

# AWS Network Firewall Master File

---

## 1 — What is AWS Network Firewall and why enterprises use it?

A foundational introduction explaining the purpose, architecture philosophy, and core security role of Network Firewall in modern AWS environments.

## 2 — How AWS Network Firewall is deployed inside VPC architectures?

Covers deployment patterns, single-VPC vs. multi-VPC, centralized inspection VPC, and ingress/egress filtering topologies.

## 3 — How the internal architecture of AWS Network Firewall works?

Explains firewall endpoints, scaling model, flow processing pipeline, distributed capacity, and packet evaluation lifecycle.

## 4 — What are Stateless Rule Groups in AWS Network Firewall?

Deep dive into stateless inspection, match conditions, priority logic, action types, and packet pathway.

## 5 — What are Stateful Rule Groups in AWS Network Firewall?

Detailed explanation of stateful processing engine, connection tracking, Suricata core, rule logic, and threat-detection capabilities.

## 6 — How Rule Priorities, Rule Order, and Policy Hierarchies operate?

Covers how the firewall evaluates stateless vs. stateful rules, override behaviors, default actions, and precedence.

## 7 — How AWS Network Firewall handles inbound, outbound, and east-west traffic flows?

Explains per-direction filtering, workload segmentation, multi-subnet inspection, and handling of return traffic.

## 8 — How network routing and traffic steering works with Network Firewall?

Includes route table changes, firewall endpoints, inspection VPC, centralized vs. distributed routing, TGW integration.

## **9 — How AWS Network Firewall integrates with AWS Transit Gateway?**

Deep integration model, inspection VPC, attachment routing, scalable multi-account architecture.

## **10 — How scaling, performance, and high availability work internally?**

Explains auto-scaling endpoints, throughput distribution, failure handling, and regional redundancy.

## **11 — How to design an enterprise-grade security architecture with AWS Network Firewall?**

Covers layered security, segmentation, multi-VPC governance, and defense-in-depth patterns.

## **12 — How to implement deep packet inspection and advanced threat detection?**

Explains signature-based detection, Suricata rules, custom rules, malware patterns, TLS challenges.

## **13 — How to configure logging, flow logs, alert logs, and centralized analytics pipelines?**

Covers S3/CloudWatch/Kinesis delivery, enterprise log analytics pipelines, SIEM integration, and retention.

## **14 — How to manage firewall policies at scale with AWS Firewall Manager?**

Deep dive into central governance, policy inheritance, drift prevention, organization-wide controls.

## **15 — How to secure hybrid networks and on-premises connectivity with Network Firewall?**

Explains Direct Connect, VPN, inspection between on-prem and cloud, east-west hybrid segmentation.

## **16 — How to integrate Network Firewall with VPC Lattice, Load Balancers, and App Mesh?**

Covers service-to-service security, application layer routing, mesh-based traffic inspection.

## **17 — How to monitor, observe, and troubleshoot Network Firewall in large environments?**

Metrics, alarms, health, packet drops, endpoint issues, and operational excellence.

## 18 — How to design disaster recovery, multi-Region strategies, and failover architectures?

Covers multi-Region firewalls, global routing, failover patterns, replication of policy sets.

## 19 — Full consolidated summary of AWS Network Firewall

A single large combined narrative summarizing the full architecture, rules, integrations, routing, governance, scaling, and security concepts.

## 20 — Common misconceptions, pitfalls, wrong architectures, and how to avoid them

Examines design mistakes, bad routing patterns, incorrect rule logic, and production-impacting traps.

# 1 — What is AWS Network Firewall and why enterprises use it?

---

### 1 — Foundational Purpose of AWS Network Firewall

AWS Network Firewall is a fully managed, cloud-native network security service designed to inspect, analyze, and control network traffic flowing inside and between Amazon VPCs. At its core, it replaces the traditional “firewall appliance in the perimeter” model with a distributed, elastic inspection layer that automatically scales across Availability Zones.

–

Traditional firewalls rely on manually deployed appliances, manual HA pairs, capacity sizing, and static throughput limits. Network Firewall eliminates all of this by allowing AWS to automatically deploy, scale, heal, and update distributed firewall endpoints in your subnets. This shifts the firewall’s role from a physical “box” to a cloud-native traffic inspection layer embedded directly into your network fabric.

–

Enterprises use AWS Network Firewall because it gives them **centralized, scalable, automated, and compliant network filtering** without managing appliances, HA topologies, patching cycles, or cluster upgrades.

### 2 — Core Capabilities Provided by AWS Network Firewall

Network Firewall provides a dual inspection engine: a *stateless engine* for fast packet filtering and a *stateful Suricata-based DPI engine* for deep threat detection. This brings IDS/IPS-grade capabilities into AWS natively.

–

Stateless rules perform header-level decisions (drop, pass, forward to stateful), allowing rapid elimination of irrelevant or unwanted traffic.

–

Stateful rules examine full connection context, packet sequences, DNS queries, HTTP metadata, TLS handshake attributes, and payload signatures. Enterprises rely on this engine for intrusion detection, malware protection, outbound domain filtering, and compliance-enforced risk reduction.

-

Network Firewall also supports logging pipelines, domain filtering, managed rule groups (vendors like Alert Logic, Suricata rulesets), and integration with AWS Firewall Manager for multi-account governance.

3 — Why Enterprises Use Network Firewall Instead of Virtual Appliances

Virtual firewall appliances require EC2 instances, AMIs, manual HA, autoscaling groups, instance families, throughput planning, license management, vendor patching, failover logic, and load-balancing across AZs.

-

Network Firewall removes these burdens: AWS manages all scaling, all endpoint instances, all failovers, and the distributed data-plane architecture. Enterprises do not size hardware, upgrade engines, or maintain clusters. They only create policies and route traffic to the firewall.

-

Because endpoints exist per-AZ and auto-scale horizontally, performance increases automatically with demand. This elastic model allows large organizations to run firewalls at massive scale without architectural redesign.

4 — Enterprise Security Functions Delivered by AWS Network Firewall

Network Firewall allows consistent application of enterprise security rules across all VPCs and accounts. It provides deep packet inspection, intrusion prevention, DNS filtering, IP/domain reputation enforcement, application/port filtering, logging to S3/CloudWatch/Kinesis, and seamless integration with SIEM pipelines.

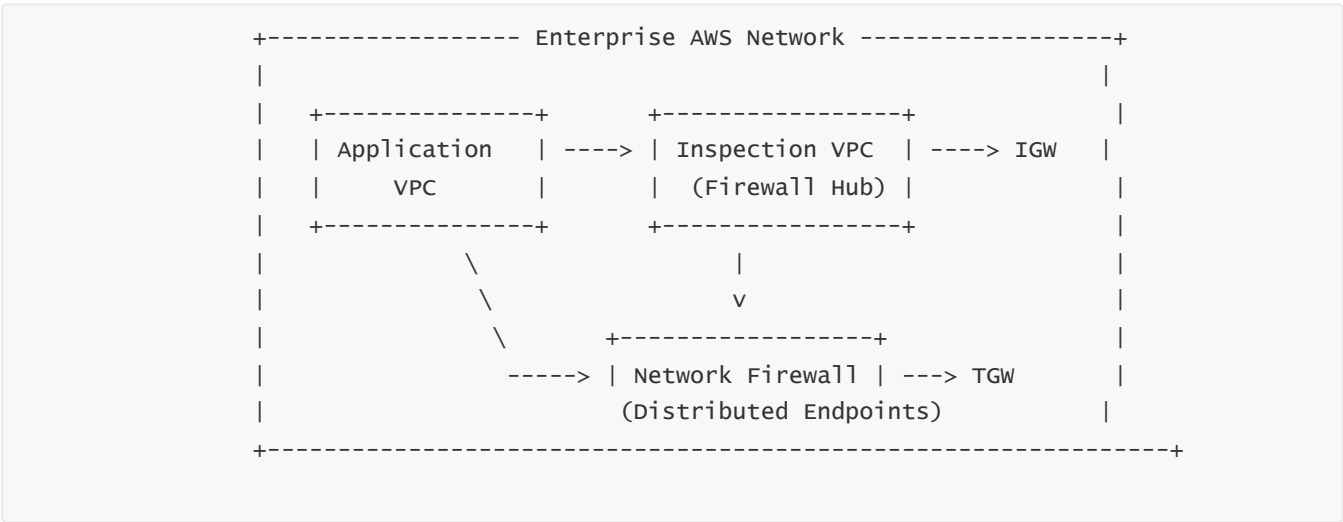
-

It becomes a key component in compliance frameworks such as PCI-DSS, ISO-27001, SOC2, and NIST 800-53. Enterprises often use it to enforce security guardrails across hundreds of accounts using AWS Firewall Manager.

-

Because the firewall sits between VPCs, the internet, and on-premises networks, it provides a unified inspection layer for north-south and east-west traffic patterns.

5 — High-Level Architecture Diagram: Role of AWS Network Firewall



Explanation:

This architecture shows how a centralized inspection VPC receives traffic from other VPCs or a Transit Gateway. Inside the inspection VPC, AWS deploys distributed firewall endpoints in special firewall subnets. All traffic must pass through these endpoints, enabling uniform enterprise security enforcement.

## 2 — How AWS Network Firewall is deployed inside VPC architectures?

### 1 — The Core Deployment Model Inside a VPC

When deploying AWS Network Firewall, you first create a **firewall policy**, then create a **firewall**, and finally attach that firewall to **firewall subnets** in each AZ. AWS automatically deploys multiple firewall endpoint nodes inside those subnets — you do not access or manage these nodes.

To enable inspection, you modify **VPC route tables** to send traffic from application subnets to the firewall subnets. The firewall evaluates the packets and then forwards them to the next hop (Internet Gateway, NAT Gateway, Transit Gateway, another VPC, or on-prem).

### 2 — The Three Standard Deployment Patterns Used by Enterprises

AWS Network Firewall can be deployed in several architectures depending on scale and governance requirements. The three most common are:

**Distributed VPC deployment:** A firewall is deployed inside each application VPC. This is simple but costly at scale because each VPC must maintain its own firewall.

**Centralized Inspection VPC:** A dedicated “security VPC” contains all firewall endpoints. All workload VPCs route their traffic to this inspection VPC through Transit Gateway or VPC peering. This is the most common enterprise pattern.

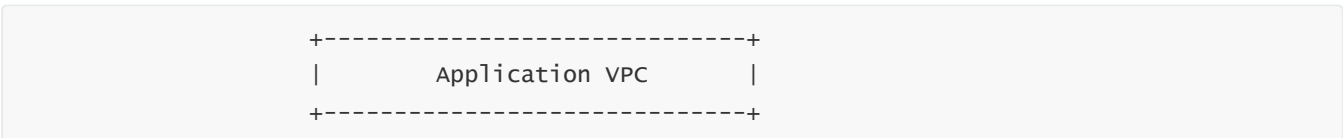
**Transit Gateway Inline Inspection:** The firewall is connected to TGW route domains, and all VPC attachments force traffic through the firewall. This pattern provides maximum scalability and multi-account governance.

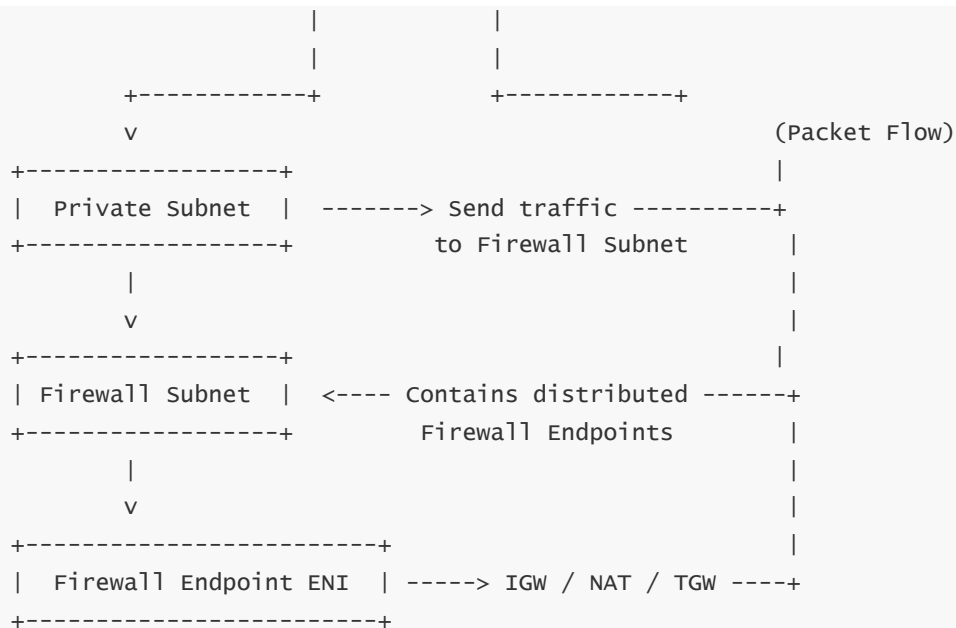
### 3 — Traffic Steering Inside a Single VPC (Route Table Design)

To enforce traffic inspection, private subnets send their traffic not directly to IGW/NAT, but to the firewall’s ENI located in the firewall subnet. After inspection, the firewall forwards packets to their original destination.

This routing mechanism ensures symmetric traffic flow, which is required for stateful processing.

### 4 — Internal VPC Routing Diagram: How Traffic Flows Through the Firewall





### Explanation:

Application traffic flows from private subnets into firewall subnets. The firewall endpoints inspect packets and forward them onward according to policy rules. AWS manages the number of firewall endpoints, scaling them automatically with traffic volume.

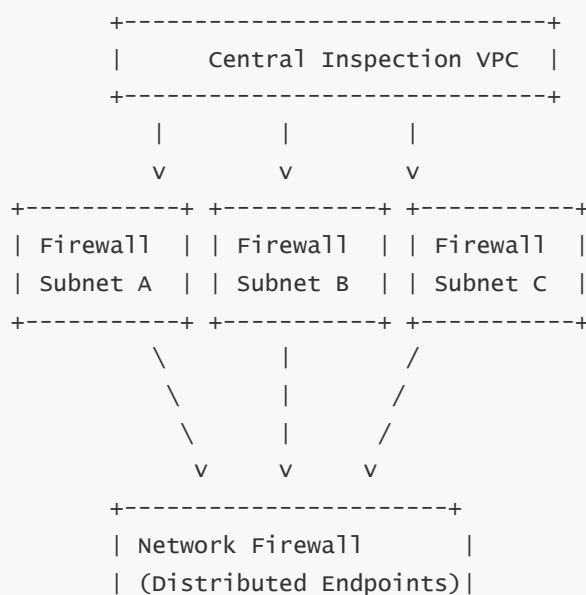
## 5 — Centralized Inspection VPC (Multi-VPC Enterprise Architecture)

In large environments, it is inefficient to deploy a firewall in every VPC. Instead, enterprises build a **centralized inspection VPC**. All workload VPCs send traffic to this VPC through Transit Gateway. The inspection VPC hosts firewall subnets and firewall endpoints.

–

This allows the organization to maintain a single rule set, simplify compliance enforcement, and avoid multi-VPC firewall sprawl.

## 6 — Centralized Inspection Architecture Diagram

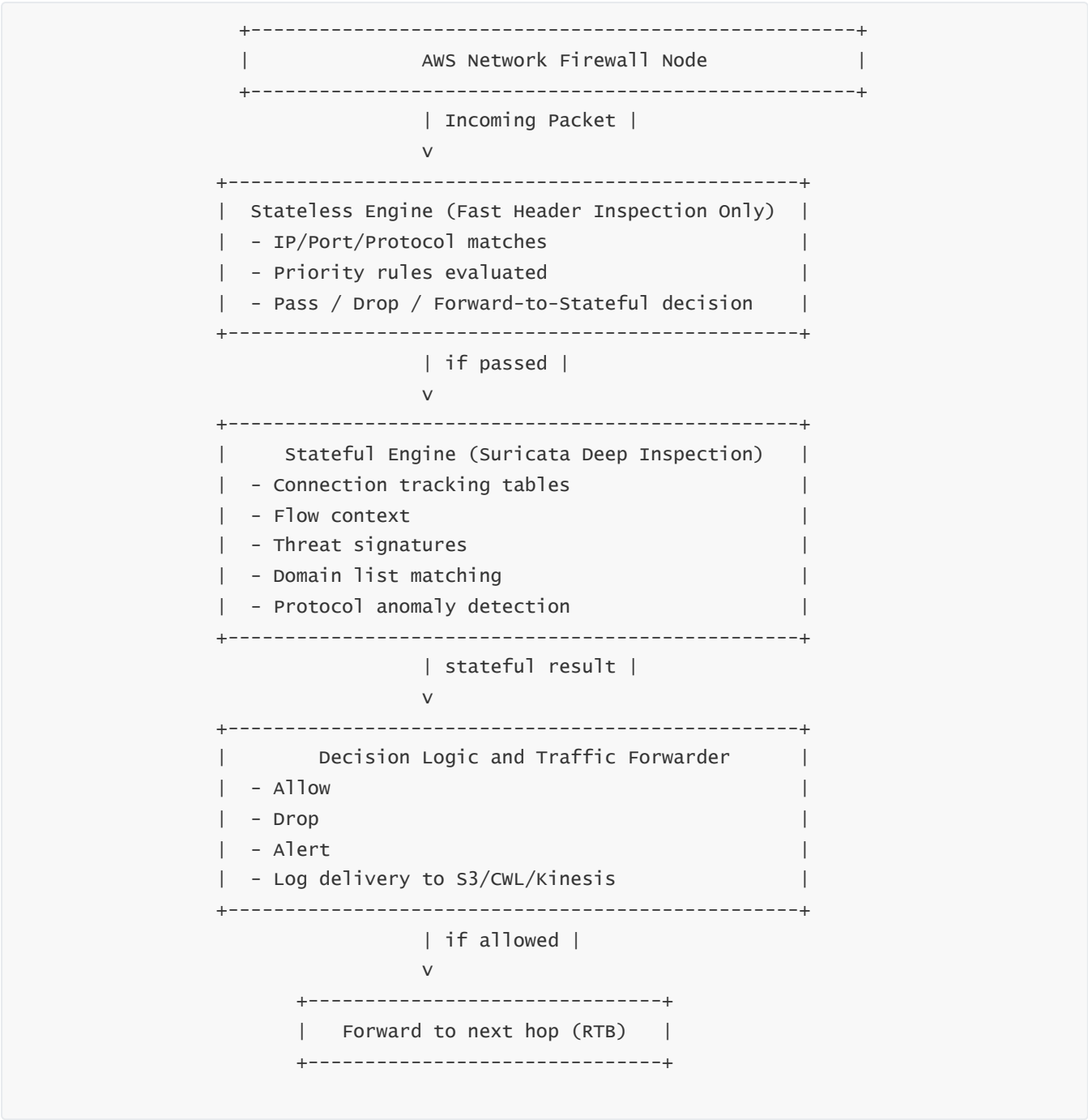




If the packet survives stateless filtering, it is passed to the stateful engine — a Suricata-based deep inspection engine that maintains connection context. The stateful engine tracks flow tables, connection sequences, TCP handshake states, session metadata, and any previous packets from the same flow. This enables the firewall to perform Intrusion Prevention System (IPS)-level detection, domain filtering, signature matching, and advanced threat detection.

Finally, the decision layer chooses how to forward or drop the packet. If allowed, the packet exits the endpoint and is delivered to the next hop. If the firewall policy indicates logging, the packet metadata or full inspection results are sent to logging pipelines.

3 — Internal Flow Diagram: Packet Processing Lifecycle



This diagram shows the three major internal phases of processing inside a firewall endpoint.



-

The stateless engine provides quick filtering and reduces load on deeper layers.

-

The stateful engine is where true threat detection, flow analysis, and Suricata-based rules execute.

-

The decision layer determines the fate of the packet and integrates with the VPC routing system to deliver the packet to the next hop.

#### 4 — The Distributed Endpoint Architecture Inside Each Availability Zone

AWS deploys the firewall endpoints as **multiple ENI-based nodes** inside the firewall subnet of each AZ. Instead of a single device, AWS maintains a cluster of nodes. When new flows arrive, the Network Firewall service automatically distributes them across available nodes. Each node tracks state for the flows it handles, ensuring consistent behavior for the duration of the connection.

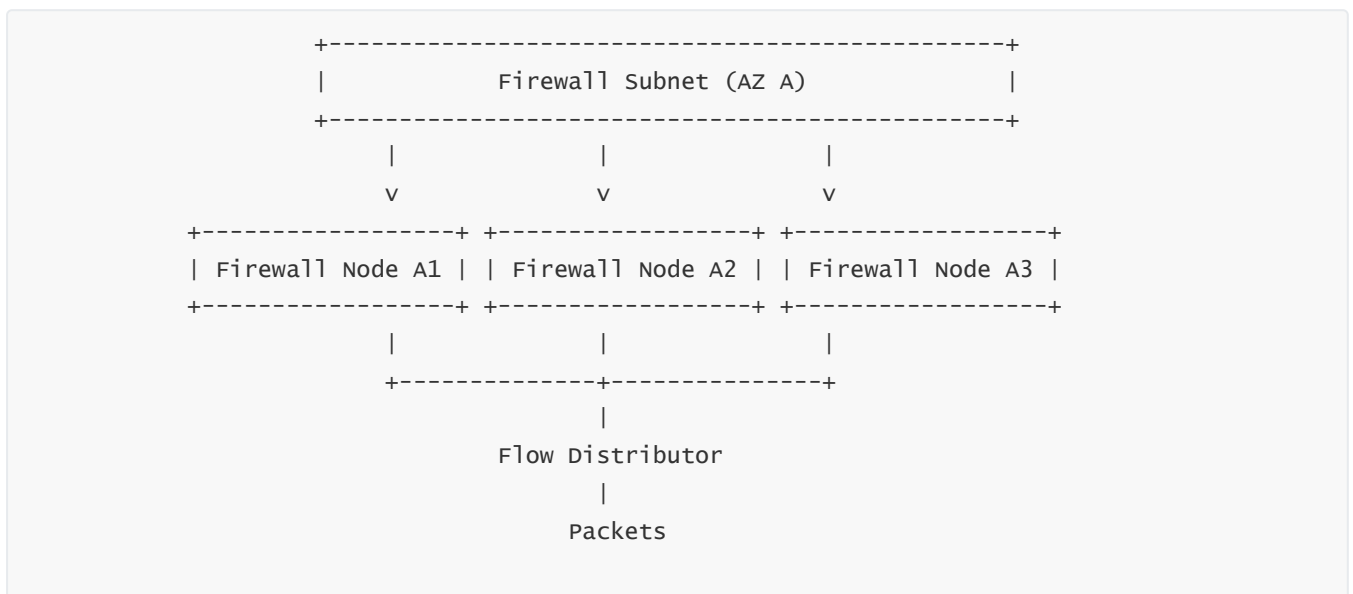
-

If a node becomes unhealthy, AWS automatically replaces it without any customer impact. If traffic volume grows, new nodes are added automatically. If volume decreases, AWS quietly scales the cluster down. This elasticity is what makes Network Firewall fundamentally different from virtual appliances, which require manual scaling and failover configuration.

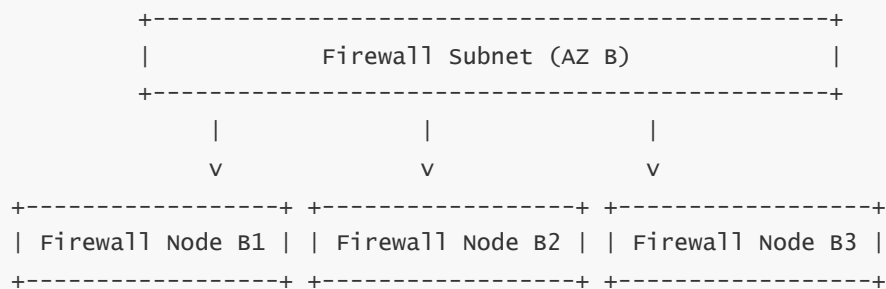
-

This model also ensures that all AZs receive dedicated, isolated endpoint clusters. This provides fault containment — a failure in AZ A does not affect AZ B. High availability is intrinsic because each AZ has its own independent firewall compute plane.

#### 5 — Internal Endpoint Distribution Diagram



Similarly, each AZ has its own cluster:



This AZ-scoped distribution ensures consistent traffic handling regardless of failures and allows AWS to support extremely high throughput by spreading the load across dozens of nodes if necessary.

## 6 — How Flow Consistency Works Internally

For every new connection — such as a TCP handshake — the Network Firewall assigns that flow to a specific firewall node. This assignment remains fixed until the flow terminates. The stateful engine stores connection metadata, sequence numbers, window sizes, DNS details, or HTTP header metadata depending on the rule evaluation needs.

