CONDITION": {
        "STRINGNOTEQUALS": { "EC2:METADATAHTTPTOKENS": "required" } }
    }
  ]
}

```

scp_EnforceEc2MandatoryTags.json - requires mandatory tags on instance launch/start and on Auto Scaling groups.

```

{
  "VERSION": "2012-10-17",
  "STATEMENT": [
    {
      "SID": "EnforceEc2MandatoryTags1",
      "EFFECT": "Deny",
      "ACTION": ["ec2:RunInstances"],
      "RESOURCE": "arn:aws:ec2:*:*:instance/*",
      "CONDITION": {
        "NULL": {
          "AWS:REQUESTTAG/${TAG1_KEY_NAME}": "true",
          "AWS:REQUESTTAG/${TAG2_KEY_NAME}": "true"
        }
      }
    },
    {
      "SID": "EnforceEc2MandatoryTags2",
      "EFFECT": "Deny",
      "ACTION": ["ec2:StartInstances"],
      "RESOURCE": "arn:aws:ec2:*:*:instance/*",
      "CONDITION": {
        "NULL": {
          "AWS:RESOURCETAG/${TAG1_KEY_NAME}": "true",
          "AWS:RESOURCETAG/${TAG2_KEY_NAME}": "true"
        }
      }
    },
    {
      "SID": "EnforceEc2MandatoryTags3",

```

```

    "EFFECT": "Deny",
    "ACTION": [
        "autoscaling:CreateAutoScalingGroup",
        "autoscaling:UpdateAutoScalingGroup"
    ],
    "RESOURCE": "*",
    "CONDITION": {
        "NULL": {
            "AWS:RESOURCE_TAG/${TAG1_KEY_NAME}": "true",
            "AWS:RESOURCE_TAG/${TAG2_KEY_NAME}": "true"
        }
    }
}
]
}

```

scp_PreventUnencryptedSnsSubscriptions.json - denies creating SNS subscriptions over plaintext HTTP.

```

{
    "VERSION": "2012-10-17",
    "STATEMENT": [
        {
            "SID": "PreventUnencryptedSnsSubscriptions",
            "EFFECT": "Deny",
            "ACTION": ["sns:Subscribe"],
            "RESOURCE": "*",
            "CONDITION": {
                "STRINGLIKEIFEXISTS": { "SNS:PROTOCOL": "http" },
                "BOOL": { "AWS:PRINCIPALISAWSSERVICE": "false" }
            }
        }
    ]
}

```

Appendix B - Governance model detail

This appendix gives the full definitions summarized in Section 3.

B.1 Guiding principles

- **Data as an accountable asset** - data is managed with the discipline applied to other strategic assets, with defined value and named owners accountable for its appropriate use, supported by stewards responsible for day-to-day management.
- **Standardization with federation** - enterprise policies and standards are defined centrally, while operational stewardship is delegated to the business domains that understand the data best.
- **Governance by design** - security, privacy, quality, and lifecycle controls are embedded into platforms and processes from the outset rather than applied retrospectively.
- **Transparency and traceability** - definitions, classifications, metadata, and lineage are documented and discoverable, so data can be understood, trusted, and audited.
- **Fit-for-purpose quality** - quality requirements are set by business need rather than pursued as an absolute and are monitored continuously.
- **Protection proportionate to sensitivity** - protection and access requirements reflect data classification, business risk, and applicable regulatory obligations, supporting continuous compliance rather than periodic audit preparation.

- **Lifecycle management** - data is governed throughout its life, from creation and ingestion through retention, archival, and secure disposal.

B.2 Core capability domains

- **Data ownership and stewardship** - establishes accountability for data assets and the roles responsible for their meaning, quality, appropriate use, and lifecycle.
- **Metadata management** - captures business, technical, and operational metadata and supports discovery, cataloging, classification, and lineage.
- **Data quality management** - defines quality dimensions and rules and the means to measure, monitor, and remediate issues.
- **Data security and privacy** - defines classification and protection requirements and governs access, encryption, masking, and appropriate use.
- **Master and reference data management** - establishes authoritative sources for shared data and promotes consistency across domains.
- **Data lifecycle management** - governs data from creation through retention, archival, and disposal per business and regulatory requirements.
- **Regulatory compliance** - maps controls to legal and regulatory obligations and provides the evidence to demonstrate compliance.
- **Operational governance** - provides the forums, processes, metrics, issue management, and reporting that keep governance active.

These domains are complementary: classification informs security controls, ownership determines accountability for quality, and lifecycle requirements drive retention and disposal. Several of them - notably classification, security and privacy, and lifecycle management - are implemented directly on AWS through the “governing data in place” control families described in Section 4.