If subsequent packets from the same connection arrive, they must go to the *same* node. AWS ensures this using flow hashing and internal load distribution algorithms. If packets were routed to different nodes, the connection would break because no other node would have the associated state.

If a node fails mid-session, AWS automatically rebalances the session with minimal interruption by reconstructing partial state. The service is designed to maintain the maximum possible session continuity, though very edge-case microflows may reset — the system is optimized for long-lived, stable flows.

## 7 — The Policy Execution Model Inside the Firewall Engine

Internally, firewall policies combine:

- rule groups
- priority ordering
- stateless rule sequences
- stateful rule layers
- default actions
- drop/alert logging settings

These elements compile into a hierarchical evaluation tree inside the firewall engine. When you create or update a policy, AWS compiles and distributes the updated rule set across all nodes in every AZ. This ensures consistency.

The engine maintains an internal “policy snapshot” for each node. When policies are updated, nodes atomically switch to the new snapshot without interrupting running flows. Stateless rules take effect immediately; stateful rules transition more carefully to preserve connection integrity.

-

All rule evaluations flow in predictable order: stateless evaluation first, followed by stateful evaluation, followed by final action.

8 — Internal Decision Diagram: Rule Evaluation Hierarchy



This hierarchical evaluation ensures predictable behavior and optimizes performance. The stateless engine handles easy decisions. The Suricata engine handles deep inspection. The decision layer outputs the final verdict.

# 4 — What are Stateless Rule Groups in AWS Network Firewall?

1 — The Purpose and Role of Stateless Rule Groups

Stateless rule groups in AWS Network Firewall operate as the very first decision layer for all incoming packets that enter a firewall endpoint. Their purpose is to provide extremely fast, header-level evaluation of traffic without looking at any connection state or historical packet context. Because the stateless engine does not maintain flow tables or track session metadata, it can evaluate packets at very high throughput, making it ideal for early filtering, basic allow/block operations, protocol validation, and traffic steering before the deeper stateful engine becomes involved.

-

In enterprise deployments, stateless rules act as the “front gate” — they perform rapid checks such as dropping known unwanted traffic, forwarding only specific protocols to deeper inspection, or bypassing inspection altogether for safe traffic. Because they are computationally lightweight, they help reduce load on the stateful engine and optimize overall firewall performance.

**2 — How Stateless Evaluation Works Internally**

The stateless engine evaluates each packet independently. It treats every packet as new, without any knowledge of whether it belongs to an existing TCP connection, handshake, or established flow. The engine matches packet headers against configured rules in priority order. A high-priority rule is always processed before a low-priority rule.

–

Each rule defines match fields such as source IP, destination IP, source port, destination port, protocol, TCP flags, or packet direction. When a packet matches a rule, the firewall executes the rule’s action and stops evaluating further stateless rules. This deterministic “first-match wins” model ensures consistent behavior and allows administrators to design precise filtering structures.

–

The action determines whether the packet is dropped, passed to the stateful engine, forwarded without stateful inspection, or custom-decision actions such as copying the packet.

**3 — The Main Actions Available in Stateless Rule Groups**

Stateless rules support several critical actions: “drop,” “pass,” “forward to stateful,” “alert,” and “custom action.”

–

A drop action discards the packet immediately with no further evaluation. This is the fastest way to eliminate undesired traffic such as botnet probes, malformed packets, or disallowed ports.

–

A pass action allows the packet to skip deeper stateful inspection and continue routing without additional analysis. This is used when certain traffic is guaranteed to be safe or when enterprises choose to avoid inspecting specific classes of traffic at deeper layers.

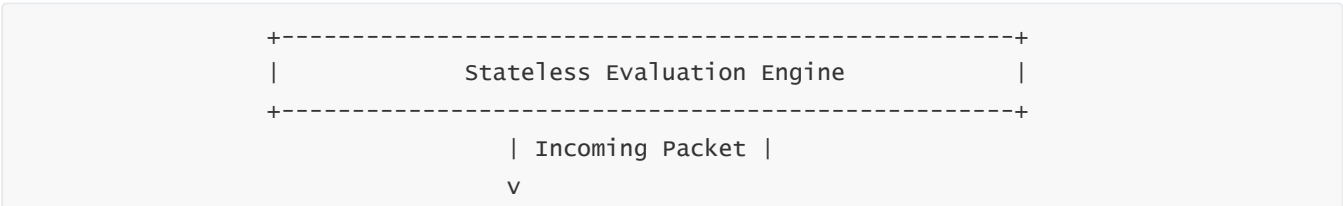
–

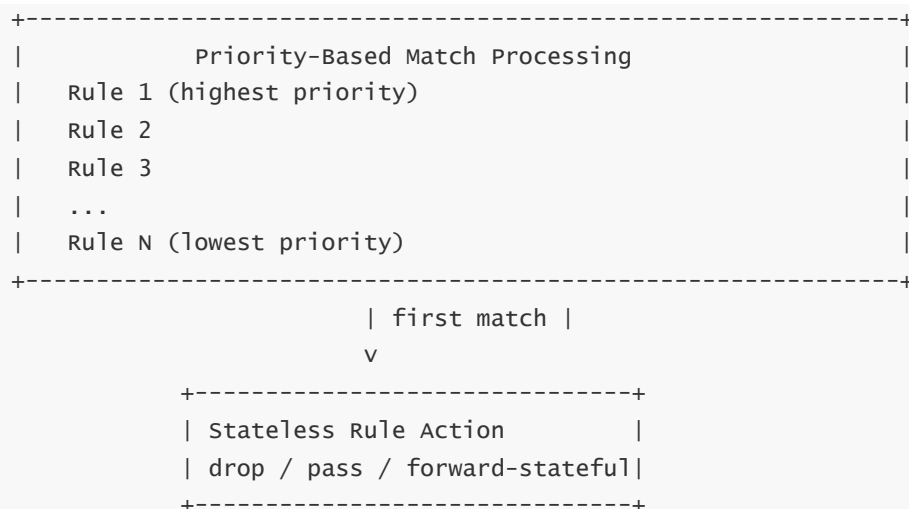
A forward\_to\_stateful action is one of the most important: it instructs the firewall to pass the packet to the stateful engine for connection tracking and deep packet inspection. Without this action, the stateful engine never sees the packet.

–

Custom actions attach metadata tags that can be logged or interpreted by external systems. They do not affect traffic flow directly but are used for advanced analytics pipelines.

**4 — Stateless Rule Evaluation Flow Diagram**





This diagram demonstrates the one-way evaluation model of stateless rules.

–

Traffic is matched top-down, and only the first matching rule determines the action.

–

If no stateless rule matches, the firewall executes the stateless default action defined in the firewall policy.

## 5 — Importance of Stateless Rules in Enterprise Architectures

Enterprises use stateless rules to ensure consistent, predictable traffic filtering before packets reach the heavier stateful inspection engine. This enables them to offload common filtering tasks such as blocking known ports, dropping non-compliant protocols, or routing DNS traffic to enforced resolvers.

–

Stateless rules also prevent unnecessary CPU usage on stateful engines. If certain traffic — such as internal health checks — does not require deep inspection, stateless pass actions can directly route the traffic onward, avoiding unnecessary inspection costs.

–

Finally, stateless filtering plays a key role in protecting internal services by preventing malformed packets from reaching deeper layers. Because stateless rules evaluate at extremely high speed, they form the main defensive layer against volumetric scanning and port-probe attacks.

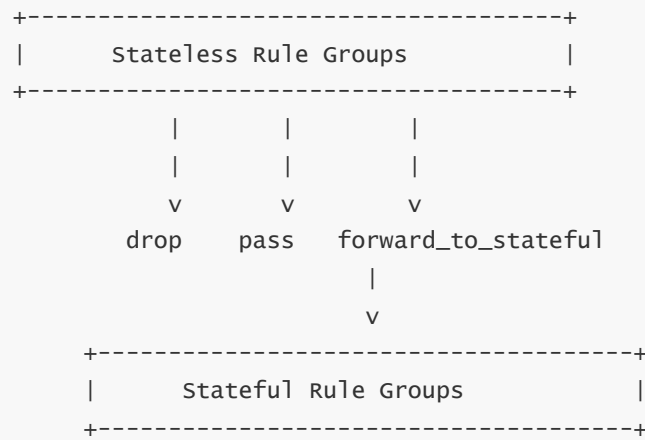
## 6 — How Stateless and Stateful Layers Work Together

Stateless and stateful layers have a hierarchical relationship. The stateless layer always executes first and can decide whether a packet should move to the stateful engine. If the stateless layer chooses to pass traffic, the deeper stateful engine never sees it. This allows administrators to design sophisticated multi-tiered filtering architectures where only relevant traffic is deeply inspected.

–

In practice, stateless rules usually enforce broad restrictions, while stateful rules enforce contextual and behavioral restrictions. This separation ensures both flexibility and performance.

## 7 — Combined Stateless-to-Stateful Decision Diagram



This diagram demonstrates the branching model created by stateless decisions.

–

Packets may be dropped immediately, bypass deep inspection, or be routed into Suricata for full connection and signature evaluation.

–

This layered model is a key architectural strength of AWS Network Firewall and is one of the reasons enterprises adopt it for large-scale, high-performance security.

## 5 — What are Stateful Rule Groups in AWS Network Firewall?

### 1 — The Purpose and Role of Stateful Rule Groups

Stateful rule groups in AWS Network Firewall provide the deep, connection-aware security analysis that enterprises require for intrusion detection, intrusion prevention, domain filtering, signature inspection, and advanced threat detection. Unlike stateless rules, which inspect only individual packets in isolation, stateful rules maintain a full **connection state table**, allowing the firewall to understand whether packets belong to an existing session, whether a TCP handshake is valid, whether packet sequences are correct, and whether traffic behavior aligns with known safe or malicious patterns.

–

Stateful rule groups operate using a **Suricata-based deep packet inspection (DPI) engine**, which is the same modern open-source engine used in many enterprise-grade IDS/IPS systems. This gives AWS Network Firewall the ability to interpret higher-level protocols, detect deviations from expected behavior, evaluate signatures, apply domain filters, and identify threats like malware callbacks, command-and-control communications, port scanning, protocol misuse, and suspicious cross-VPC traffic.

–

In enterprise architectures, stateful rule groups form the core detection and protection layer. They enforce organizational security policies, compliance requirements, and threat-prevention rules that cannot be achieved with simple stateless packet filtering.

### 2 — How Stateful Evaluation Works Internally (Suricata Processing Pipeline)

Stateful inspection begins once a packet is forwarded to the stateful engine by the stateless layer. This engine maintains a **flow table**, where every new connection is inserted when the initial packet arrives. The table includes protocol details, IP/port mappings, sequence numbers, handshake metadata, and timestamps.

-

The Suricata engine then parses traffic at multiple layers, from IP/TCP/UDP all the way up to application protocols such as DNS, HTTP metadata, SMB headers, TLS handshakes, and more (excluding full TLS decryption unless rules match on metadata-only patterns). The engine evaluates each packet in the context of previous packets, enabling it to detect anomalies such as out-of-order sequence numbers, invalid TCP flag combinations, or unexpected protocol flows.

-

Once a flow is established, Suricata applies signature rules, domain filtering rules, protocol-normalization rules, and behavioral correlation to identify malicious or non-compliant traffic. If a rule triggers, the engine can drop the packet, alert, log metadata, or permit the packet depending on the configured rule action.

### 3 — Types of Stateful Rules (Signatures, Domains, IPS Rules)

Stateful rule groups support several rule types that expand the inspection capabilities far beyond the stateless model. Suricata signatures allow matching on packet contents, payload signatures, header fields, protocol anomalies, or known attack patterns. This includes IPS patterns commonly found in enterprise security solutions.

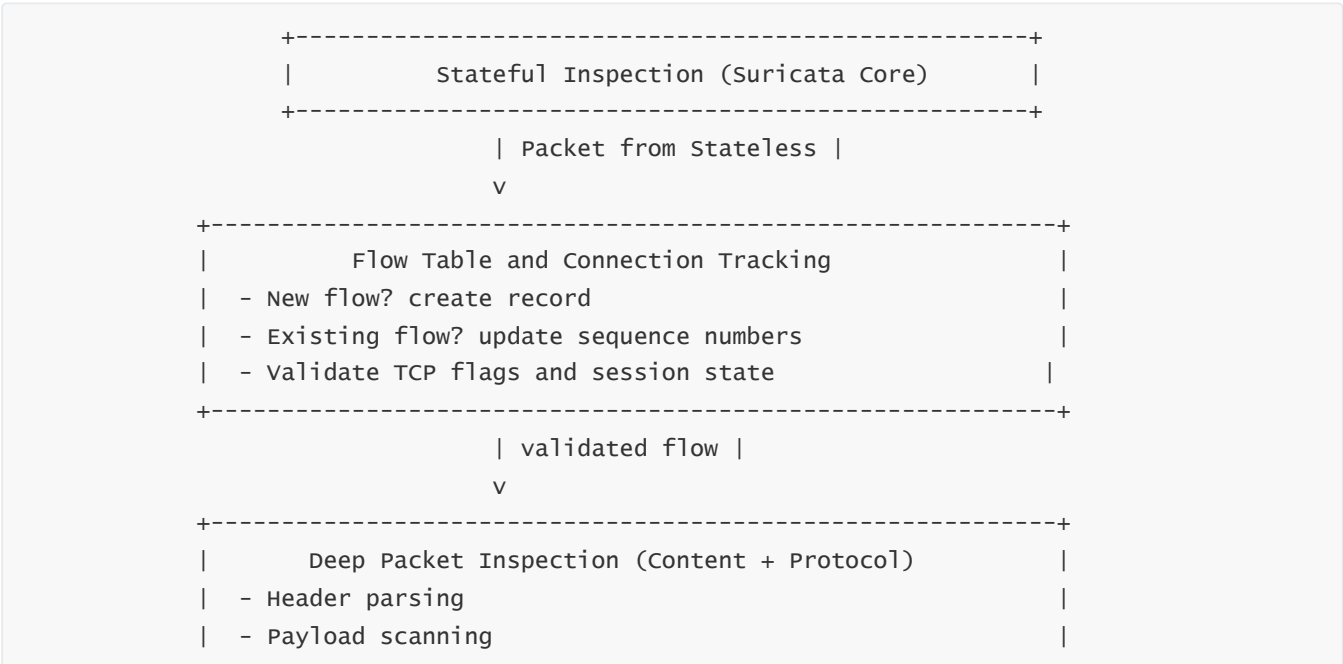
-

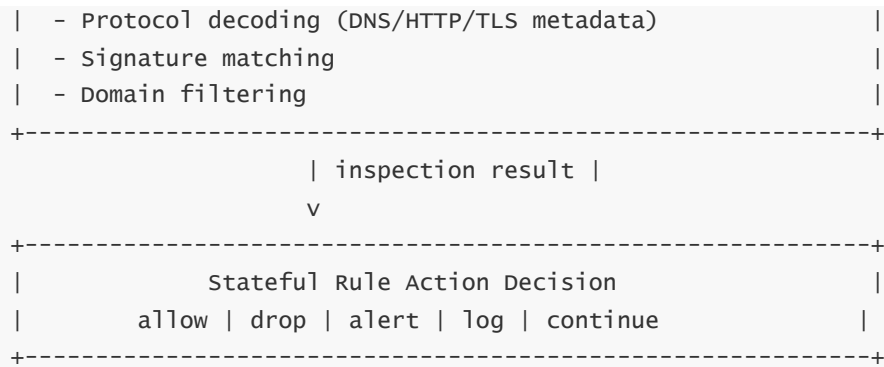
Domain filtering rules allow the firewall to block or allow DNS queries or responses that match specific domain name lists. This is critical for preventing malware from communicating with command-and-control servers.

-

Behaviorally oriented IPS rules detect issues such as port scans, abnormal TCP flag sequences, invalid packet lengths, and suspicious protocol flows. These rules help detect attacks that do not rely on simple payload signatures but instead rely on abnormal behavior patterns.

### 4 — The Stateful Processing Pipeline Diagram





This multi-stage architecture allows the firewall to combine connection awareness with content and behavioral inspection, enabling enterprise-level threat protection.

## 5 — Signature-Based Threat Detection and DPI Behavior

The stateful engine's Suricata rules enable pattern-based detection of malicious traffic. For example, if a packet contains a known malware signature, a suspicious HTTP header, or a DNS query for a malicious domain, the stateful engine can drop the traffic immediately.

–

The engine also detects protocol-level inconsistencies such as invalid SMB negotiation attempts, malformed TLS handshake structures, suspicious SSH banners, and other indicators that point to malicious activity or misconfigured applications.

–

Because Suricata signatures are community-supported and widely adopted, AWS Network Firewall can use proven rule formats, allowing customers to bring existing IDS/IPS rule sets into the cloud. This gives enterprises continuity between on-premises and cloud threat detection strategies.

## 6 — Flow Consistency and Stateful Session Behavior

In stateful inspection, the firewall must ensure that all packets belonging to the same session are processed by the same firewall endpoint node. AWS accomplishes this by performing deterministic flow hashing so that each new flow is consistently assigned to a specific node in an Availability Zone.

–

If a session remains active, subsequent packets always return to the same endpoint, ensuring the Suricata engine has the necessary context to evaluate packets correctly. If stateful inspection were to lose flow consistency, the engine would be unable to track sequence numbers, handshake progress, or protocol state, causing packet drops or invalid alerts.

–

If an endpoint becomes unhealthy, AWS attempts to restore flow continuity by reassigning flows and reconstructing partial state. While extremely rare, momentary flow disruptions can occur during reconvergence, but AWS designs the system to minimize this impact.

## 7 — Stateful Action Outcomes and Logging Behavior

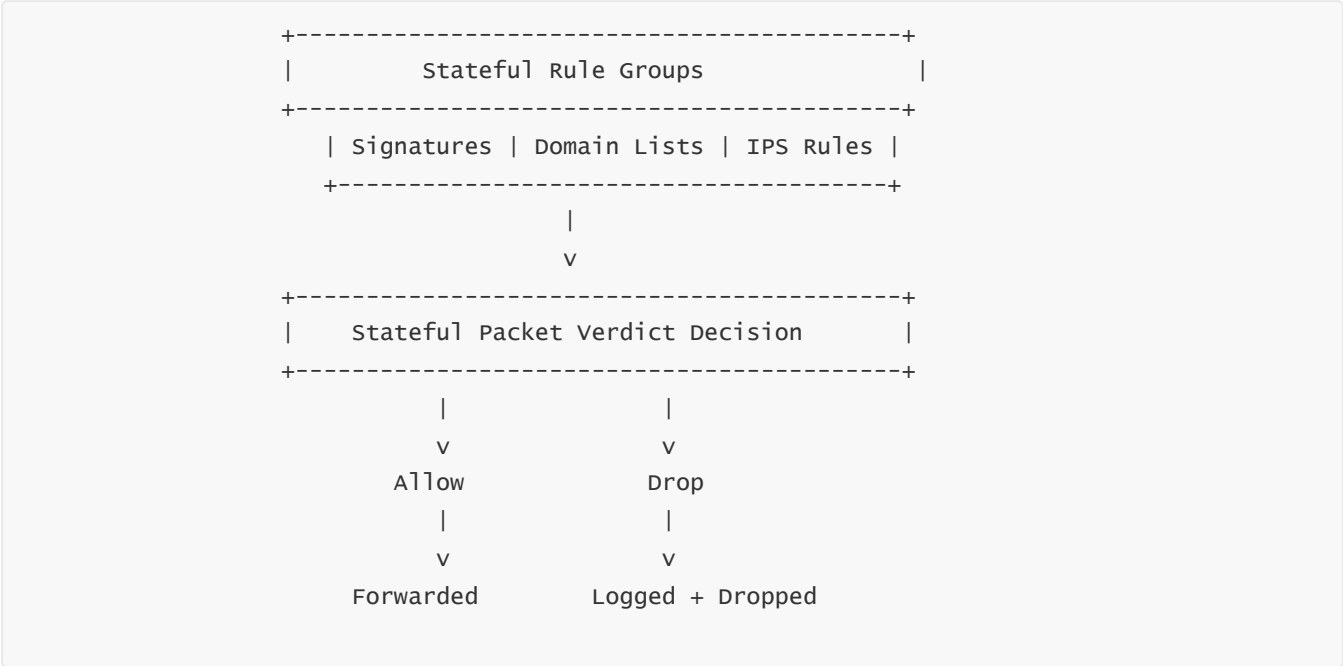


When a stateful rule matches traffic, the firewall triggers one of several possible actions: allow, drop, or alert. An allow action permits the packet to continue, while a drop action terminates the packet immediately. An alert action does not change packet forwarding but logs the event for analytics and SIEM correlation.

Stateful logs contain rich metadata such as domain names, signature IDs, rule identifiers, flow metadata, and protocol-specific fields. These logs feed enterprise detection pipelines, enabling threat hunting, compliance reporting, and SOC visibility.

Because logs can be streamed to S3, CloudWatch Logs, or Kinesis Data Firehose, enterprises can push firewall logs into Splunk, ELK, Snowflake, or security analytics platforms for deeper analysis.

8 — Stateful Evaluation and Policy Integration Diagram



This diagram illustrates how stateful rule groups contribute to the final decision after deep inspection. Stateful outcomes combine with stateless outcomes to produce the final traffic verdict.

# 6 — How Rule Priorities, Rule Order, and Policy Hierarchies operate in AWS Network Firewall

1 — Why Rule Priority and Ordering Matter in Network Firewall

AWS Network Firewall is designed around a strict, deterministic evaluation sequence where stateless rules always evaluate before stateful rules, and where rule priority determines which rule executes first. Because traffic inspection is a multi-layered pipeline, the firewall must know which rule to apply when multiple rules match the same packet. Rule ordering becomes critical because stateless rules may decide whether a packet is dropped immediately, passed onward without deeper inspection, or forwarded to the Suricata-based stateful engine for deep inspection.

–

Enterprises rely heavily on rule priorities to enforce consistent outcomes, prevent traffic bypass, avoid rule conflicts, and ensure that high-severity rules (like known malicious IPs or disallowed ports) execute before broader “allow” rules. Incorrect priority design can cause serious outages or security gaps, so AWS Network Firewall uses a clear and predictable priority system.

## 2 — How Stateless Rule Priorities Work Internally

Stateless rule groups follow a *top-down, first-match-wins* evaluation model. Each stateless rule has a numerical priority. A lower number means higher priority. When a packet enters the stateless engine, the firewall evaluates rules in ascending priority order: priority 10 before 20, 20 before 30, and so on.

–

Once a matching rule is found, the stateless engine stops evaluating additional rules and immediately performs the rule’s action. Because the stateless engine does not maintain any session or flow state, every packet is evaluated independently using this strict sequence.

–

This means an “allow” rule with priority 200 may be overridden by a “drop” rule with priority 100. Enterprise security teams often place high-severity denies in the top-priority slots so that no lower rule can override them.

## 3 — How Stateful Rule Order Works (Suricata Sequencing Model)

Stateful rule groups do not rely on simple priority numbers. Instead, Suricata processes stateful rules based on internal classification: flow direction, protocol type, signature categories, and the logical arrangement of rules inside the rule group. Rules are grouped by category (domain filtering, signature matching, protocol detection), and Suricata prioritizes rules based on how specific they are and how early they must be applied within the inspection pipeline.

–

For example, flow-establishment rules (TCP handshake validation, sequence number tracking) are evaluated before payload signature rules. DNS domain filtering rules apply only to DNS flows and are processed early within the DNS decoder pipeline.

–

Suricata’s engine uses a compiled rule tree that optimizes matching for performance, meaning rules may not execute in the exact order they appear in the rule file but in the optimized structure the engine builds internally.

## 4 — How Policy Hierarchies Combine Stateless and Stateful Rule Groups

AWS Network Firewall policies contain two main sections: **stateless rule groups** and **stateful rule groups**. The firewall always evaluates stateless rules first. If the stateless rules decide “drop,” the packet is discarded immediately with no further evaluation. If they decide “pass,” the packet proceeds directly to routing without stateful inspection. Only if stateless rules decide “forward\_to\_stateful” does the stateful engine see the packet.

–

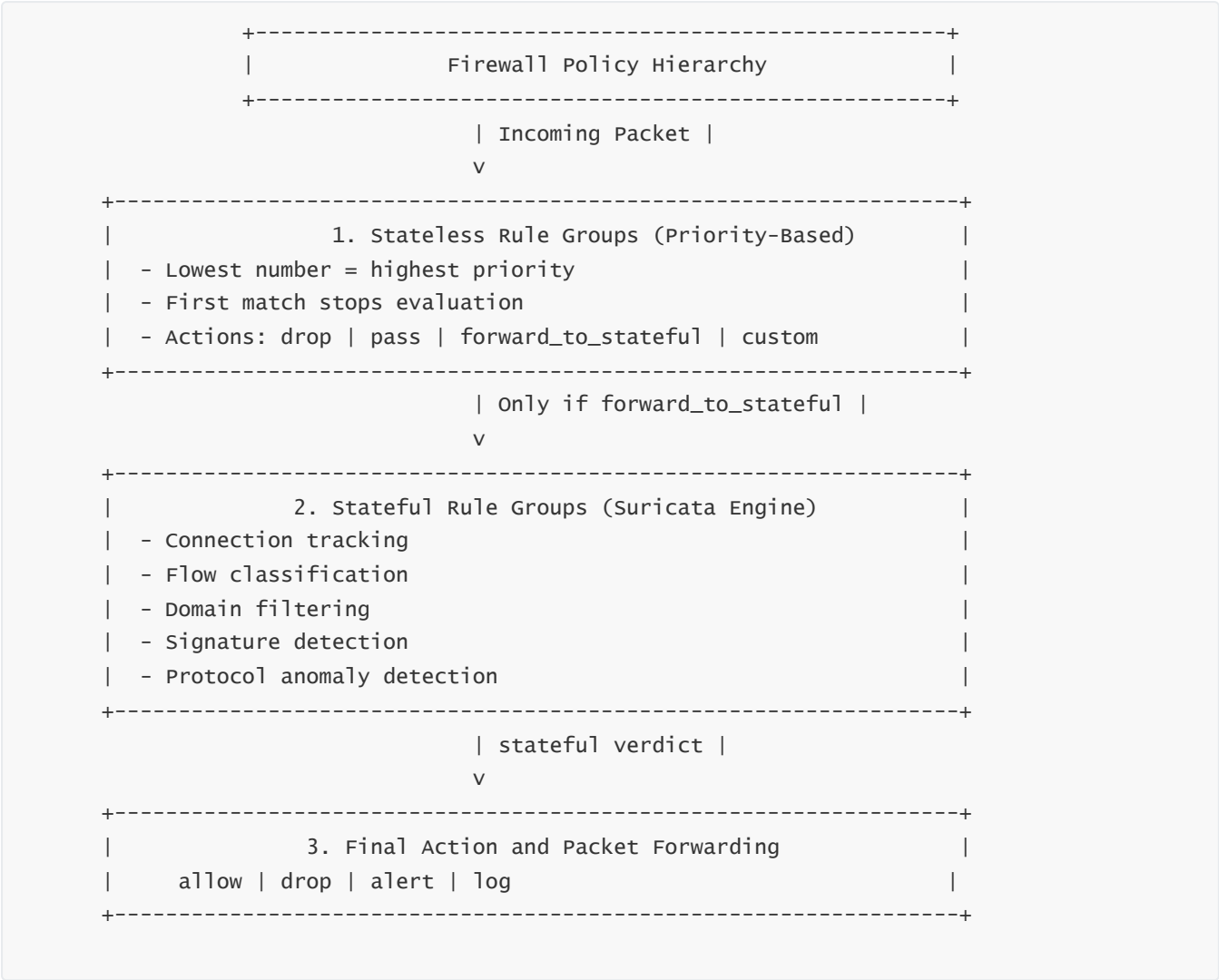
This layering creates a predictable hierarchy:

(1) Stateless rules determine the packet’s path and early decisions.

- (2) Stateful rules apply deep inspection only when allowed by stateless rules.
- (3) The policy's default stateless action applies if no stateless rule matches.
- (4) The policy's default stateful action applies if no stateful rule matches.
- 

Because both engines have defaults, a firewall policy must be designed carefully to ensure traffic is not accidentally allowed or dropped due to misconfigured default behaviors.

### 5 — Rule Processing Hierarchy Diagram



**Explanation:**

This diagram shows the structured pipeline the firewall uses. Stateless rules always evaluate first and determine whether deep inspection is required. Stateful rules only apply after stateless logic explicitly forwards traffic to them. The final decision layer executes the outcome and passes traffic to the next hop.

### 6 — Understanding Default Actions (Stateless Default vs. Stateful Default)

Each firewall policy includes two different default actions:

- The **stateless default action**, used when no stateless rule matches a packet.

- The **stateful default action**, used when the stateful engine receives a flow that does not match any stateful rule.
- 

These defaults are extremely important. If the stateless default action is “forward\_to\_stateful,” you are essentially forcing all unmatched traffic into deep inspection. If the default is “drop,” then all unmatched packets are discarded immediately. Enterprises often use default “forward\_to\_stateful” to ensure visibility and comprehensive inspection.

-

The stateful default action is typically “alert” or “pass,” depending on the enterprise’s policy. A default “drop” in the stateful engine is very strict and often causes unintentional outages unless carefully tested.

**7 — How Rule Conflicts Are Resolved in Network Firewall**

Rule conflicts occur when two rules could match the same traffic but specify different outcomes. The firewall resolves these conflicts as follows:

- In stateless rules, priority determines the winner: the rule with the lowest priority number always wins.
- In stateful rules, Suricata resolves conflicts using internal precedence, giving precedence to higher-specificity signatures (more specific match conditions override broader rules).
- Policy-level conflicts are resolved by the firewall policy hierarchy: stateless decision always overrides stateful evaluation if stateless chooses “drop” or “pass.”
- 

Because the system is deterministic, enterprises avoid ambiguous behavior as long as they correctly design their rule hierarchies and priorities.

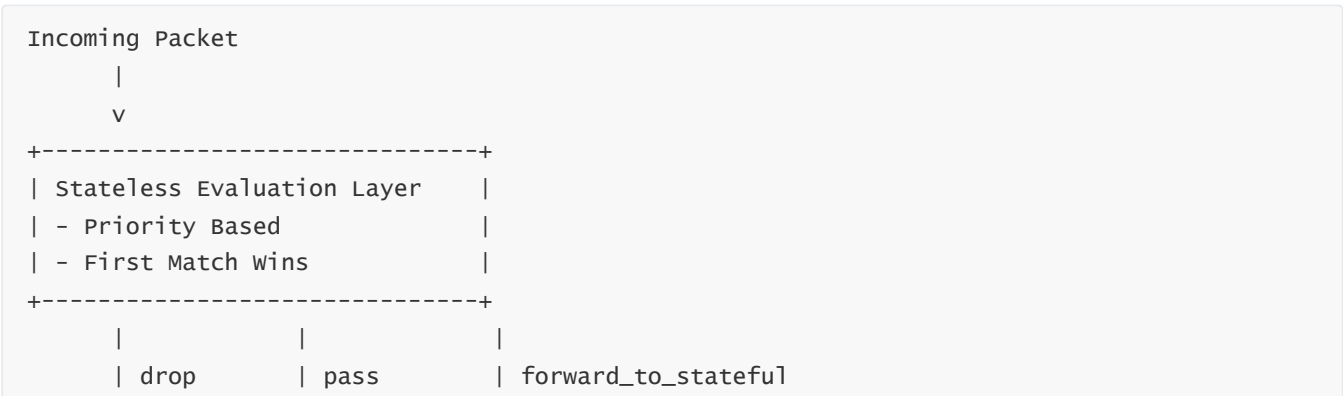
**8 — Enterprise Governance and Large-Scale Priority Management**

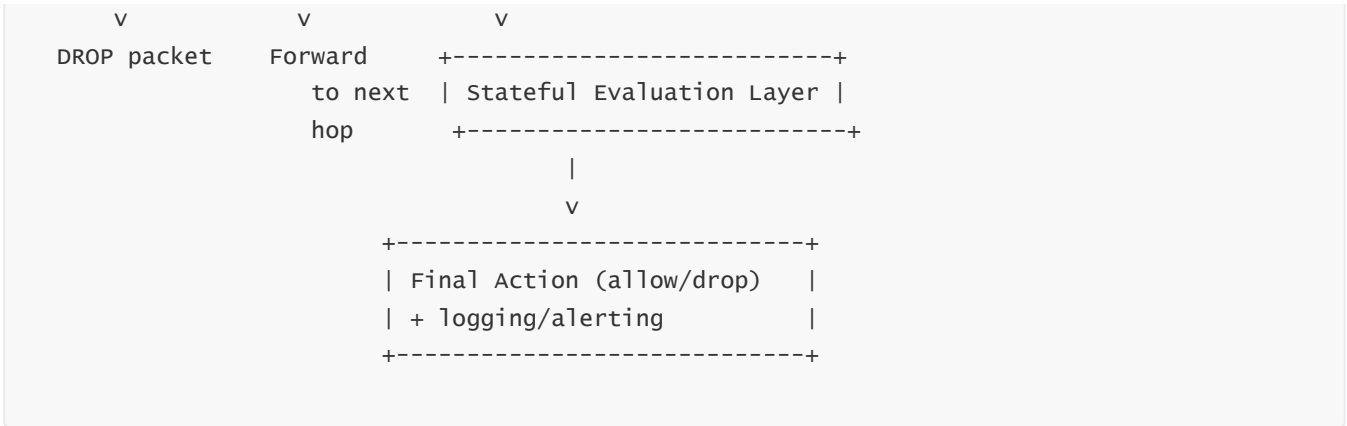
In multi-account environments, AWS Firewall Manager is often used to ensure consistent rule hierarchy across all accounts and VPCs. Firewall Manager can enforce mandatory stateless high-priority deny lists at the top of all firewalls, ensuring no developer or application owner can inadvertently override corporate security decisions.

-

This model gives enterprise security teams full control over rule precedence, even across hundreds of accounts, ensuring no drift or bypass occurs.

**9 — Full Evaluation Flow Diagram (Combined End-to-End)**





**Explanation:**

The stateless engine acts as the gatekeeper. Only packets allowed by stateless rules reach the stateful engine. After deep inspection, the final verdict determines whether packets are forwarded or dropped. This ensures predictable and high-performance packet processing.

# 7 — How AWS Network Firewall handles inbound, outbound, and east-west traffic flows

## 1 — Understanding Traffic Directions in Cloud Architectures

In AWS environments, traffic moves through three fundamental directions: inbound (internet → VPC), outbound (VPC → internet), and east-west (VPC → VPC or subnet → subnet inside the cloud). AWS Network Firewall must evaluate all three directions using the same distributed, auto-scaling inspection fabric, but the traffic paths and routing designs differ significantly for each direction.

–

Inbound traffic typically flows from Internet Gateway or ALB/NLB into workloads. Outbound traffic flows from private subnets through NAT gateways or egress gateways. East-west traffic flows laterally within the organization, such as between microservices, between VPCs, or between Regions or on-prem environments.

–

The firewall must enforce symmetry for all stateful connections. That means both directions of the traffic must pass through the same firewall endpoints so the Suricata engine can maintain accurate flow state.

## 2 — How AWS Network Firewall Handles Inbound (North-South) Traffic

Inbound traffic begins outside AWS and enters a VPC through an Internet Gateway or via a load balancer. If inspection is required, routing tables must steer this inbound traffic first into firewall subnets instead of directly to application subnets.

–

Once routed to the firewall endpoint, stateless rules decide whether the traffic is dropped immediately or forwarded to stateful inspection. Stateful inspection validates the entire TCP handshake, performs protocol analysis, detects malicious payloads, and enforces inbound security restrictions (blocked ports, disallowed IPs, threat signatures, domain restrictions).

-

After inspection, allowed traffic is forwarded to the workload's private subnets. The return traffic follows the reverse path, ensuring symmetric flow.

**Inbound Flow Diagram**



**Explanation:**

Inbound traffic is forced into the firewall before reaching workloads. This allows DPI to protect applications from external threats such as port scanning, exploit kits, C2 callbacks, and malicious payloads.

**3 — How AWS Network Firewall Handles Outbound (North-South Egress) Traffic**

Outbound filtering is one of the most common use cases. Private workloads often need internet access for updates, API calls, telemetry, third-party services, or partner systems. Without a firewall, outbound security is minimal and relies primarily on security groups and NACLs, which are insufficient for enterprise governance.

-

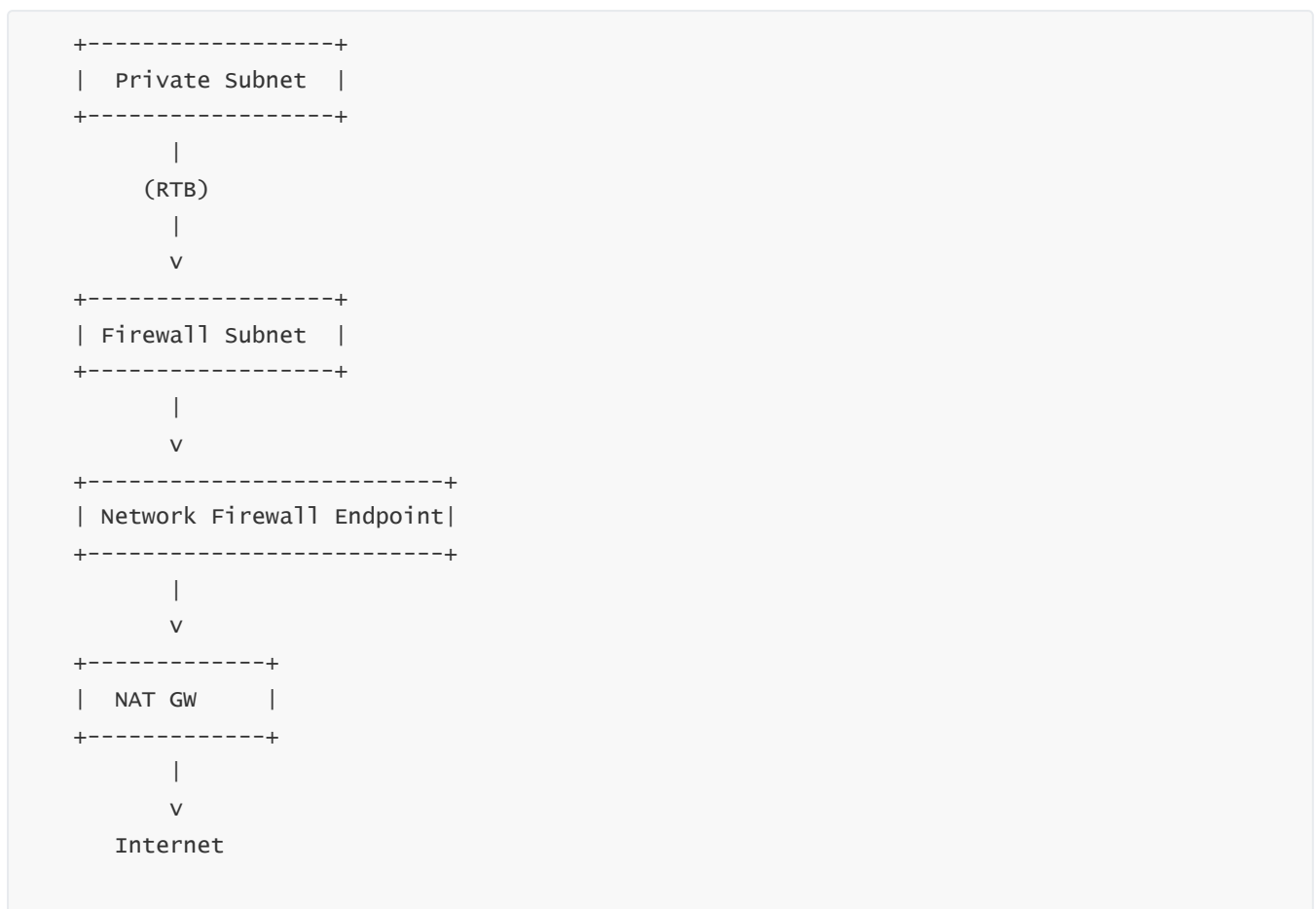
In outbound inspection, private subnets route their egress traffic to the firewall subnets first. The firewall can then enforce rules such as:

- blocking access to malicious domains
- restricting outbound ports
- preventing data exfiltration
- enforcing DNS resolution policies
- applying strict IPS signatures on outbound flows

–

After inspection, the firewall forwards traffic to NAT gateways or egress internet gateways, from where it reaches external destinations.

### Outbound Flow Diagram



### Explanation:

Outbound traffic is fully inspected before hitting the NAT gateway. This allows organizations to enforce domain filtering, stop malware callbacks, and detect compromised workloads before data leaves the organization.

## 4 — Handling East-West Traffic Within a VPC (Microservice Lateral Movement)

East-west traffic refers to communication within a VPC—between private subnets, between AZs, or between microservices. Traditional firewalls often ignore east-west flows, creating blind spots where lateral movement attacks thrive.

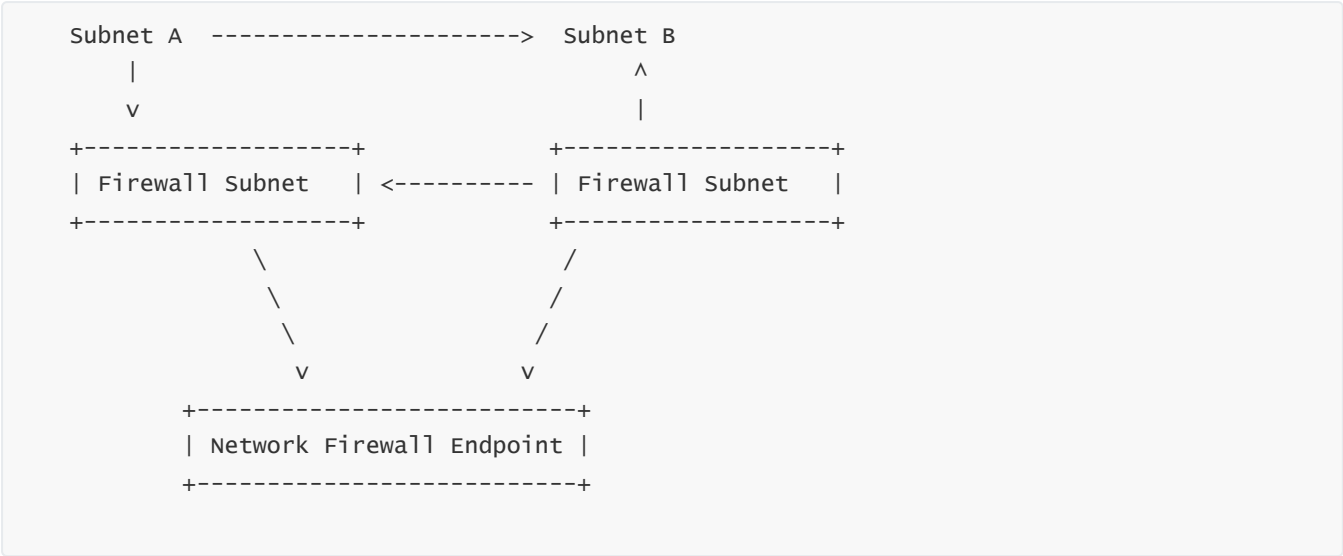
–

AWS Network Firewall can inspect east-west flows by steering internal subnet-to-subnet traffic through firewall subnets. This involves careful routing: instead of sending traffic directly from subnet A to subnet B, the route table sends it through the firewall first.

-

The stateful engine analyzes whether the lateral movement matches abnormal behavior: unexpected SMB traffic, unauthorized SSH attempts, or unusual port scans within the VPC. Enterprises use this functionality to secure internal workloads and prevent lateral spread of threats.

**East-West Flow Diagram (Within VPC)**



**Explanation:**

Both directions of east-west traffic pass through the firewall to ensure full symmetry and allow accurate state tracking.

**5 — Handling East-West Traffic Between VPCs (Using Transit Gateway)**

When multiple VPCs communicate through a Transit Gateway, all inter-VPC traffic can be routed through a centralized inspection VPC that contains Network Firewall.

-

TGW route tables determine which attachments forward traffic into the inspection VPC. After inspection, the firewall returns the traffic to the Transit Gateway, which routes it to the target VPC.

-

This model is a large-scale enterprise pattern: a single firewall cluster protects hundreds of VPCs using a single inspection hub.

**East-West Inter-VPC Diagram (With TGW)**



**Explanation:**



The firewall becomes a centralized choke point for all inter-VPC communication, ensuring consistent lateral traffic protection across the enterprise.

### 6 — Symmetric Traffic Requirement and How AWS Enforces It

Because stateful inspection requires flow context, **traffic must enter and exit through the same firewall endpoint** for each flow. AWS ensures symmetry automatically by:

- using deterministic flow hashing
- ensuring return routes match forward routes
- distributing flows to the correct firewall endpoint
- maintaining per-AZ endpoint symmetry
- replicating flow state in local node memory
- avoiding any cross-AZ state transfer to prevent latency
- 

If routing is misconfigured (asymmetric routing), the firewall drops packets because the second endpoint does not have the flow state. Enterprises must design routing carefully to guarantee symmetric paths for all inspected traffic flows.

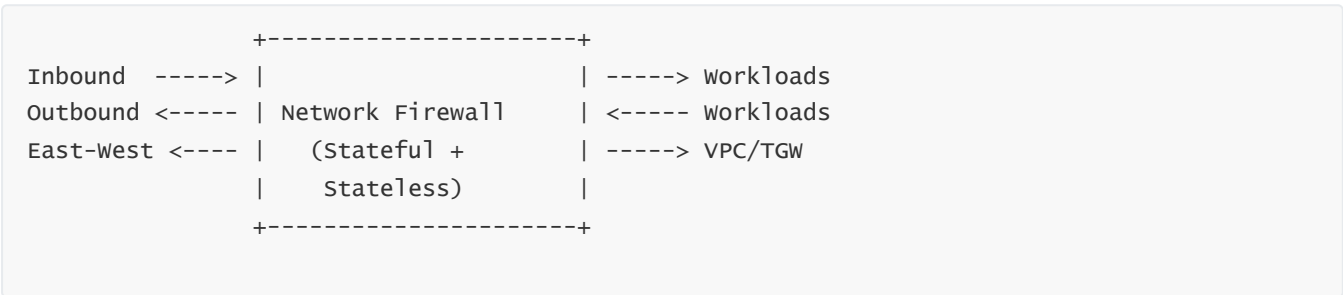
### 7 — How Policies Differ for Each Traffic Direction

Enterprises often create separate rule groups for each traffic direction:

- inbound policies focus on allowed services, malicious payload detection, and external threat protection
- outbound policies focus on domain filtering, egress restrictions, DLP controls, and malware callbacks
- east-west policies focus on segmentation, lateral movement protection, microservice governance, and internal attack detection
- 

Policies attach to one firewall, but rule groups allow separation of concerns for clarity and governance.

### 8 — End-to-End Combined Traffic Flow Model



**Explanation:**

Inbound, outbound, and east-west flows all enter the same distributed firewall fabric and undergo stateless → stateful → final decision processing.

---

## 8 — How network routing and traffic steering works with AWS Network Firewall

---

### 1 — Why Routing Design Is the Foundation of Network Firewall Architecture

Unlike traditional hardware firewalls placed physically inline, AWS Network Firewall becomes part of your network **only when routing tables direct traffic to it**. The firewall does not “see” packets automatically — it inspects only the flows that your VPC or Transit Gateway routing explicitly forwards into the firewall subnets.

–

This makes routing the single most important part of the architecture. The firewall endpoints sit inside special **firewall subnets**, and all traffic must be routed from application subnets to those firewall subnets before reaching external destinations or internal peer networks.

–

Routing, therefore, determines whether packets are inspected, bypassed, or dropped due to asymmetric paths. Good routing design ensures symmetric traffic, correct next hops, and predictable flow handling across stateless and stateful engines.

---

### 2 — The Three Routing Models Used With AWS Network Firewall

AWS provides three supported methods to steer traffic into the firewall:

- **VPC route table steering** (standard inspection inside a single VPC)
- **Transit Gateway inline routing** (centralized inspection for multi-VPC architectures)
- **Hybrid routing for on-prem traffic** (VPN/DX routing through firewall subnets)

The firewall is always deployed in **firewall subnets** in each AZ. Routing is then configured so that all desired traffic flows (inbound/outbound/east-west) must traverse those firewall subnets.

---

### 3 — How Routing Works Inside a Single VPC

Inside a VPC, the routing design is straightforward. You configure the **private subnet route table** to point all egress or east-west traffic to the firewall endpoint by setting the firewall subnet as the next hop.

–

Once traffic enters the firewall endpoint, the firewall evaluates it and forwards it to the appropriate destination: IGW, NAT GW, TGW, or another VPC.

–

The **firewall subnet route table** has a return route that directs inspected traffic to NAT/IGW/TGW depending on the architecture.

---

–

This ensures symmetrical flow: outgoing packets go from private subnet → firewall → NAT gateway → internet, and return packets follow NAT → firewall → private subnet.

Single VPC Routing Diagram



Explanation:

The private subnet sends all traffic to the firewall subnet. After inspection, the firewall forwards the packet to NAT/IGW/TGW. The reverse path follows the same route, preserving symmetry for stateful inspection.

4 — Traffic Steering for Inbound Traffic (IGW → Firewall → Private Subnet)

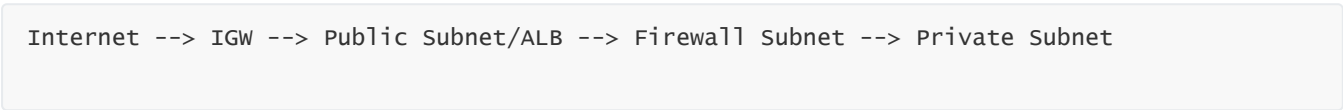
When inbound inspection is required, the default behavior of IGW (which can only route to public subnets) must be modified using load balancers or proxy layers. Typically, inbound flows use this pattern:

IGW → Public Subnet/ALB → Firewall Subnet → Private Subnet

–

Public-facing ALBs or NLBs receive traffic first. Their target group then routes traffic into firewall subnets (via ENI-level routing), ensuring that inbound flows are fully inspected before reaching workloads.

Inbound Steering Diagram



Explanation:

ALB acts as an entry point into the VPC. Its private ENIs allow routing to firewall subnets where DPI occurs.

5 — Routing With Transit Gateway (Centralized Inspection VPC)

For multi-VPC networks, Transit Gateway becomes the central routing control-plane. When using TGW inline inspection, traffic flows:

**App VPC → TGW → Inspection VPC → Network Firewall → TGW → Destination (Internet or another VPC)**

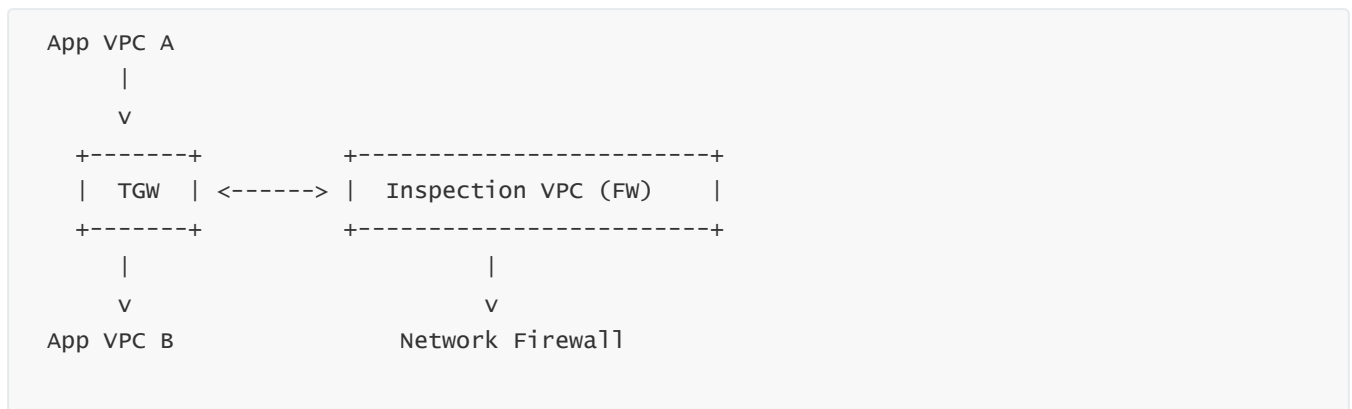
-

This creates a fully centralized inspection hub where a single firewall cluster protects hundreds of VPCs.

-

TGW route tables control which attachments forward their traffic to the inspection VPC. The inspection VPC receives the traffic, passes it through firewall endpoints, and returns it to TGW for final delivery.

### Transit Gateway Routing Diagram



### Explanation:

TGW attachments use separate route tables for fine-grained control. The inspection VPC is simply another TGW attachment where firewall subnets exist.

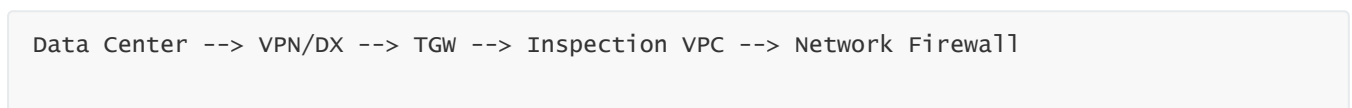
## 6 — Hybrid Routing (On-Prem → VPN/DX → TGW → Firewall)

Hybrid networks involve traffic from on-prem data centers. Routing directs on-prem flows into TGW, and from TGW into the inspection VPC firewall subnets.

-

This allows enterprises to enforce cloud-level security on hybrid connectivity, ensuring that even private MPLS or on-prem datacenter traffic must pass through firewall inspection.

### Hybrid Routing Diagram



### Explanation:

This model protects against east-west and hybrid-security risks, such as compromised servers or lateral spread of malware between datacenters and cloud workloads.

## 7 — The Symmetric Routing Requirement

Stateful firewalls require symmetrical flow:

**Forward direction and return direction must travel through the same firewall endpoint.**

–

AWS ensures symmetry using:

- deterministic flow hashing
- AZ-local endpoints
- consistent route tables
- separate route tables per subnet
- static or TGW policies

–

If asymmetric routing occurs (packet forward path differs from return path), the firewall drops the packet because the second endpoint lacks the flow state.

### Symmetric Routing Diagram

```
Forward Path: Private Subnet --> Firewall --> NAT
Return Path:  NAT --> Firewall --> Private Subnet
```

#### Explanation:

Both directions pass through the same firewall. If return traffic bypasses the firewall, the flow breaks.

---

## 8 — How to Prevent Routing Loops and Bypass Paths

Enterprises must carefully avoid:

- Firewall → TGW → Firewall loops
- Private Subnet → Firewall → Private Subnet recursion
- TGW propagating routes incorrectly
- Route tables accidentally allowing direct IGW/NAT bypass

–

AWS recommends using **dedicated firewall subnets** in each AZ and **strict RTB separation** between private, firewall, inspection, and public subnets.

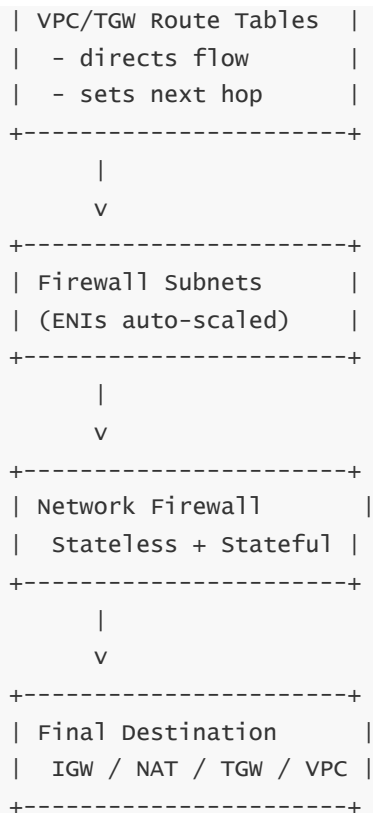
–

The firewall should never be placed directly in public subnets; it must always live in dedicated firewall subnets.

---

## 9 — End-to-End Routing Decision Diagram

```
Incoming Packet
  |
  v
+-----+
```



#### Explanation:

The routing plane controls entry into the firewall. The firewall controls packet inspection. The routing plane then controls final delivery.

## 9 — How AWS Network Firewall integrates with AWS Transit Gateway (TGW)

### 1 — Why Transit Gateway Integration Matters for Enterprise Security Architecture

In modern AWS organizations, applications do not live in a single VPC. Large enterprises may have **dozens, hundreds, or even thousands of VPCs** distributed across multiple accounts and Regions. Without a scalable hub, every VPC would require its own firewall deployment, leading to operational overhead, inconsistency, cost explosion, and policy drift.

–

**AWS Transit Gateway (TGW)** solves the network connectivity problem by serving as a **central routing hub**. When AWS Network Firewall is placed inline with TGW, the firewall becomes a **central inspection point** for **all north-south and east-west traffic** flowing between:

- VPCs
- On-prem datacenters via VPN/DX
- The internet via egress VPCs
- Multi-Region networks via inter-region peering

–

This turns a single Inspection VPC into an enterprise-level, cloud-native security perimeter. TGW inline inspection is the **most scalable** and **most widely used** architecture for AWS Network Firewall at enterprise scale.

---

## 2 — The Basic Integration Model: TGW → Inspection VPC → Firewall → TGW

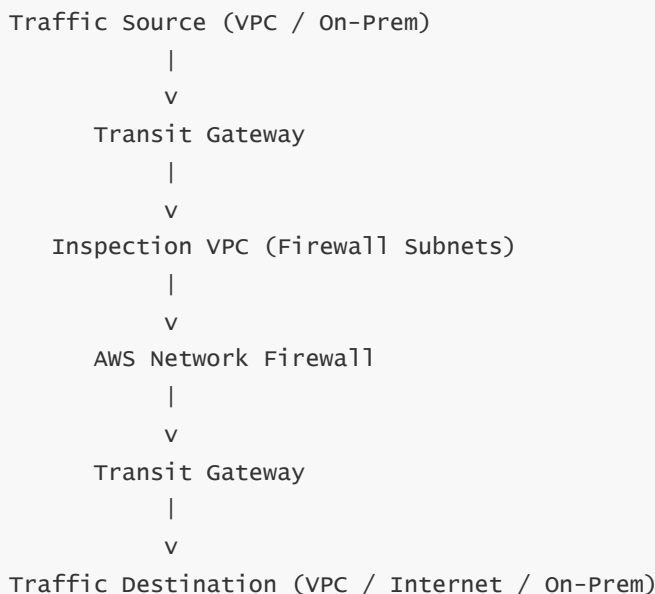
TGW does not embed the firewall internally. Instead, TGW steers traffic to an **Inspection VPC** that contains the firewall subnets. TGW then expects the firewall to return the inspected traffic back into TGW before forwarding it to its final destination.

–

This concept is called **inline inspection** or **service insertion**.

–

The high-level flow is:



### Explanation:

Traffic leaves the source VPC, enters TGW, is forwarded to the Inspection VPC, hits the firewall, returns to TGW, and finally reaches the destination. The firewall becomes a mandatory checkpoint for every cross-VPC, outbound, or inbound flow.

---

## 3 — The Route Table Structure Required for TGW Integration

TGW integration requires **three types of route tables**:

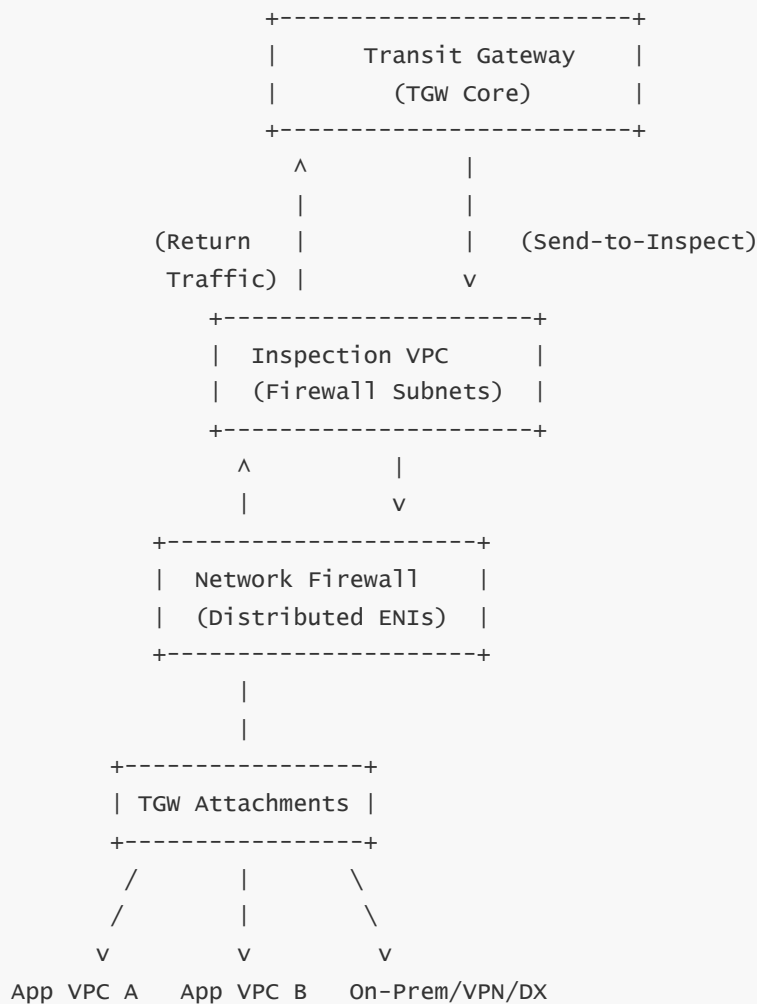
1. **Attachment Route Table (for traffic sources like VPCs)**
  - Directs all traffic to the Inspection VPC attachment
2. **Inspection VPC Attachment Route Table (receives traffic)**
  - Steers the incoming packet into firewall subnets

### 3. Firewall Return Route Table (inside Inspection VPC)

- Returns inspected packets back to TGW for onward delivery

Each route table must be configured symmetrical to maintain stateful flow handling.

## 4 — Detailed Architecture Diagram



### Explanation:

TGW acts as the brain of the network. The Inspection VPC hosts firewall subnets in each AZ. All attachments (VPCs, VPN, DX) route traffic into the inspection path.

## 5 — Step-by-Step Flow for Inter-VPC Communication Through TGW Firewall

### Example: VPC A → VPC B

Step 1: VPC A sends traffic to TGW

- VPC A's subnet route table sends traffic (e.g., 10.20.0.0/16) to its TGW attachment.

Step 2: TGW receives traffic and forwards to Inspection VPC

- TGW attachment route table says: "For any VPC-to-VPC traffic, send to the Inspection VPC."



Step 3: Inspection VPC receives traffic

- The Inspection VPC's ingress route table directs the packet to firewall subnets.

Step 4: Firewall inspects traffic

- Stateless rules evaluate first
- Stateful Suricata engine evaluates flow context
- Final allow/drop/alert action is applied

Step 5: Firewall returns traffic to TGW

- Firewall endpoint sends packet back to TGW attachment via return route table.

Step 6: TGW delivers packet to VPC B

- TGW attachment route table forwards traffic to destination VPC attachment.

### Full Flow Diagram

```
VPC A --> TGW --> Inspection VPC --> Network Firewall --> TGW --> VPC B
```

### Explanation:

Both forward and return flows pass through the firewall to maintain symmetric traffic requirements.

---

## 6 — Step-by-Step Flow for Outbound Internet Traffic Through TGW Firewall

### Example: VPC A → Internet

Step 1: VPC A sends outbound traffic to TGW

Step 2: TGW forwards traffic to Inspection VPC

Step 3: Firewall inspects the traffic

Step 4: Firewall forwards to NAT Gateway in the Egress VPC

Step 5: NAT Gateway → Internet

Step 6: Return traffic comes back along the exact reverse route

### Outbound Flow Diagram (TGW + Egress VPC)

```
VPC A --> TGW --> Inspection VPC --> Firewall --> Egress VPC (NAT) --> Internet
```

## 7 — The Requirement for AZ-Alignment Between TGW and Firewall Endpoints

TGW enforces AZ-locality when sending traffic to VPC attachments. This means TGW chooses the attachment's ENI that corresponds to the same AZ as the source of the traffic.

–

Therefore, the Inspection VPC must have firewall subnets in *every AZ where TGW might send traffic*.

-

If TGW sends traffic to an AZ where no firewall subnet exists, traffic is dropped.

This is a critical design principle.

### 8 — Preventing Asymmetric Routing in TGW + Firewall Designs

Asymmetric routing is the primary operational risk. It occurs when:

- Forward path goes TGW → FW → TGW
- Return path goes TGW → VPC directly, bypassing firewall

The firewall drops the return packet because its endpoint does not have the connection’s state.

-

To prevent this, all TGW attachments must use route tables that force both forward and return flows through the Inspection VPC.

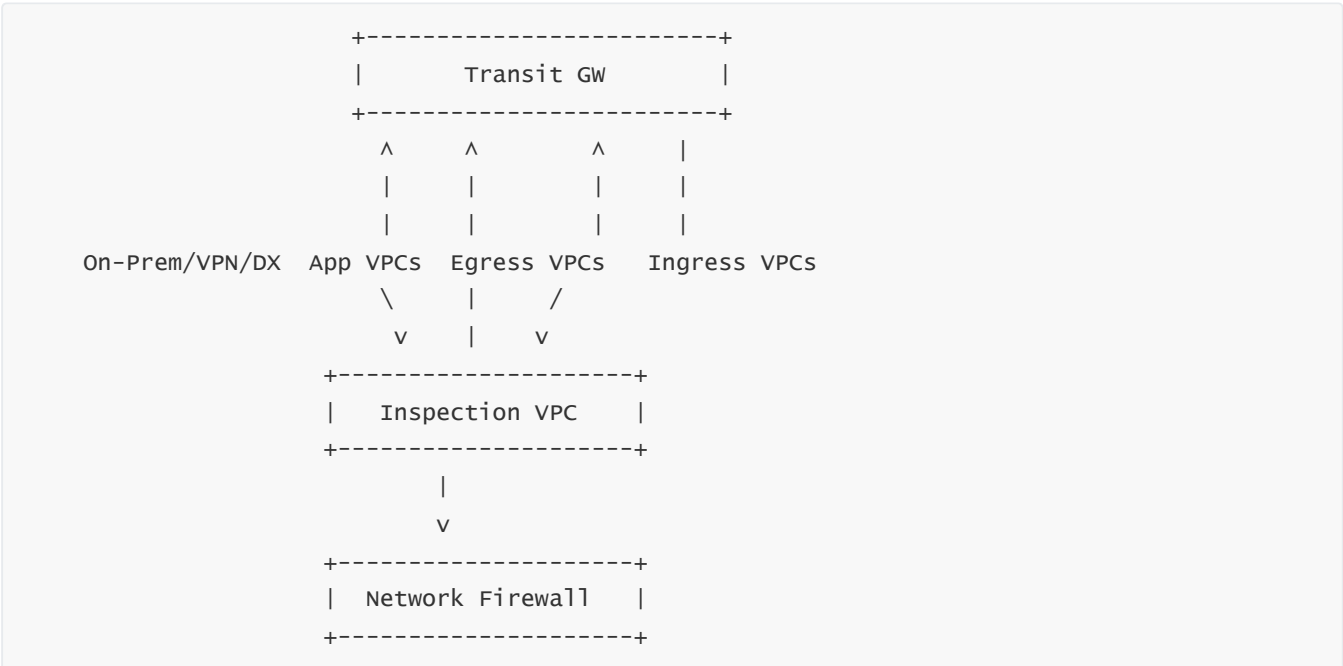
#### Asymmetric Routing Drop Example

```
Forward Flow: VPC A ---> TGW ---> Firewall ---> TGW ---> VPC B
Return Flow:  VPC B ---> TGW -----> VPC A    (WRONG) ❌
```

#### Correct Symmetric Path

```
Return Flow: VPC B ---> TGW ---> Firewall ---> TGW ---> VPC A (Correct) ✔
```

### 9 — End-to-End TGW Inline Inspection Flow (All Traffic Directions)



## Explanation:

All inbound, outbound, and east-west transit flows across TGW are inspected centrally.

---

# 10 — How scaling, performance, and high availability work internally in AWS Network Firewall

---

## 1 — The Cloud-Native Scaling Philosophy Behind AWS Network Firewall

AWS Network Firewall is not a single appliance, not a cluster you manage, and not a pair of HA instances. It is a **distributed, horizontally auto-scaling fabric** deployed across your VPC, where AWS automatically provisions multiple firewall endpoint nodes inside each firewall subnet.

–

This design gives the firewall near-infinite horizontal scaling capacity because AWS can add new nodes whenever traffic increases. The firewall's performance is not tied to instance size or pair-based clustering. Instead, the underlying infrastructure resembles an elastic pool of high-performance data-plane processors that inspect flows in parallel.

–

Enterprises do not handle failover, cluster orchestration, disk replacements, throughput tuning, or appliance patching. AWS manages everything automatically.

---

## 2 — The Internal Scaling Model: “Firewall Endpoints” in Each AZ

For each AZ where the firewall is deployed, AWS automatically creates **multiple stateless + stateful endpoint nodes**, each represented as an ENI in your firewall subnet. These nodes operate independently and handle separate traffic flows.

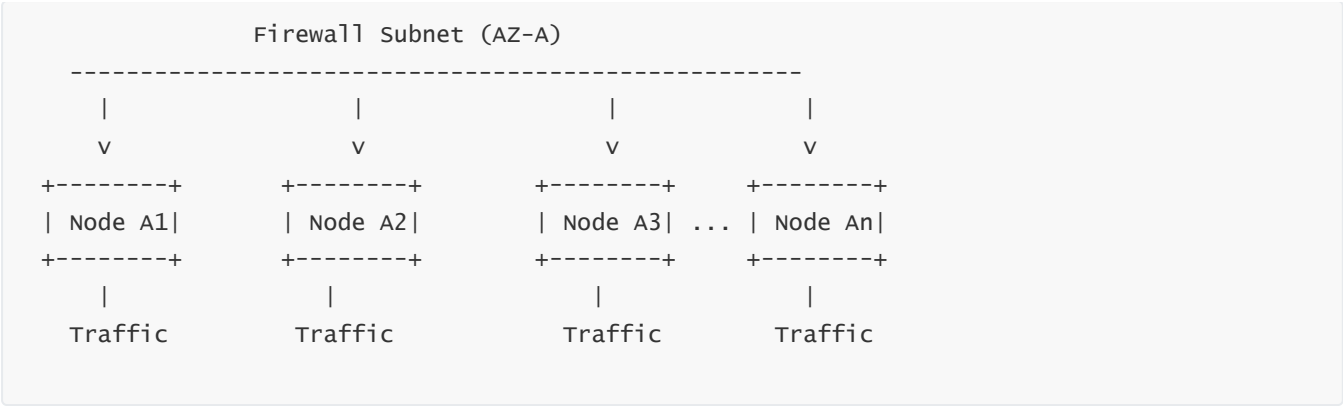
–

When traffic increases, AWS silently adds more nodes. When load decreases, AWS scales the pool down. If a node becomes unhealthy, AWS replaces it automatically.

–

The scaling is granular and fully automated—no alarms, no scaling groups, no warm pools, no customer operations.

## Diagram — Scaling Node Structure in One AZ



**Explanation:**

Each node inspects different flows. AWS expands or contracts this node pool dynamically to meet throughput needs.

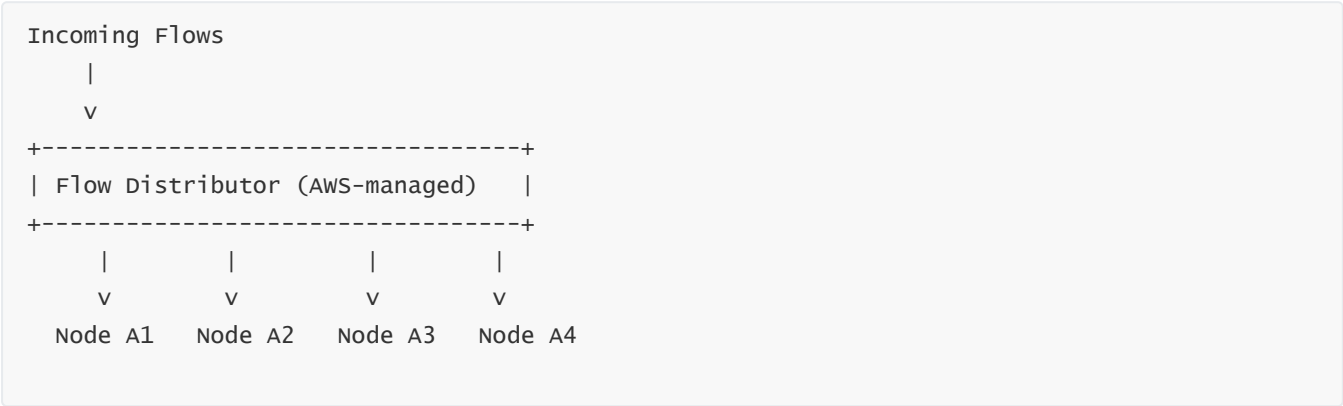
**3 — Flow Distribution and Load Balancing Across Nodes**

When a packet arrives at the firewall, AWS applies deterministic hashing on flow attributes (5-tuple: source/destination IP, source/destination port, protocol).

This hash maps the flow to a specific node. The mapping remains consistent throughout the flow's lifetime, ensuring that stateful inspection has the correct flow context.

AWS automatically redistributes flows when nodes scale in or out. New flows may be assigned to new nodes, while existing flows remain pinned to their original nodes.

**Flow Distribution Diagram**



**Explanation:**

The flow distributor sends flows to nodes deterministically. This avoids cross-node flow confusion and preserves stateful consistency.

**4 — High Availability Through AZ-Distributed Endpoint Clusters**

AWS Network Firewall is inherently multi-AZ. Each AZ where the firewall is deployed receives its own **independent node cluster**.

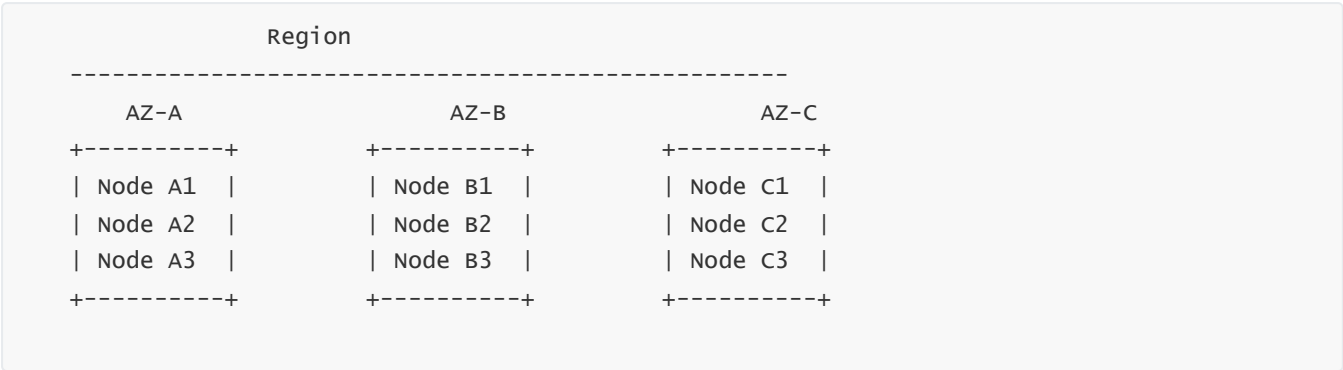
-

If AZ-A fails, traffic in AZ-B is unaffected. If a node in AZ-A fails, only flows handled by that node fail; AWS immediately replaces it.

-

Because AWS drives endpoint health monitoring and node replacement autonomously, firewalls maintain extremely high availability without customer-designed HA pairs.

**HA Diagram Across AZs**



**Explanation:**

Each AZ has its own node cluster. No cross-AZ node dependency exists, ensuring that regional failures do not break stateful flow tracking.

**5 — Stateless vs. Stateful Scaling Behavior**

Both stateless and stateful engines scale automatically, but their behavior differs:

- **Stateless scaling:** Focuses on packet-per-second load. Stateless rules are CPU-light, so scaling is triggered mainly by extremely high traffic.
- **Stateful scaling:** Focuses on flows. Stateful rules require tracking connection data and DPI, so scaling is based on flow count, memory usage, and CPU load inside the Suricata engine.

AWS monitors node performance and metrics internally (not visible to customers) and triggers scaling events in milliseconds.

**6 — Performance Characteristics of the Stateful Suricata Engine**

The stateful engine performs deep packet inspection (DPI), domain filtering, protocol decoding, and signature matching.

-

This engine is CPU- and memory-intensive, especially for encrypted traffic (where only TLS metadata is inspected). As flows grow, AWS adds more nodes to distribute load.

-

Suricata rule complexity also affects performance. Large numbers of custom IPS rules may increase inspection time. However, AWS's horizontally distributed model ensures that no single node becomes a bottleneck under typical enterprise loads.

---

## 7 — Return Traffic and Performance Symmetry

Return traffic must follow the same path as forward traffic to maintain flow state.

–

As long as routing is symmetrical, performance remains consistent because:

- forward packets and return packets hit the same AZ
- the same flow hash routes both to the same node
- TCP window scaling, sequence tracking, and state tables remain consistent

If customers accidentally misconfigure routing (e.g., return path bypasses firewall), performance degrades and packets will be dropped.

---

## 8 — Firewall Throughput and Scaling Limits

AWS does not publish specific throughput numbers, because they vary based on:

- rule complexity
- Suricata rule groups
- packet size distribution
- percentage of packets forwarded to stateful inspection
- directional flow distribution

–

However, the firewall is designed to scale to handle **tens of gigabits per second** and **millions of concurrent connections** without customer intervention.

–

The distributed architecture removes the need for throughput calculators or appliance sizing.

---

## 9 — Failure Scenarios and Auto-Healing Behavior

AWS Network Firewall automatically handles:

- node process failure
- ENI endpoint failure
- OS-level issues
- Suricata engine crashes
- hypervisor-level node failures

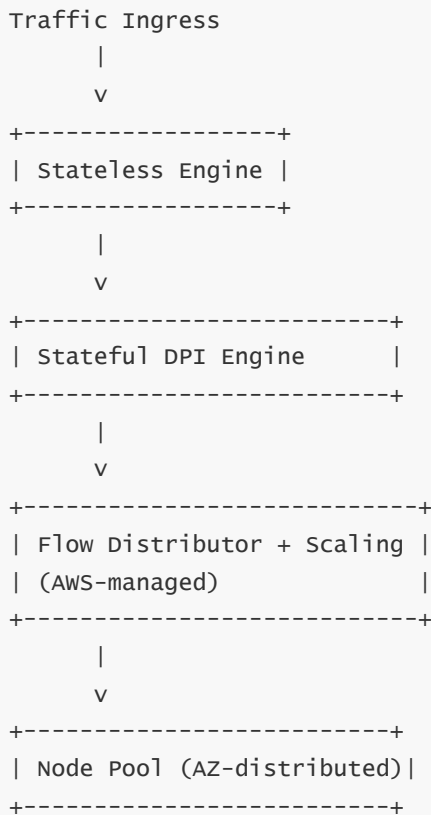
- AZ-level partial failures

–

When a node fails, flows handled by that node are reset (as expected with any IPS). New flows are re-hashed to new nodes. AWS replaces the node instantly and restores full capacity.

---

## 10 — End-to-End Performance and Scaling Pipeline



### Explanation:

Stateless and stateful engines perform inspection. The flow distributor assigns flows to nodes. AWS scaling logic ensures sufficient nodes exist to handle current traffic. Nodes continue processing until flows end or nodes are replaced.

---

# 11 — How to design an enterprise-grade security architecture with AWS Network Firewall

## 1 — The Purpose of an Enterprise-Grade Firewall Architecture

An enterprise-grade security architecture must deliver more than simple packet filtering. It must enforce **multi-layer defense, centralized governance, deep inspection, east-west segmentation, north-south control**, and **organization-wide consistency** across many VPCs, accounts, and environments. AWS Network Firewall provides these capabilities only when deployed with the correct architectural structure — a centralized

inspection model, TGW routing domains, VPC separation, strict rule governance, and audit/compliance pipelines.

-

In small environments, a firewall inside a single VPC is enough. But at enterprise scale, you must build a dedicated Inspection VPC, use Transit Gateway for traffic aggregation, enforce AWS Organizations-based governance, and route all inter-VPC and internet-bound flows through the firewall. This centralization eliminates security drift, prevents bypasses, and ensures every packet takes the correct inspection path.

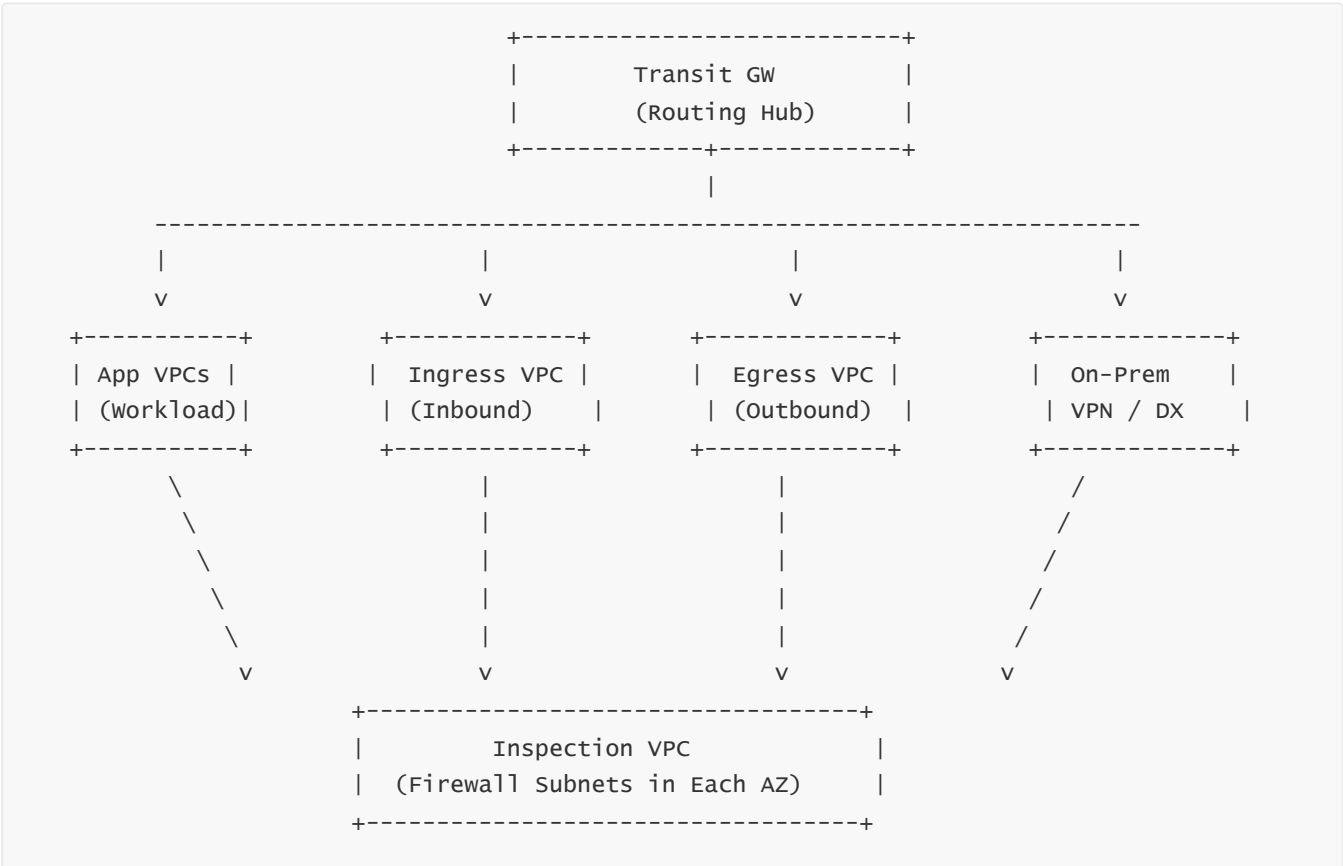
## 2 — The Core Building Blocks of an Enterprise Design

A complete enterprise architecture includes the following mandatory components:

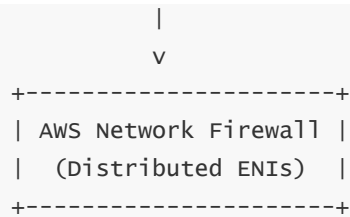
- 1. **Inspection VPC** — A dedicated VPC hosting firewall subnets in every AZ.
- 2. **Transit Gateway (TGW)** — Aggregates all VPCs, on-prem networks, ingress VPCs, and egress VPCs.
- 3. **Firewall Subnets** — One per AZ, hosting the distributed firewall endpoints.
- 4. **Outbound (Egress) VPC** — NAT gateways and internet routing for outbound traffic.
- 5. **Inbound (Ingress) VPC** — ALB/NLB + firewall inspection for inbound flows.
- 6. **Logging + Analytics Layer** — S3, Kinesis Firehose, CloudWatch Logs, SIEM integrations.
- 7. **Firewall Manager Governance** — Enforces rule groups and prevents drift across accounts.

These components create a **hub-and-spoke** security architecture that scales to hundreds of VPCs while preserving consistent enforcement.

## 3 — High-Level Enterprise Architecture Diagram







#### Explanation:

All VPCs attach to TGW. TGW forwards traffic into the Inspection VPC. The firewall performs stateless + stateful inspection, then traffic is returned to TGW for delivery to its destination.

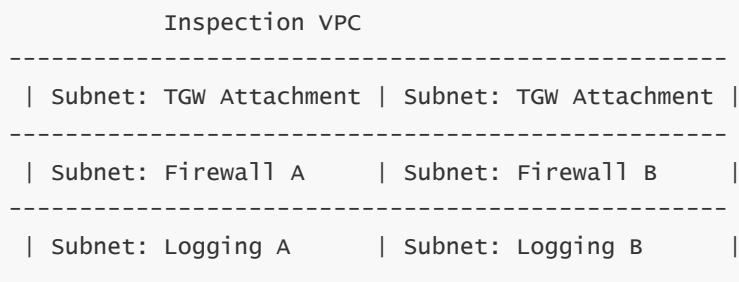
### 4 — Designing the Inspection VPC (Critical Enterprise Component)

The Inspection VPC must include:

- **Firewall subnets in every AZ** used by TGW.
- **Separate route tables** for firewall subnets, private subnets, and attachment subnets.
- **No public subnets** containing firewall endpoints (firewalls must never sit in public subnets).
- **Dedicated logging subnets** if sending logs through Kinesis/Firehose inside the VPC.

The firewall subnets act as the **mandatory chokepoint**. All inbound, outbound, and east-west traffic flows through them.

#### Inspection VPC Internal Layout Diagram



#### Explanation:

This layout isolates firewall operations, simplifies routing, and ensures predictable multi-AZ operation.

### 5 — Designing the Outbound (Egress) Architecture

For outbound internet access, enterprise-grade design requires:

- **Private subnets in App VPCs** → **TGW** → **Firewall** → **Egress VPC**
- NAT Gateway placement **only inside the Egress VPC**
- Strict stateless deny lists for ports, IPs, and protocols
- Stateful DNS/domain filtering to block malicious destinations

- Deep packet inspection for outbound traffic (Suricata IPS rules)

### Outbound Flow Diagram

```
App VPC → TGW → Inspection VPC → Network Firewall → Egress VPC (NAT) → Internet
```

#### Explanation:

This model centralizes all outbound controls, preventing workloads from bypassing the firewall.

---

## 6 — Designing the Inbound Architecture (Enterprise Grade)

Inbound traffic must enter via a dedicated Ingress VPC, not directly into application VPCs. The pattern is:

- Internet → ALB/NLB → TGW → Inspection VPC → Firewall → TGW → Application VPC
- Stateless rules block unwanted ports
- Stateful rules detect malicious payload signatures
- Protocol-aware logic (HTTP, DNS, TLS metadata) executed by Suricata
- Logging delivered to CloudWatch/S3 for SIEM correlation

### Inbound Flow Diagram

```
Internet → Ingress VPC (ALB/NLB) → TGW → Firewall → TGW → App VPC
```

#### Explanation:

This architecture eliminates the “open public subnet” anti-pattern and places inbound traffic behind multiple protective layers.

---

## 7 — Designing East-West Microsegmentation

Enterprise segmentation requires preventing lateral movement inside AWS.

Network Firewall implements segmentation at scale by:

- Establishing **per-VPC routing domains** via TGW
- Sending VPC-to-VPC traffic through the firewall
- Writing stateful rules for:
  - port restrictions
  - protocol compliance
  - internal threat signatures
  - domain restrictions
- Using DNS filtering for internal service security

## East-West Flow Diagram

```
VPC A → TGW → Firewall → TGW → VPC B
```

### Explanation:

No VPC communicates directly with another VPC. All flows are centrally inspected.

---

## 8 — Governance Using AWS Firewall Manager

Firewall Manager enforces consistent rules across hundreds of accounts. It allows:

- Mandatory stateless deny rules at the top of all policies
- Mandatory stateful domain filtering profiles
- Automatic remediation of drifted or missing firewalls
- Centralized deployment of firewall resources
- Audit visibility via AWS Organizations

Enterprises typically:

- Maintain a “Corporate Mandatory Policy”
- Add department-specific optional rules
- Use automation to force on-boarding of any new VPC

This ensures no environment bypasses inspection.

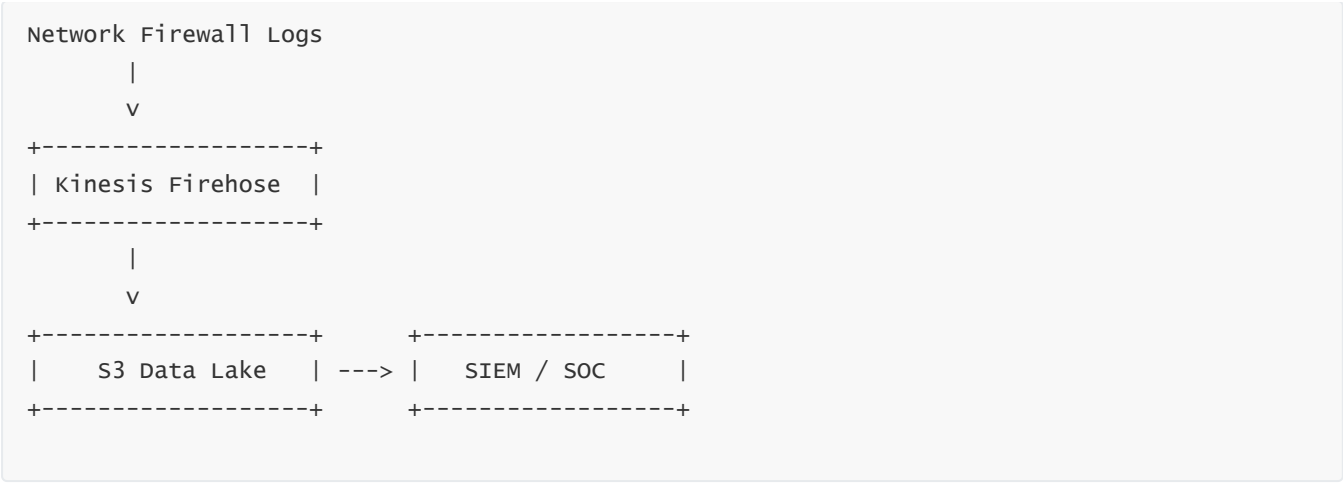
---

## 9 — Enterprise Logging and Threat Analytics Architecture

Logs must feed into a centralized analytics pipeline. Enterprise design uses:

- Flow logs → S3 (lake storage)
- Alert logs → Kinesis Firehose → SIEM/Splunk/ELK
- Stateful Suricata alerts → CloudWatch Logs
- Threat correlation via GuardDuty or custom engines

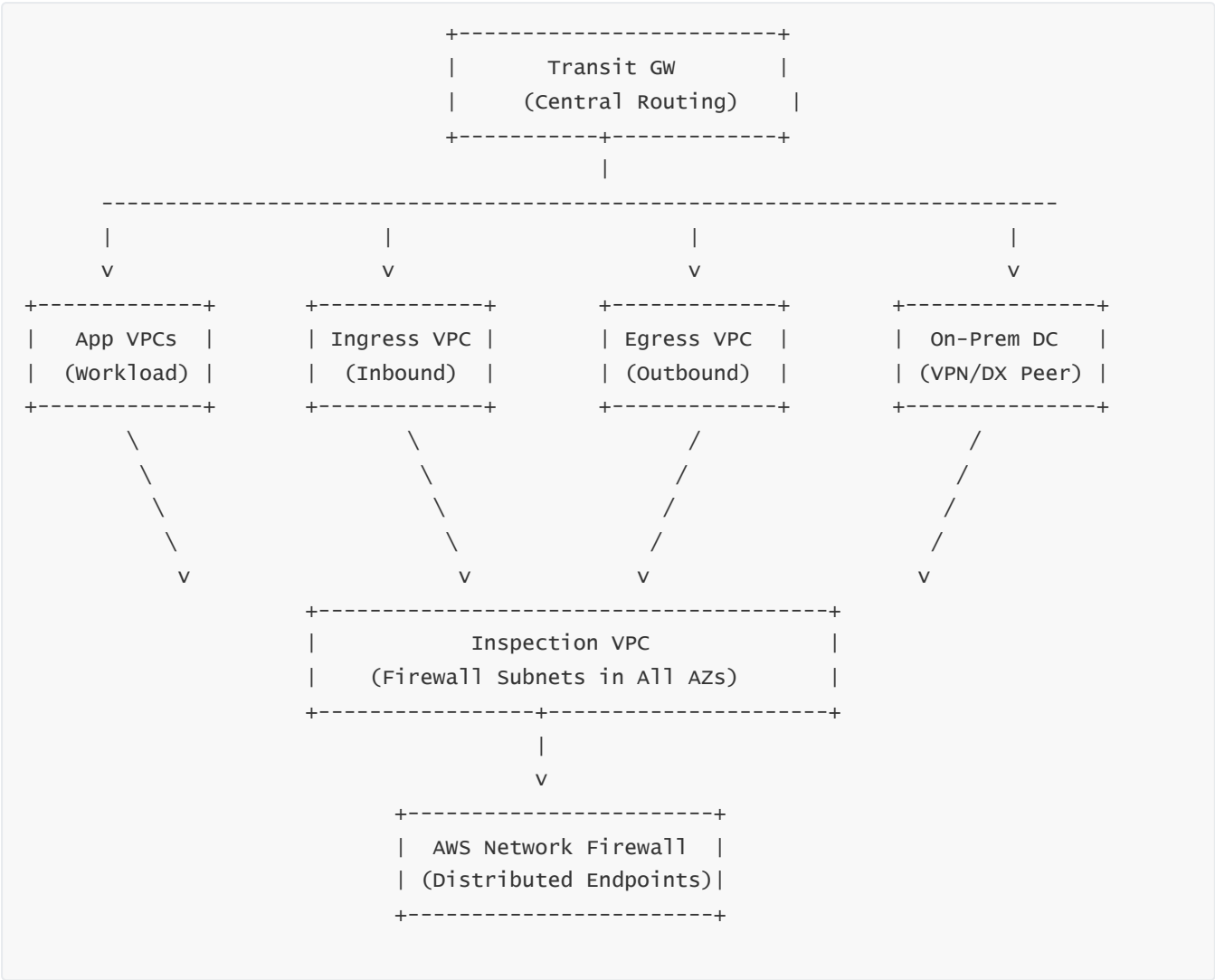
### Logging Pipeline Diagram



**Explanation:**

This ensures complete visibility for threat hunting and compliance audits.

**10 — Full Enterprise-Grade AWS Network Firewall Architecture (Final Integrated View)**



**Explanation:**

This architecture centralizes all inbound, outbound, and inter-VPC traffic into a single security perimeter, enforces Suricata-powered deep inspection, and uses TGW to scale across the entire organization.

---

# 12 — How to implement deep packet inspection and advanced threat detection in AWS Network Firewall

---

## 1 — The Purpose of Deep Packet Inspection (DPI) in Enterprise Security

Deep Packet Inspection (DPI) enables security teams to analyze not just the headers of network packets but also their **content, behavior, metadata, and protocol structure**. This moves security beyond simple allow/deny rules and into full contextual threat detection: ransomware callbacks, C2 (command-and-control) activity, malicious DNS queries, suspicious HTTP headers, anomalous TLS handshakes, and protocol misuse.

–

AWS Network Firewall performs DPI using a **Suricata-based stateful engine**, which is one of the most widely used IDS/IPS engines globally. Suricata brings enterprise-grade capabilities such as signature matching, application-protocol decoding, threat intelligence list enforcement, and connection-state validation.

–

Enterprises rely on this DPI layer to detect sophisticated threats that bypass stateless filters, evade SG/NACL controls, or hide within permitted ports like HTTPS, DNS, or SSH. DPI ensures that even allowed connections are evaluated for correctness, behavior, and malicious intent.

---

## 2 — How Deep Packet Inspection Works Internally (The Suricata Engine Pipeline)

When traffic is forwarded to the stateful engine by the stateless rules, the packet enters a multi-stage DPI pipeline:

### 1. Flow Identification & Connection Tracking

- Creates/updates flow tables
- Validates SYN/SYN-ACK/ACK correctness
- Detects invalid TCP flags, sequence anomalies, and fragmentation issues

### 2. Protocol Decoding

- Interprets the packet based on its protocol: DNS, HTTP metadata, SMB, TLS handshake, SSH banner, SIP, FTP, etc.
- Applies protocol-specific detection logic

### 3. Signature Matching (Rule Evaluation)

- Compares packet content and metadata to Suricata IPS signatures
- Signature actions: drop, alert, allow, log

### 4. Domain Filtering (DNS-Aware Inspection)

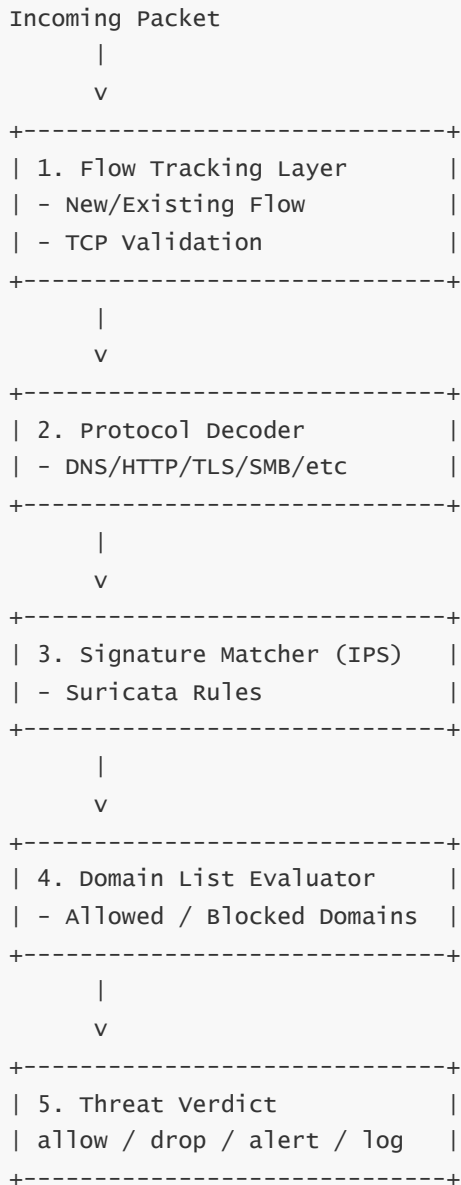
- Matches DNS queries/responses against allowed/blocked domain lists
- Blocks C2 domains, malicious top-level domains, phishing domains

## 5. Threat & Behavior Correlation

- Detects port scanning, slow-loris attacks, protocol abuse
- Identifies tunneling attempts, exfiltration patterns

The DPI engine then issues a verdict (allow/drop/alert) and optionally logs the event.

### DPI Pipeline Diagram



### Explanation:

This layered model ensures full packet, flow, and protocol-aware inspection, allowing enterprises to detect sophisticated threats hidden behind otherwise normal traffic.

## 3 — Suricata IPS Rule Capabilities (Advanced Threat Detection)

Suricata rules are extremely powerful. They can inspect:

- IP/port combinations

- TCP flags, sequences, anomalies
- HTTP headers, hostnames, user-agents
- DNS queries and responses
- TLS fingerprints (SNI, issuer, certificate metadata)
- File signatures and payload patterns (up to DPI limit)
- Known malware callbacks and beaconing patterns
- SSH banners, FTP/SIP/SMTP patterns
- SMB protocol structures

Suricata supports **hundreds of rule keywords**, enabling detection of protocol misuse, malware signatures, suspicious metadata, and attack patterns.

Examples of threat categories DPI detects:

- C2 traffic from malware
- Data exfiltration via DNS tunneling
- Port scanning and lateral movement
- Suspicious HTTP uploads or strange user-agents
- TLS handshake anomalies (self-signed certs, mismatched SNI)
- Payload signatures of exploit kits
- DNS queries to known malicious domains

Because the engine is Suricata-compatible, enterprises can import existing IPS rule sets, maintaining consistency between on-prem and cloud threat detection.

---

## 4 — Domain Filtering and Malware Command-and-Control Detection

One of the strongest DPI capabilities in Network Firewall is **stateful domain filtering**.

This mechanism allows the firewall to block DNS queries to:

- malicious command-and-control domains
- phishing sites
- known malware distribution hosts
- unauthorized third-party services
- corporate policy violation domains (e.g., personal cloud storage)

The firewall compares queried domains against allowlists/denylists and optional threat-intelligence lists.

### Domain Filtering Flow Diagram

```
DNS Query --> Firewall --> Domain List Match --> Verdict
```

Blocking malicious DNS at the firewall prevents malware from completing its infection chain, even if the instance is compromised.

---

## 5 — Advanced Threat Signatures and Threat Intelligence Feeds

Network Firewall supports:

- customer-managed Suricata rule groups
- AWS-managed rule groups (threat protection by Amazon)
- third-party managed rule groups (e.g., CrowdStrike, Trend Micro, Fortinet, Alert Logic)

These rule groups add enterprise-grade threat intelligence, including:

- Zero-day exploit patterns
- URL reputation feeds
- Malware file signatures
- Botnet C2 infrastructure lists
- Known phishing domains
- Suspicious TLS fingerprints
- Protocol misuse detection

Suricata rules can also detect:

- SQL injection patterns
  - XSS payloads embedded inside HTTP metadata
  - DDoS patterns
  - Known exploit frameworks
- 

## 6 — Detecting Lateral Movement and Internal Threat Activity (East-West DPI)

DPI is not just for internet traffic. The stateful engine inspects **east-west flows**, enabling detection of:

- unauthorized SSH or RDP between internal systems
- SMB lateral movement patterns
- unexpected service-to-service communication
- credential spraying or brute-force attempts
- abnormal internal DNS queries
- scanning activity inside the VPC
- traffic that jumps between isolation boundaries

In microservice-heavy architectures, east-west DPI becomes one of the strongest defenses against internal propagation of threats.



## East-West DPI Diagram

```
Subnet A ----> Firewall ----> Subnet B
                |
                +---> DPI: Detect scans / SMB misuse / internal C2
```

---

## 7 — DPI Logging and SIEM Integration for Threat Hunting

DPI logs contain detailed information:

- signature ID
- matched rule name
- domain name
- flow metadata
- protocol details
- source/destination IP and port
- event severity
- timestamps
- packet metadata
- action taken (alert/drop/allow)

Logs are delivered to:

- S3 (for long-term storage)
- CloudWatch Logs
- Kinesis Firehose (for Splunk, ELK, Snowflake, etc.)

Security teams analyze these logs in SIEM tools for:

- threat correlation
- anomaly detection
- incident response
- forensic investigations
- compliance auditing

## Logging Pipeline Diagram



## 8 — How DPI Works With Encrypted Traffic (TLS Inspection Limits)

AWS Network Firewall does **not decrypt TLS traffic**, but it still performs meaningful DPI using metadata-based analysis:

It inspects:

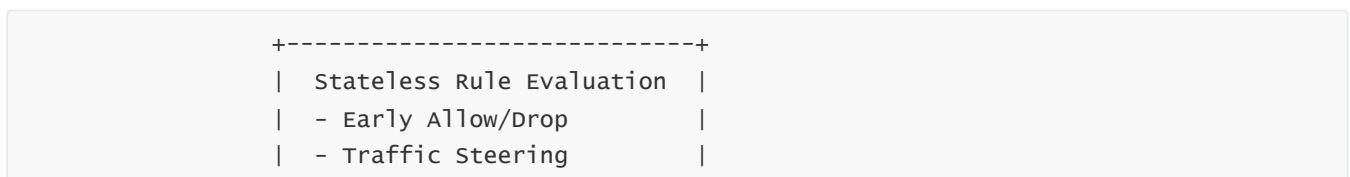
- TLS SNI (Server Name Indication)
- certificate issuer
- certificate age/metadata
- encrypted handshake anomalies
- protocol version
- cipher choices
- known bad TLS fingerprints
- suspicious certificate reuse
- domain name inside the SNI (for blocking)

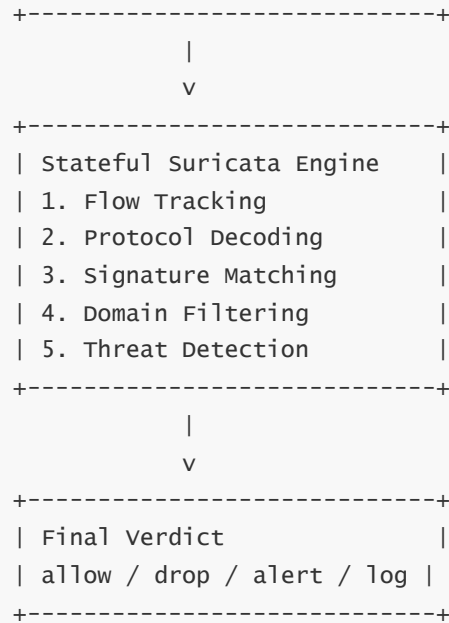
This is enough to block:

- malicious HTTPS domains
- self-signed / suspicious certificates
- mismatched SNI vs destination IP
- TLS downgrade attacks

Even without decryption, metadata-based TLS inspection is an effective enterprise control.

## 9 — End-to-End DPI and Threat Detection Architecture





### Explanation:

This depicts the complete DPI lifecycle: stateless → stateful → inspection → threat decision.

## 13 — How to configure logging, flow logs, alert logs, and centralized analytics pipelines in AWS Network Firewall

### 1 — The Purpose of Logging in Enterprise Security Architecture

Logging is one of the most critical components of AWS Network Firewall because the firewall's effectiveness depends not only on blocking malicious traffic but also on **detecting, analyzing, auditing, and responding to threats**. Large enterprises cannot operate a security program without full visibility into:

- which traffic was allowed
- which traffic was dropped
- which rules were triggered
- which domains were queried
- which IPS signatures matched
- what type of lateral movement occurred
- which workloads attempted unauthorized communication
- whether outbound internet calls carried malicious intent
- how traffic behaved over time across all VPCs
- who attempted to bypass security rules

-

AWS Network Firewall logs provide granular insights into every inspected flow and signature-triggered event. These logs become the foundation for incident response, threat hunting, forensics, compliance reporting, and continuous security monitoring.

---

## 2 — Types of Logs Generated by AWS Network Firewall

AWS Network Firewall generates **three primary categories of logs**, each serving a distinct purpose:

### a) Flow Logs (Stateful Flow Metadata)

Flow logs record connection-level metadata such as:

- source/destination IP
- ports
- protocols
- action (allow/drop)
- byte/packet counts
- connection state changes
- flow start/stop timestamps

Flow logs show *behavior*, not threats — very important for segmentation validation and traffic pattern analysis.

### b) Alert Logs (DPI + Suricata IPS Events)

Alert logs provide deep packet inspection results and record every time a Suricata rule is triggered. These logs contain:

- rule ID and rule name
- domain name matched
- signature matched
- protocol decoder triggered (DNS/HTTP/TLS/SMB/etc.)
- threat category
- severity

These logs represent **actual security-relevant events**.

### c) Custom Action Logs (Tagging Events for Analytics)

Custom stateless actions let you attach labels or metadata to flows. These tags appear in logs and are used for:

- analytics
- segmentation audits
- SOC investigation workflows
- routing analysis

Custom tags do not modify packet flow but add observability to logging streams.

---

### 3 — How Logs Are Delivered (Three Destinations)

AWS Network Firewall supports three destinations for all log types:

1. **Amazon S3** — long-term immutable storage
2. **Amazon CloudWatch Logs** — real-time analysis, dashboards, alerts
3. **Amazon Kinesis Data Firehose** — streaming into SIEM platforms (Splunk, ELK, Snowflake, QRadar)

You may choose one or many destinations simultaneously.

#### Logging Delivery Path Diagram



#### Explanation:

The firewall pushes logs to one or more destinations. Firehose can then forward logs to SIEMs for advanced analytics.

### 4 — Configuring S3 Logging (Best for Long-Term Retention)

S3 logging is ideal for compliance-driven organizations (PCI-DSS, SOX, NIST, ISO). Logs stored in S3 can be retained for years, analyzed using Athena, or loaded into a data lake.

To configure S3 logging:

- Create an S3 bucket specifically for firewall logs
- Enable server-side encryption (SSE-KMS) for compliance
- Apply bucket lifecycle rules to archive logs (Glacier)
- Set the bucket policy to allow the firewall log delivery IAM service principal
- Enable log override protection (blocked log overwrites)

S3 logs are batched efficiently and cost-effective at scale.

### 5 — Configuring CloudWatch Logging (Real-Time Monitoring)

CloudWatch Logs allow SOC teams to build real-time dashboards and alerts.

Use CloudWatch for:

- alerting on repeated malicious signatures

- anomaly detection
- operational monitoring (traffic drops, unusual patterns)
- dashboards for inbound/outbound flows
- integration with Lambda for automated responses

However, CloudWatch is **not suitable** as the long-term archive due to cost.

Best practice: use CloudWatch for real-time + S3 for long-term retention.

---

## 6 — Configuring Kinesis Firehose (SIEM Streaming + Real-Time Security Analytics)

Kinesis Firehose is the recommended method for sending logs into enterprise SIEM platforms such as:

- Splunk
- Elastic (ELK Stack)
- Snowflake
- Datadog
- QRadar
- ArcSight
- Chronicle
- Panther

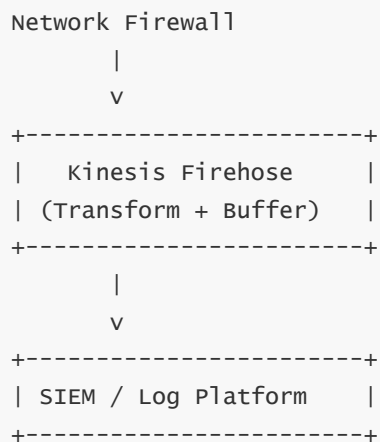
–

It allows near real-time ingestion, buffering, compression, normalization, and encryption before delivering logs to downstream systems.

Firehose enables:

- SOC dashboards
- threat hunting
- correlation with IAM events, VPC flow logs, CloudTrail
- automatic incident ticket creation
- merging firewall and workload telemetry for full context

### Firehose Pipeline Diagram



---

## 7 — Log Structure (What Information Logs Contain)

Logs include a rich set of attributes such as:

- timestamp
- firewall name
- firewall policy
- rule group
- Suricata rule signature ID
- domain queried (for DNS logs)
- threat category
- severity
- source/destination IP and port
- protocol
- flow direction
- action taken (allow/drop/alert)
- packet flags (SYN, ACK, FIN, RST)
- flow state transitions
- session duration
- bytes transferred

This level of detail allows deep forensic analysis and full replay of suspicious traffic events.

---

## 8 — Centralized Logging for Multi-Account Environments

When operating in an AWS Organizations environment, logs must be centralized.

The architecture is:

Multiple Accounts --> Firewall Logs --> Central Logging Account

You configure the logging account to:

- own the S3 bucket
- own the CloudWatch Log Groups
- own all Firehose streams
- enforce the retention and lifecycle policies
- serve as the SOC data lake

This ensures consistency and prevents individual teams from modifying logs or disabling logging.

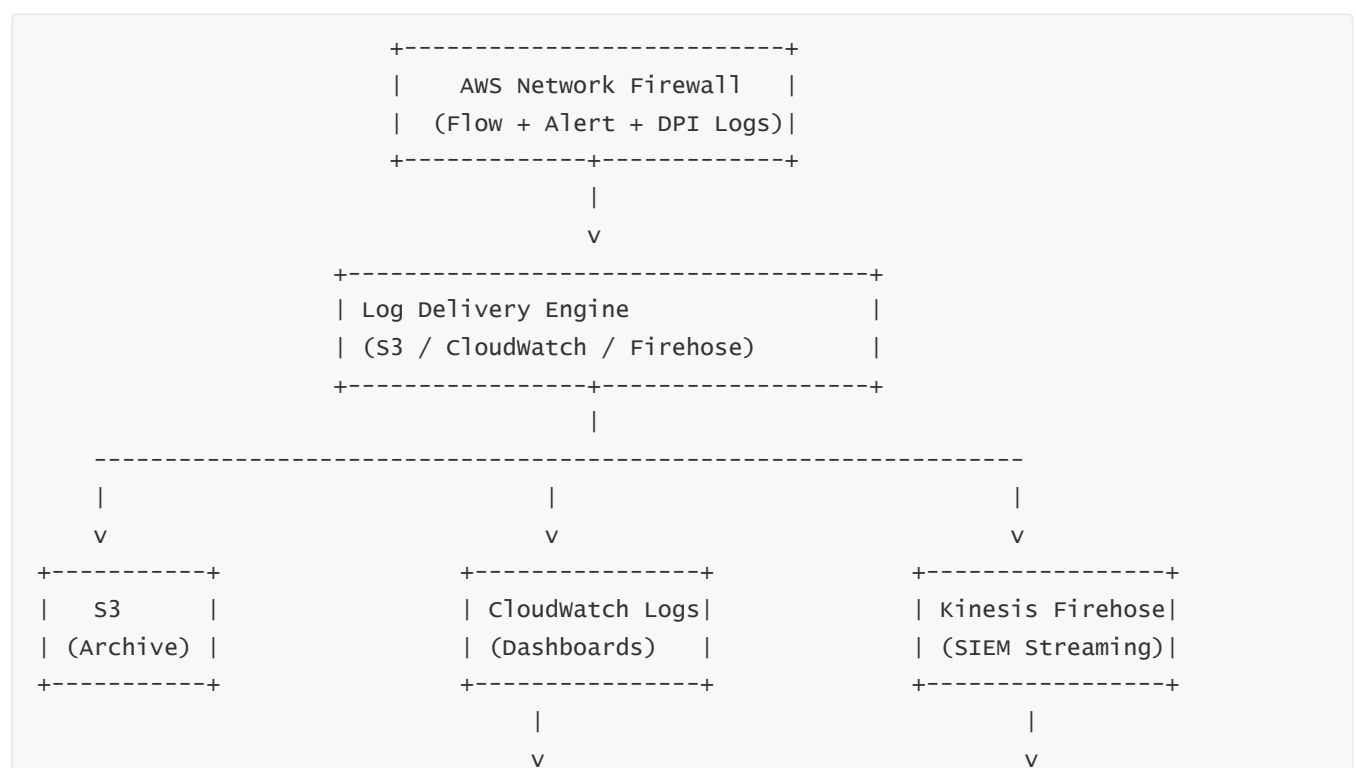
## 9 — How Logs Integrate With Firewall Manager Governance

AWS Firewall Manager ensures that:

- every firewall deployed sends logs to the correct central bucket
- logging cannot be disabled by child accounts
- all required log types (flow + alert) are enabled
- SIEM ingest pipelines remain consistent
- compliance teams receive uniform visibility

This prevents teams from accidentally or intentionally circumventing log requirements.

## 10 — End-to-End Logging and Analytics Architecture (Final Diagram)





```
+-----+
| Real-Time Ops |
| Dashboards   |
+-----+
```

```
+-----+
| SIEM / Data Lake |
| Threat Correlation |
+-----+
```

### Explanation:

All logs flow to the central analytics pipeline, from where they feed SOC dashboards, SIEM threats, incident response workflows, and audit systems.

## 14 — How to manage firewall policies at scale with AWS Firewall Manager

### 1 — Why Enterprise-Scale Policy Management Is Necessary

In large AWS organizations, security teams must enforce consistent firewall rules across **hundreds of VPCs and accounts**. Without centralized governance, every development team could create its own rules or accidentally weaken security controls by misconfiguring a firewall policy.

–

AWS Network Firewall alone provides powerful inspection capabilities, but when used at scale, you need a management layer that can:

- push mandatory policies to all accounts
- prevent security drift
- enforce organization-wide rules
- ensure uniform logging
- auto-remediate missing firewalls
- simplify deployments for new VPCs

–

This is exactly the purpose of **AWS Firewall Manager** — a governance service that tightly integrates with AWS Organizations and enforces consistent policies across the entire environment.

### 2 — How AWS Firewall Manager Works With Network Firewall

Firewall Manager uses **security policies** that define how Network Firewall should be deployed and configured across accounts. These policies automatically:

- create firewalls where required
- attach mandatory rule groups
- ensure logging is enabled
- enforce tag and resource compliance

- block attempts to remove or modify governance rules
- keep configuration consistent as new accounts or VPCs are created
- remediate drift if someone changes a firewall locally
- 

Firewall Manager thus acts as a **central enforcement engine** for Network Firewall, ensuring “one source of truth” for all firewall rules in the organization.

### 3 — Key Components of Firewall Manager for Network Firewall

A Firewall Manager policy includes the following components:

#### 1. Scope (Target Accounts & VPCs)

- Which AWS accounts (OUs) the policy applies to
- All VPCs or only specific tagged VPCs
- Enforcement across all Regions or selected Regions

#### 2. Mandatory Firewall Policy

- Stateless rule groups
- Stateful rule groups
- Default stateless/stateful actions
- Required domain filtering
- IPS signatures

#### 3. Optional Customer-Owned Policies

- Application teams may add their own rule groups
- But they cannot override mandatory rules

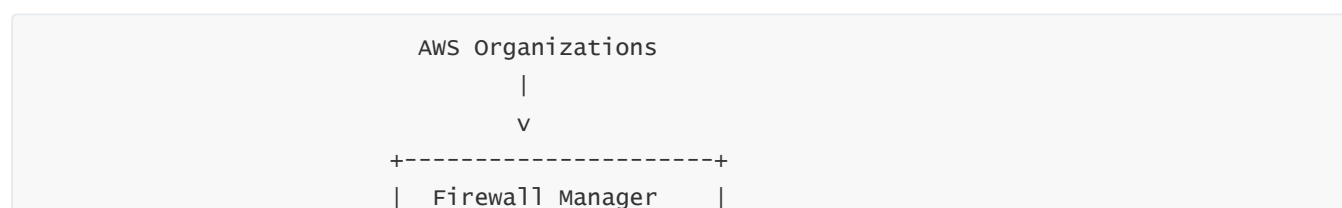
#### 4. Logging Configuration

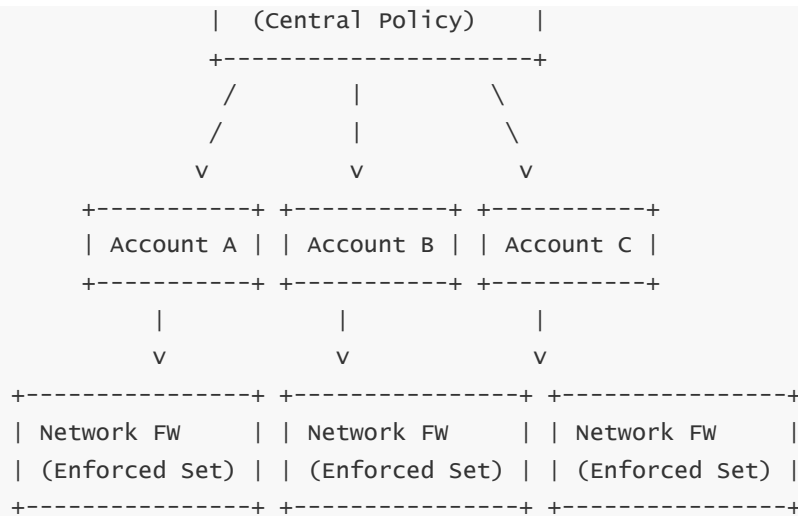
- Mandatory flow logs and alert logs
- Log destinations (S3, CloudWatch, Firehose)
- Retention enforcement

#### 5. Remediation Settings

- Automatically create firewalls in new VPCs
- Automatically fix drift
- Automatically reapply missing rule groups

### 4 — High-Level Firewall Manager Architecture Diagram





### Explanation:

Firewall Manager distributes mandatory firewall rules to all accounts. Each VPC receives a firewall with the same base policy.

## 5 — How Policy Hierarchy Works (Mandatory vs. Application-Specific Rules)

Firewall Manager supports a layered rule structure:

- **Mandatory Rule Groups:**

Applied by security teams; cannot be removed or modified by application teams.

- **Optional/Customer Rule Groups:**

Application teams can add rules for their workloads, but these must comply with mandatory rules.

Mandatory groups always apply first. This ensures that developers cannot weaken security or bypass corporate controls.

## 6 — Policy Deployment Workflow Across the Organization

### Step 1 — Security team creates a Firewall Manager policy

Defines mandatory stateless/stateful rule groups.

### Step 2 — Policy targets organizational units (OUs)

Example: All production accounts, or all accounts except security account.

### Step 3 — Firewall Manager scans the environment

Finds all VPCs within the targeted OUs.

### Step 4 — Firewall Manager automatically deploys Network Firewall

Creates firewall subnets, attaches firewalls, and configures route tables if required.

### Step 5 — Continuous compliance monitoring

If someone modifies or deletes a rule group, Firewall Manager reverts the change.

## **Step 6 — Enforcement for new VPCs**

Any new VPC created in targeted OUs automatically receives a firewall and matching policy.

---

## **7 — Drift Detection and Automatic Remediation**

Firewall Manager continuously monitors all member accounts for configuration drift:

Examples of drift detected and fixed:

- Someone removes a mandatory rule group
- Someone disables logging
- Someone modifies a firewall policy manually
- Someone deletes a firewall subnet
- Someone misconfigures a route table to bypass inspection
- Someone attempts to remove a firewall managed by Firewall Manager

When drift is detected, Firewall Manager:

- 1. Reverts the change**
- 2. Restores the correct configuration**
- 3. Logs a violation for SOC teams**

This guarantees compliance and eliminates gaps caused by human error.

---

## **8 — Enforcing Logging Across All Accounts**

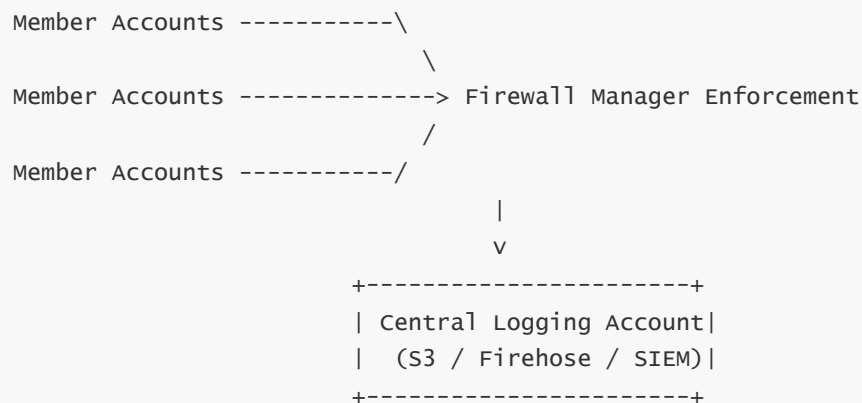
Firewall Manager ensures logging consistency:

- Flow logs and alert logs must be enabled
- Logs must be sent to a central S3 bucket or Firehose
- Retention and encryption policies cannot be bypassed
- Logging cannot be disabled by member accounts

This gives the SOC team full visibility across the entire enterprise.

---

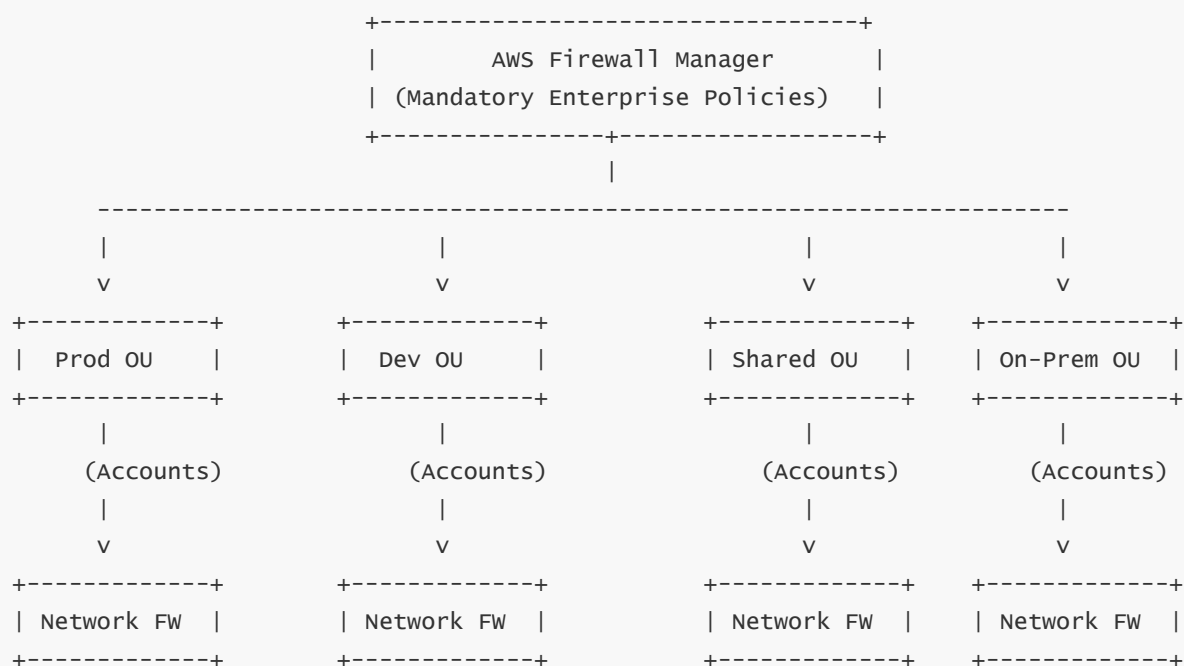
## **9 — Enterprise-Level Multi-Account Logging Pipeline**



### Explanation:

Firewall Manager gathers logs from all firewalls and consolidates them into a single logging account.

## 10 — Full Scaling and Governance Architecture (Integrated View)



### Explanation:

Firewall Manager ensures all environments meet the same security standards, regardless of who owns the account or workload.

# 15 — How to secure hybrid networks and on-premises connectivity with AWS Network Firewall

---

## 1 — Why Hybrid Network Security Is a Critical Enterprise Requirement

Modern enterprises rarely operate purely in the cloud. Most maintain **hybrid environments**, where workloads run across:

- on-premises datacenters
- private MPLS networks
- VPN gateways
- AWS Direct Connect (DX) links
- cloud VPCs across multiple Regions
- branch offices and remote sites
- partner/customer networks
- 

This creates a massive attack surface spanning both cloud and physical infrastructure. The challenge is ensuring that **traffic flowing between AWS and on-prem** is not trusted by default.

–

AWS Network Firewall becomes the **central inspection layer** for all hybrid flows by enforcing deep packet inspection (DPI), domain filtering, IDS/IPS threat detection, segmentation, and compliance policies as on-prem traffic enters or leaves AWS.

–

Without the firewall, hybrid networks create blind spots that attackers exploit through lateral movement, compromised servers, cross-environment pivot attacks, and internal threat propagation.

---

## 2 — Where Hybrid Traffic Enters AWS (The Three Entry Points)

Hybrid networks connect to AWS using:

### 1. Site-to-Site VPN

- IPSec tunnels between on-prem devices and AWS

### 2. AWS Direct Connect (DX)

- Dedicated private fiber link to AWS

### 3. SD-WAN / Third-Party Transit Providers

- CloudEdge, Megaport, Equinix, etc.

–

All three methods typically terminate at a **Transit Gateway (TGW)** or a **Virtual Private Gateway**

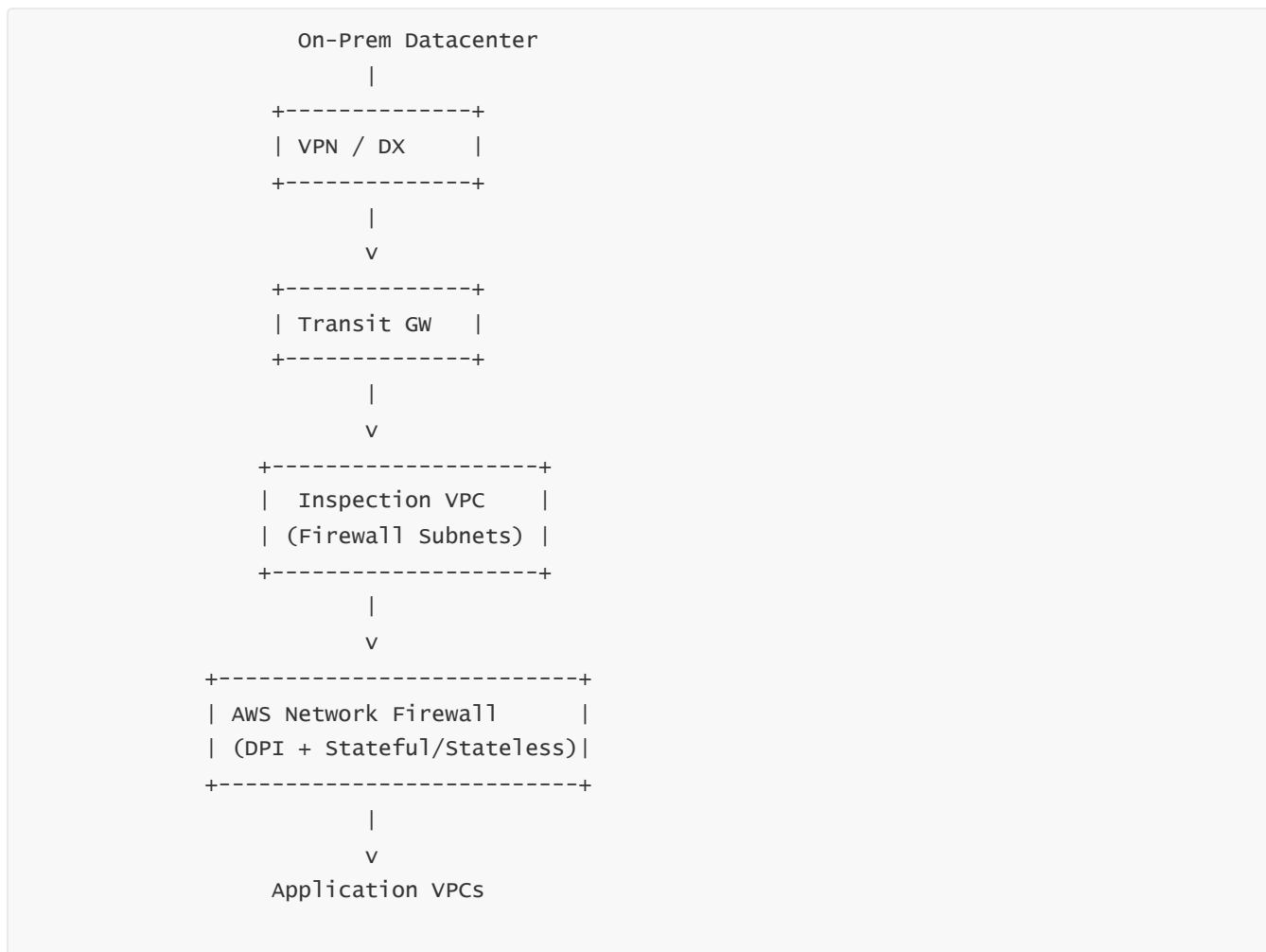
(VGW).

-

From there, traffic must be sent to **an Inspection VPC** containing the Network Firewall before reaching application VPCs.

---

### 3 — High-Level Hybrid Security Architecture Diagram



#### Explanation:

All hybrid traffic must flow from on-prem → TGW → Inspection VPC → Network Firewall → TGW → App VPCs.

---

### 4 — Securing Site-to-Site VPN Traffic

When on-prem networks connect via VPN, all decrypted IPSec traffic enters AWS unencrypted and must be inspected.

-

TGW receives this traffic from the VPN attachment.

-

The **TGW route table** must send the traffic to the Inspection VPC attachment.

-

Inside the Inspection VPC, firewall subnets route packets into Network Firewall.

–

The firewall performs:

- IPS-based threat detection
- domain filtering
- protocol anomaly detection
- segmentation enforcement

–

After inspection, the firewall returns traffic to TGW for delivery to application VPCs.

#### **VPN → TGW → Firewall Flow Diagram**

```
VPN --> TGW --> Inspection VPC --> Firewall --> TGW --> App VPCs
```

---

## **5 — Securing Direct Connect (DX) Traffic**

Direct Connect brings private traffic into AWS via physical fiber circuits.

–

Many enterprises mistakenly assume DX traffic is safe because it is private. This is a major security misconception.

–

Compromised on-prem systems can freely move laterally into AWS unless controlled.

–

Therefore, DX attachments must route traffic through Network Firewall exactly like VPN attachments.

DX security pipeline:

```
DX --> TGW --> Inspection VPC --> Firewall --> TGW --> VPC
```

#### **Key Controls Applied to DX Traffic:**

- IPS signatures for malware spreading between datacenters
  - DNS filtering for outbound C2 attempts
  - detection of abnormal SMB/RDP/SSH lateral movement
  - micro-segmentation enforcement
  - prevention of cross-environment pivot attacks
  - outbound domain restrictions and compliance filters
-



## 6 — How Network Firewall Prevents Lateral Movement Across Hybrid Networks

Hybrid networks are a major source of lateral movement because attackers pivot from compromised on-prem assets into cloud workloads.

–

Network Firewall blocks this by enforcing:

- Stateful connection validation
- Protocol decoding (SMB, RDP, SSH, LDAP, Kerberos, HTTP metadata)
- Signature detection (known attack families)
- East-west segmentation
- Source/destination restrictions
- Internal DNS protections
- Outbound blocklists

–

The firewall prevents malicious SMB traffic, unauthorized RDP, brute-force attempts, port scanning, and suspicious internal communication between environments.

### Lateral Movement Protection Diagram

On-Prem Server ----> TGW ----> Firewall ----> (Blocked or Allowed)

---

## 7 — Hybrid Microsegmentation Using Network Firewall

Segmentation is the cornerstone of hybrid security.

Network Firewall enables microsegmentation across AWS and on-prem by:

- Defining network zones (Prod, Dev, Corporate, DMZ, Partner)
- Writing stateful rules for each zone boundary
- Restricting protocols between zones
- Enforcing identity-based DNS filtering rules
- Controlling east-west flows between VPCs and on-prem subnets

Example segmentation rule:

“Only HTTPS from on-prem CRM servers to AWS App VPC is allowed. Block all other protocols including SMB, SSH, RDP.”

The firewall enforces this at the packet + protocol layer.

---

## 8 — Handling Asymmetric Routing in Hybrid Environments

Hybrid networks often face asymmetric routing issues:

**Forward Path:**

On-Prem → VPN/DX → TGW → Firewall → TGW → VPC

**Return Path (wrong):**

VPC → TGW → Directly back to DX (without Firewall)

This causes stateful drops because the firewall sees the forward flow but not the return flow.

To fix this:

- TGW route tables must force **both forward and return traffic** through the same Inspection VPC.
- Firewall subnets must exist in all AZs associated with the TGW.

---

## 9 — Hybrid Threat Detection: What the Firewall Identifies

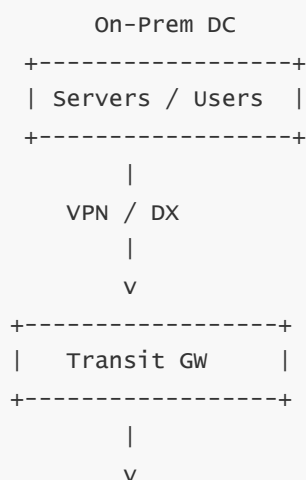
AWS Network Firewall's Suricata engine detects threats commonly found in hybrid networks:

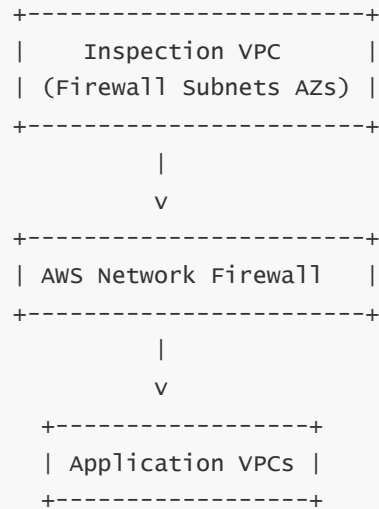
- Ransomware lateral movement (SMB spreading)
- Malware C2 callbacks (DNS/TLS metadata)
- Unauthorized database connections
- Suspicious remote admin traffic
- VPN abuse attempts
- Internal scanning or brute-force activity
- Data exfiltration via DNS or HTTP metadata
- Old/on-prem exploit kits
- Legacy protocol misuse

By inspecting all hybrid flows, you prevent both inbound and outbound vulnerabilities.

---

## 10 — End-to-End Hybrid Security Architecture (Final Integrated View)





### Explanation:

Hybrid traffic is fully inspected at the cloud entry point, preventing lateral movement, malware spread, unauthorized access, and non-compliant communication.

## 16 — How AWS Network Firewall integrates with VPC Lattice, Load Balancers, and App Mesh

### 1 — Why Application-Layer Integration Matters for Enterprise Security

Network Firewall operates at **L3/L4 (network + transport layers)** with deep packet inspection capabilities at **L7 (application metadata)**. But modern cloud architectures include **service meshes, service-to-service routing layers, and application load balancers**.

To secure such architectures, Network Firewall must integrate meaningfully with higher-level traffic management systems like:

- **VPC Lattice** (application-to-application connectivity)
- **Elastic Load Balancing (ALB/NLB)**
- **AWS App Mesh** (service mesh running sidecars/proxies)

These integrations ensure that **application traffic**, not just subnet-level traffic, passes through the firewall, providing complete inspection for all east-west and north-south flows across services, microservices, accounts, and environments.

## 2 — Integration With VPC Lattice (Service-to-Service Communications)

---

### Lattice Overview

VPC Lattice provides application-level connectivity across multiple VPCs, removing the need for complex routing. It creates **service networks** that allow services in different VPCs to talk to each other securely without managing IP routing manually.

–

But even with Lattice, enterprises still require **network-level security inspection**. Lattice focuses on **application connectivity, authN/authZ**, while Network Firewall focuses on **traffic inspection and threat detection**.

### How Network Firewall Integrates With Lattice

There are two ways to integrate Lattice with Network Firewall:

---

#### A) Network Firewall as a Mandatory Path for all Lattice Traffic

Traffic flow becomes:

```
Service A (VPC A)
  |
  v
VPC Lattice Edge
  |
  v
Transit Gateway (optional)
  |
  v
Inspection VPC
  |
  v
AWS Network Firewall
  |
  v
Service B (VPC B)
```

#### Explanation:

Lattice traffic moves through ENIs in VPCs, then flows through TGW or routing layers, eventually hitting the Network Firewall. This ensures:

- DPI on inter-service communications
- detection of malicious service-to-service traffic
- enforcement of zero-trust east-west controls
- prevention of lateral movement across service networks

---

## B) Using Lattice With Dedicated “Inspection Service Network”

Organizations may create **two service networks**:

1. Application service networks
2. A dedicated security/inspection network

Traffic between them is steered through Network Firewall by routing or TGW policies.

---

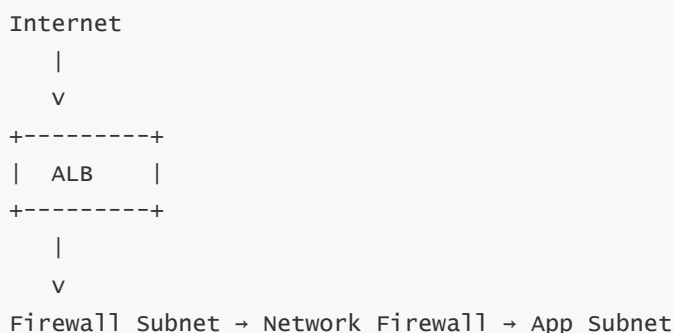
## 3 — Integration With Elastic Load Balancers (ALB/NLB)

Load balancers are the **entry and exit points** for most cloud applications. For inbound or outbound flows, Network Firewall must inspect traffic **before or after** the ALB/NLB, depending on the architecture.

---

### A) Inbound Traffic (Internet → ALB/NLB → Firewall → App)

Inbound inspection typically follows this pattern:



#### Explanation:

ALB receives external traffic, forwards it to private IP targets (firewall ENIs), and firewall inspects traffic before delivering to app servers.

#### Use Case:

- Protecting web servers from malicious payloads
  - Detecting inbound exploits
  - Enforcing allowed domains/ports
  - Suricata-based HTTP metadata inspection
-

## B) Outbound Traffic (App → Firewall → NAT → Internet)

Outbound patterns with load balancers (e.g., NLBs fronting outbound proxies):

```
App → Internal NLB → Firewall → NAT → Internet
```

### Use Case:

Outbound proxy chains, application-specific egress monitoring.

---

## C) East-West Traffic via Internal Load Balancers

Service-to-service calls inside a VPC often go through internal NLBs or ALBs:

```
Service A → Internal ALB/NLB → Firewall → Service B
```

### Purpose:

- detect internal misuse
  - prevent lateral movement
  - enforce segmentation boundaries
  - inspect traffic for abnormal behavior
- 

# 4 — Integration With AWS App Mesh (Envoy Proxy Mesh)

App Mesh provides **service mesh functionality** by installing **Envoy sidecar proxies** next to application containers/EC2 instances.

–

While App Mesh secures service-to-service traffic at L7 (authentication, retries, service discovery), it still relies on **network-level routing** underneath.

–

That network routing can (and should) pass through AWS Network Firewall.

---

## A) How App Mesh Traffic Flows

```
Service A
  |
Envoy Proxy A
  |
```



**Explanation:**

Even though Envoy handles service-level routing, the underlying packet path still passes through subnets and VPC route tables — which can be forced into the firewall for inspection.

## B) Why DPI Is Critical Even With a Service Mesh

Even with mTLS and encryption, Network Firewall performs:

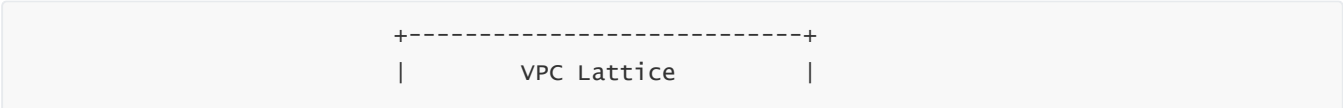
- metadata-based DPI
- SNI-based domain filtering
- TLS handshake anomaly detection
- flow-level IDS/IPS detection
- policy enforcement at the network boundary

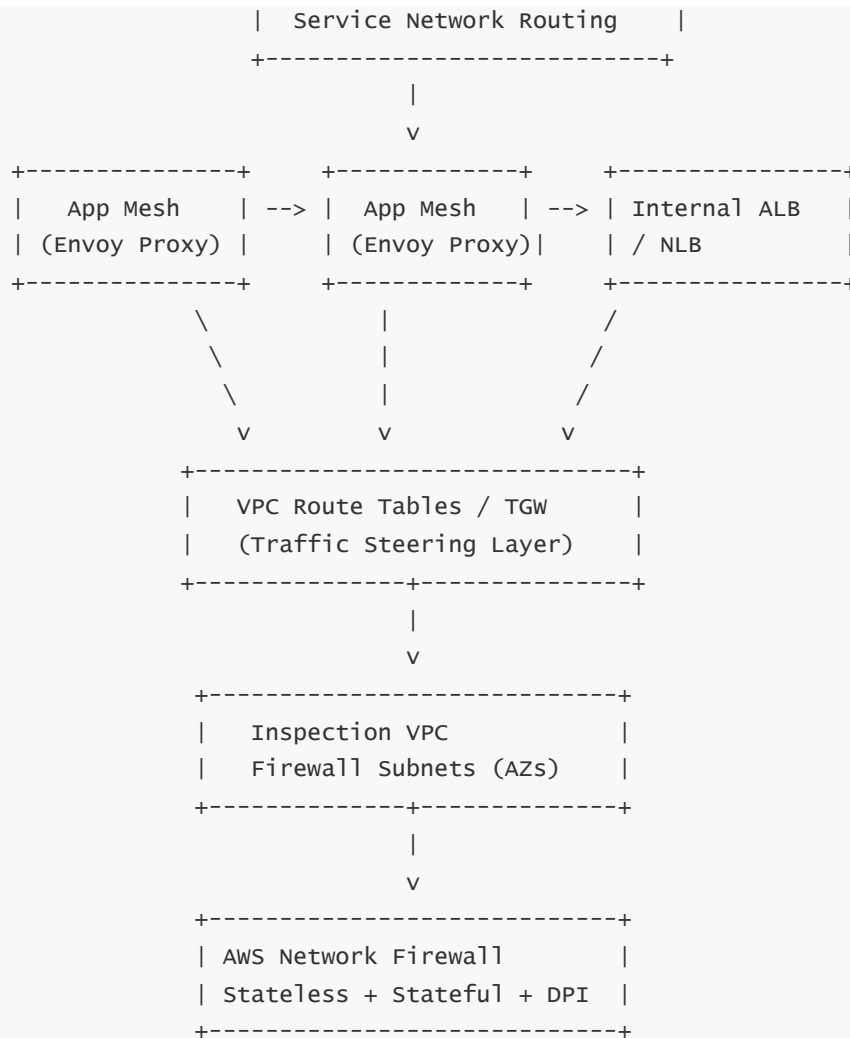
App Mesh does not provide:

- malware detection
- signature inspection
- domain reputation filtering
- threat correlation
- protocol anomaly detection

Network Firewall fills this gap.

## 5 — Combined Architecture Diagram: Lattice + LB + App Mesh + Firewall





### Explanation:

All traffic — whether from Lattice, App Mesh, or Load Balancers — eventually flows through the routing layer (VPC RTB / TGW), which enforces Network Firewall inspection.

## 6 — Security Use Cases Enabled Through These Integrations

### A) Lattice + Firewall

- Secure multi-VPC service-to-service communication
- Prevent cross-account lateral movement
- Enforce global segmentation policies
- Apply DPI to microservice boundaries

### B) Load Balancers + Firewall

- Block malicious inbound requests (HTTP exploits, C2 payloads)
- Enforce port/protocol policies before workloads receive traffic



- Protect internal services from misconfigured or compromised clients

### C) App Mesh + Firewall

- Detect east-west attacks within mesh
- Protect encrypted mTLS traffic using metadata-based DPI
- Maintain segmentation boundaries below the service mesh
- Prevent bypass of mesh-level policies through IP-level misuse

---

## 7 — Final End-to-End Integration Flow (Unified Model)

---

```
Service→Envoy(App Mesh)→Lattice/NLB/ALB→VPC Route Tables/TGW  
→Inspection VPC→Network Firewall→Destination Service/VPC
```

### Explanation:

Regardless of the application-level routing (Lattice, Mesh, LB), the network routing beneath it enforces firewall inspection. This ensures unified threat detection, segmentation, and compliance across all application connectivity methods.

---

## 17 — How to monitor, observe, and troubleshoot AWS Network Firewall in large-scale environments

---

### 1 — Why Monitoring and Troubleshooting Are Critical in Enterprise Deployments

AWS Network Firewall becomes the **security backbone** for inbound, outbound, and east-west traffic. When operating at scale across hundreds of VPCs and thousands of workloads, monitoring and troubleshooting are essential for:

- detecting anomalies
- validating segmentation boundaries
- ensuring firewall capacity health
- identifying rule conflicts
- tracing dropped packets
- investigating DPI alerts
- ensuring symmetric routing
- preventing service outages

- correlating firewall events with application behavior

- 

Without strong observability, enterprises risk silent packet drops, rule misconfigurations, and blind spots in threat detection.

AWS Network Firewall provides multiple monitoring layers:

**VPC Metrics, Firewall Metrics, Logging Streams, Flow Records, DPI Alerts, TGW Route Analysis, Packet Tracing, CloudWatch Dashboards, SIEM Integration,** and more.

---

## 2 — Core Monitoring Signals for Network Firewall

---

AWS exposes several categories of metrics and logs:

### A) Firewall-Level Metrics (CloudWatch)

Key metrics include:

- BytesProcessed
- PacketsProcessed
- AllowedPackets
- DroppedPackets
- AlertedPackets
- FlowCount
- EndpointHealth (per-AZ endpoint status)
- FirewallCapacityUsage (auto-scaling indicators)

These metrics show:

- traffic volume
  - whether firewall endpoints are healthy
  - whether flows are being dropped
  - potential bottlenecks
  - scaling behavior of distributed endpoints
-

## B) Endpoint-Level Metrics (Internal AWS Managed)

AWS monitors the health of each firewall endpoint node across each AZ.

These internal metrics are not exposed directly, but failures result in:

- new nodes being launched
- unhealthy nodes replaced
- flows rebalanced

Your signals are visible through:

- CloudWatch Events (health notifications)
  - metrics spikes
  - sudden packet drops
- 

## C) Logging Streams (Flow / Alert / Custom)

Logging is the **primary observability mechanism**:

**Flow Logs** → show behavior

**Alert Logs** → show threats

**Custom Action Logs** → show segmentation and policy actions

These logs are essential for troubleshooting rule decisions and drop behavior.

---

# 3 — Real-Time Monitoring Dashboards

Security teams build **CloudWatch dashboards** that show:

- per-AZ packet count
- firewall endpoint health
- count of allowed vs. dropped packets
- flows per second (stateful engine load)
- DPI alert count grouped by signature category
- DNS domain filtering hits
- inbound vs. outbound traffic ratios
- abnormal increases in drop traffic
- anomalies in flow creation

**Dashboard View (Conceptual Diagram)**

Enterprise Firewall Monitoring Dashboard	
Allowed Packets (graph)	Dropped Packets (graph)
Stateful Flow Count	DNS Blocked Domains
DPI Alerts by Severity	Endpoint Health Status

## 4 — Monitoring Firewall Endpoint Health (AZ-Level)

Because Network Firewall uses **distributed endpoint nodes**, endpoint health is crucial.

A firewall is considered healthy when:

- All configured AZs have active endpoints
- Nodes within each AZ respond to AWS health checks
- Flow distribution across nodes is balanced
- No AZ lacks firewall subnets
- No route table bypass exists

Indicators of endpoint issues include:

- sudden drop spikes
- sudden drop to zero allowed packets in one AZ
- asymmetric routing errors
- failures in cross-AZ traffic flows (should never happen)
- health notifications in CloudWatch Events

## 5 — Troubleshooting Stateful Drops (Most Common Issue)

Stateful inspection requires **flow symmetry**.

If return traffic bypasses the firewall or enters through another AZ, the packet is dropped because the firewall cannot find the flow state.

## Symptoms of asymmetric routing:

- connections reset
- intermittent failures in multi-AZ architectures
- inbound/outbound half-working
- flows drop only in one direction
- DPI not triggered for return flows

## How to troubleshoot:

1. Verify that forward and return paths go through the same firewall.
2. Check route tables for incorrect next-hop entries.
3. Ensure TGW routing sends both directions into the Inspection VPC.
4. Confirm firewall subnets exist in every TGW AZ.
5. Use VPC Reachability Analyzer (supports Network Firewall).

---

## Flow Asymmetry Diagram (Problem)

```
Forward: VPC A --> Firewall --> TGW --> VPC B
Return:  VPC B --> TGW --> VPC A  (bypasses firewall) ❌
```

## Correct Flow Diagram

```
Forward: VPC A --> Firewall --> TGW --> VPC B
Return:  VPC B --> TGW --> Firewall --> VPC A ✓
```

If return traffic does not follow the same path, stateful engine drops the packets.

---

# 6 — Troubleshooting Stateless Drops

Stateless rules drop traffic **before** stateful inspection.

Possible causes:

- rule priority too high
- wide match patterns (0.0.0.0/0)
- incorrect port/protocol definitions
- default stateless action is “drop”
- incomplete rule groups

- custom action incorrectly configured

To troubleshoot:

- check the stateless rule group ordering
  - inspect flow logs (stateless drops appear clearly)
  - confirm default rule actions
  - test using packet capture (VPC Traffic Mirroring)
- 

## 7 — Troubleshooting Stateful DPI Alerts

---

When DPI identifies a signature match, it may drop or alert.

Common reasons for DPI-triggered drops:

- HTTP exploit signatures
- malicious DNS queries
- suspicious TLS metadata
- protocol decoding anomalies
- SMB or RDP internal threats
- lateral movement detection
- known malware callback signatures

### How to troubleshoot:

- inspect alert logs
  - identify Suricata rule ID
  - verify if false positive
  - adjust rule group ordering or bypass rules for known safe flows
  - ensure domain lists are correct
- 

## 8 — How to Diagnose Routing Problems (Most Important Operational Task)

---

Routing is responsible for at least **80%** of firewall-related outages.

Key checks:

- firewalls must never be placed in public subnets
- private subnet RTBs must send 0.0.0.0/0 to the firewall
- firewall subnet RTBs must send outward traffic to NAT/IGW/TGW

- TGW attachment route tables must point to Inspection VPC
- reverse path must be identical
- ensure no overlapping CIDRs bypass firewall inadvertently

## Routing Troubleshooting Diagram

```

Private Subnet ---> Firewall Subnet ---> NAT/TGW/IGW
          ^                               |
          |                               |
          +----- Must match -----+

```

# 9 — Advanced Tools for Observability

## A) VPC Flow Logs + Firewall Logs Integration

Combining VPC flow logs with firewall logs creates a full picture:

- VPC Flow Logs show network-level behavior
- Firewall logs show allow/drop/alert decisions

Together they reveal:

- who initiated traffic
- where it was blocked
- which firewall rule applied
- which application was involved

## B) VPC Reachability Analyzer (Now Supports Network Firewall)

This tool helps validate connectivity through:

- firewall endpoints
- route tables
- TGW routing domains
- NAT/IGW
- security groups
- NACLs

It visually shows where packets are dropped.

## C) Traffic Mirroring

For deep troubleshooting, traffic mirroring captures packets from EC2 instances and forwards them to packet analyzers for comparison with firewall logs.

## D) SIEM Platforms

SIEMs correlate firewall logs with:

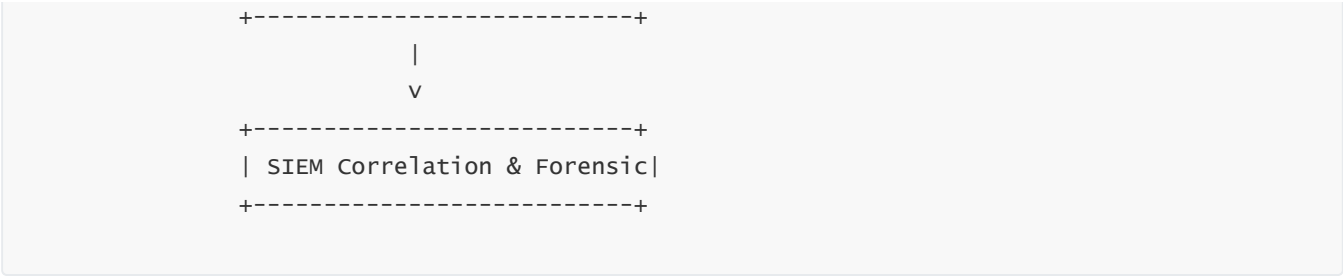
- CloudTrail
- IAM logs
- GuardDuty findings
- VPC Flow Logs
- Endpoint detection logs
- Application logs

This provides complete incident investigation capabilities.

# 10 — Full Troubleshooting Workflow Diagram (End-to-End)







**Explanation:**

This workflow ensures systematic detection, diagnosis, and resolution of network firewall issues.

This workflow ensures systematic detection, diagnosis, and resolution of network firewall issues.

# 18 — How to design disaster recovery, multi-Region strategies, and failover architectures with AWS Network Firewall

## 1 — Why Disaster Recovery and Multi-Region Security Are Critical

Enterprises increasingly operate in **multi-Region architectures** for compliance, latency optimization, business continuity, and geopolitical sovereignty.

When workloads fail over from one Region to another (planned or unplanned), **security controls must fail over with them.**

AWS Network Firewall plays a central role in ensuring that:

- inbound, outbound, and east-west traffic is still inspected
- segmentation boundaries remain intact
- threat detection continues uninterrupted
- compliance requirements are still met
- routing continues to enforce mandatory inspection

An outage in one Region must not break security protections or result in uninspected bypass paths. Therefore the firewall architecture must be globally resilient.

## 2 — Multi-Region Inspection Patterns (Three Core Models)

Enterprises generally adopt one of three patterns:

## A) Active-Active Multi-Region Firewall Inspection

Both Regions run:

- their own Transit Gateways
- their own Inspection VPCs
- their own firewall subnets
- their own firewall policies
- their own egress and ingress VPCs
- 

Traffic in each Region is inspected locally.

If Region A fails, workloads shift to Region B where inspection is already running.

### Diagram — Active-Active

```
Region A: TGW → Inspection VPC → Network Firewall → App VPCs
Region B: TGW → Inspection VPC → Network Firewall → App VPCs
```

This provides the **fastest failover** and no delays in scaling security.

---

## B) Active-Passive Multi-Region Firewall Inspection

Region A has the primary firewall environment.

Region B has the secondary firewall environment that is scaled down, but ready.

-

When workloads fail over, routing (TGW + VPC RTBs) switches traffic into Region B's firewall.

### Diagram — Active-Passive

```
Primary Region A: Firewall always active
Secondary Region B: Firewall cold/standby, activated during failover
```

This reduces cost but increases failover time.

---

## C) Centralized Multi-Region Inspection Using Inter-Region TGW Peering

In very large enterprises, Regions are connected using **inter-Region Transit Gateway peering**, and each Region has its own Inspection VPC.

Traffic pattern:

```
Region A TGW → Region A Firewall  
Region B TGW → Region B Firewall  
Inter-Region traffic → Peering → Respective firewalls
```

This allows consistent security across global networks.

---

## 3 — Replicating Firewall Policies and Rule Groups Across Regions

---

AWS Network Firewall configurations are **Region-scoped**.

This means:

- Stateful and stateless rule groups must be recreated in each Region
- Firewall policies must be duplicated
- Rule IDs and signatures must match
- Domain filtering lists must be synchronized
- Logging destinations must be Region-appropriate
- Firewall Manager policies can automate replication

### **Best Practice:**

Store firewall rules as **infrastructure-as-code** (IaC) using:

- AWS CloudFormation
- Terraform
- AWS CDK
- Firewall Manager policy templates

Then deploy these templates into all Regions.

---

## 4 — Designing Multi-AZ Resiliency Inside Each Region

---

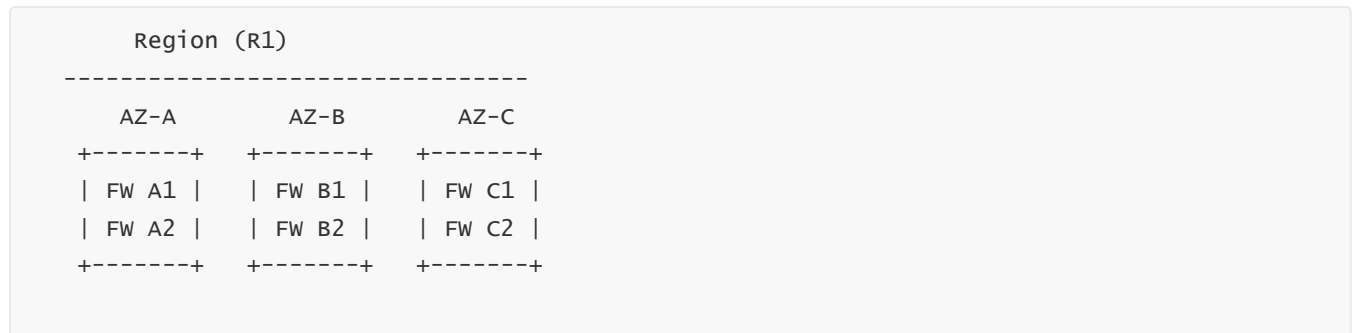
Within one Region, high availability is provided by:

- firewall subnets in each AZ
- distributed endpoint nodes per AZ
- routing symmetry enforced by VPC/TGW
- automatic endpoint scaling
- AZ-locality rules for TGW

- endpoint auto-healing

Even if one AZ fails, traffic automatically flows through other AZs.

#### Diagram — Multi-AZ Resilience



All AZs must have firewall subnets, or TGW will drop traffic.

## 5 — Designing DR-Ready Routing and Steering

Routing is the foundation of multi-Region DR.

### A) DNS + Route 53 Failover

Applications fail over through Route 53 health checks.

But firewall inspection must also fail over.

So the failover sequence must update:

- VPC route tables
- TGW route tables
- NAT/IGW egress routing
- inbound ALB/NLB targets
- Inspection VPC endpoints (if cross-Region failover requires reconstruction)

### B) TGW Failover

Multi-Region TGW architectures require:

- TGW peering active in both directions
- Static routes for cross-Region paths
- Firewall policies identical in each Region

## C) Egress Failover

Outbound traffic must fail over to secondary NAT gateways in secondary Regions.

```
App VPC (Region A) → TGW A → Firewall A → NAT A
Failures move to:
App VPC (Region B) → TGW B → Firewall B → NAT B
```

---

## 6 — Handling Stateful Failover (Critical Design Detail)

Stateful engines like Suricata maintain flow tables.

Flows **cannot** be migrated across Regions.

Therefore, ongoing TCP sessions will drop during cross-Region failover.

Acceptable behavior? Yes.

For DR events, long-lived sessions must reconnect.

### Implication:

- Ensure idempotent, retry-safe application design
- Use stateless microservice patterns
- Implement retry logic in SDKs and client libraries

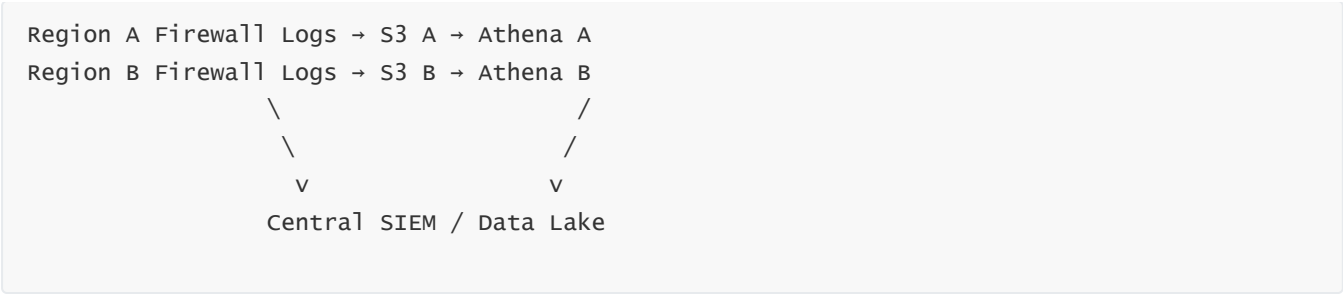
---

## 7 — Multi-Region Logging and Analytics Architecture

Logs must be:

- Regionally present for local compliance
- Globally aggregated for SOC visibility
- Durable across failures
- Queryable across Regions

### Multi-Region Logging Pipeline



Use:

- S3 bucket replication
- Firehose to SIEM
- Cross-Region log aggregation

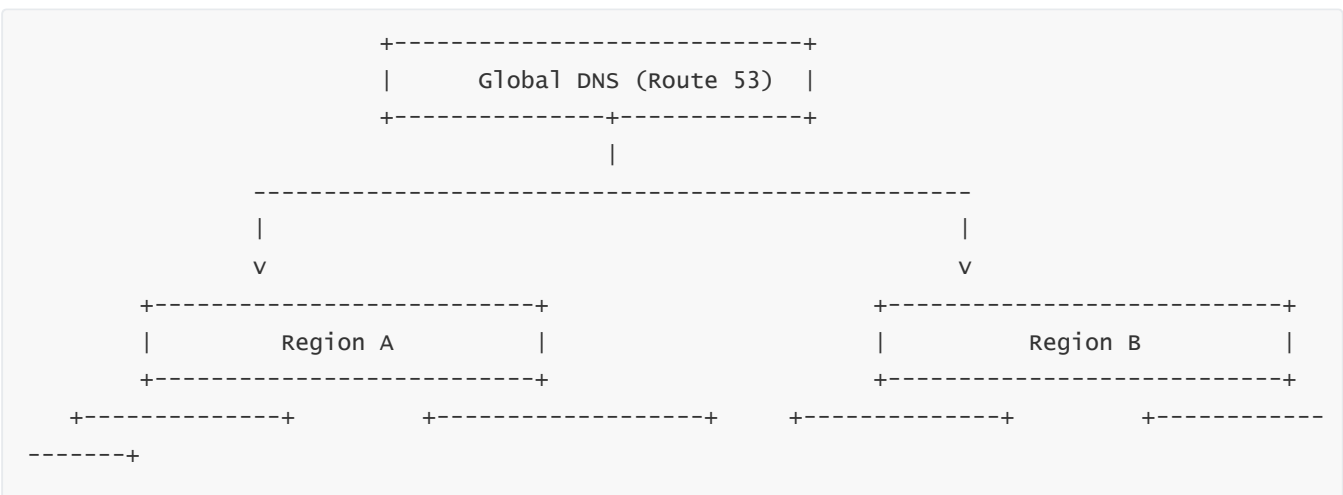
## 8 — Enforcing Multi-Region Governance With Firewall Manager

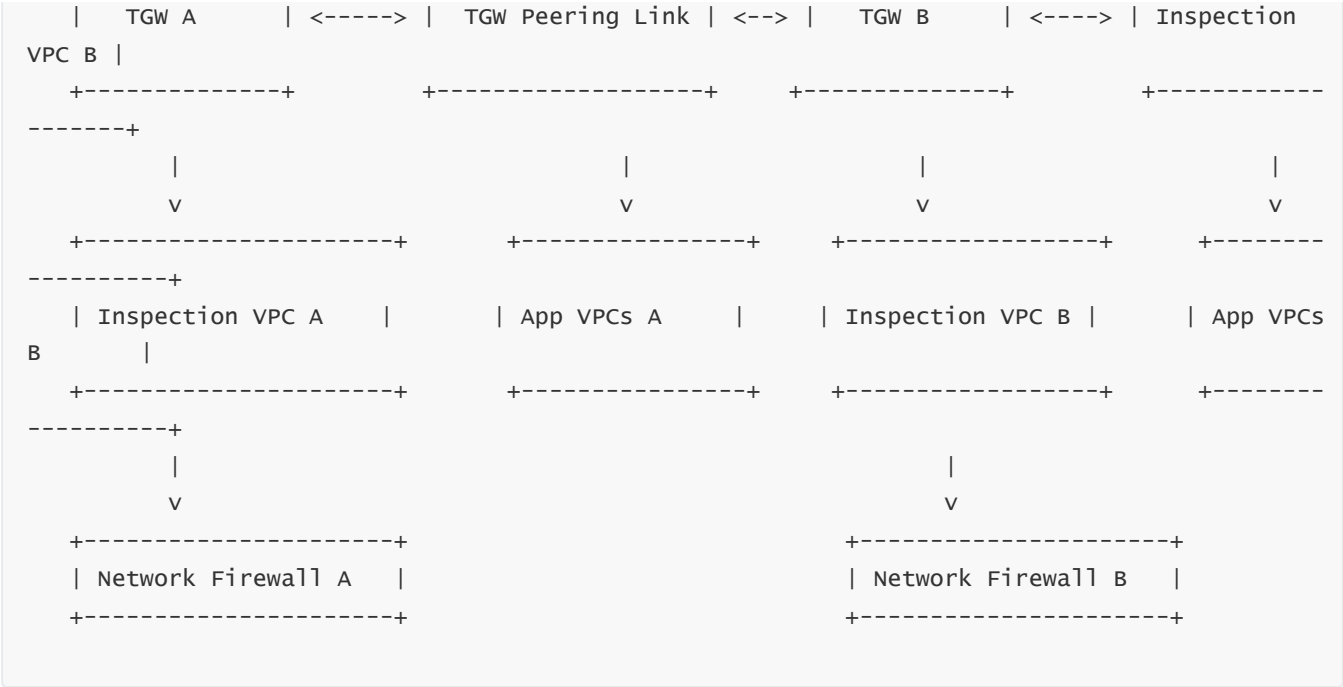
Firewall Manager automatically:

- creates firewalls in all Regions under governance
- syncs mandatory rule groups
- enforces logging destinations
- detects drift in any Region
- auto-remediates missing firewall deployments
- tracks policy violations across accounts and Regions

This ensures global uniformity.

## 9 — Full Multi-Region DR Architecture Diagram





**Explanation:**

Each Region has its own independent firewall stack, but TGW peering, DNS failover, and consistent policies ensure seamless DR and business continuity.

# 10 — Summary: Key Principles of Multi-Region Firewall DR Architecture

- Every Region must have its **own** firewall deployment
- Rule groups and policies must be **replicated** across Regions
- Inspection VPCs must exist in **all Regions** supporting workloads
- TGW routing must enforce symmetric paths in each Region
- DNS failover must redirect **both application traffic and inspection traffic**
- Log pipelines must be regional and globally aggregated
- Stateful connections must be expected to drop during DR
- Firewall Manager simplifies multi-Region governance

# 19 — Full consolidated summary of AWS Network Firewall (Unified Deep-Dive Narrative Across All 18 Questions)

## 1 — The Strategic Role of AWS Network Firewall in Modern Cloud Security

AWS Network Firewall is the **central traffic inspection layer** in enterprise AWS architectures. It is designed to secure **north-south** (inbound/outbound) and **east-west** (inter-VPC, intra-VPC, hybrid, microservice) traffic using a distributed, auto-scaling, fully managed firewall fabric.

–

At its core, Network Firewall provides **stateless packet filtering**, **stateful deep packet inspection (DPI)**, **domain filtering**, **Suricata-based IDS/IPS capabilities**, and **central policy governance**.

–

Enterprises depend on it to enforce segmentation, prevent lateral movement, detect malware, block malicious domains, secure hybrid connections, and ensure consistent governance at scale across hundreds or thousands of VPCs.

---

## 2 — The Distributed Architecture: Firewall Subnets and Endpoint Nodes

AWS Network Firewall runs inside **firewall subnets** deployed in each AZ. Inside these subnets, AWS automatically provisions multiple **firewall endpoint nodes**.

–

These nodes scale horizontally based on traffic volume and flow count. The architecture is elastic, self-healing, and transparent: AWS handles node creation, deletion, replacement, and flow redistribution.

–

Traffic enters the firewall only when routing tables or Transit Gateway (TGW) send it into these firewall subnets. Therefore, **routing is the primary control plane**, while **firewall is the data-plane inspection layer**.

---

## 3 — Stateless vs. Stateful Inspection: Two-Layer Engine

Network Firewall evaluates traffic using a two-phase process:

### Stateless Engine:

- Evaluates packets individually using match conditions (IP, port, protocol, flags).
- Uses priority numbers (lowest = highest priority).
- Actions: pass, drop, forward\_to\_stateful, custom action.
- First match wins.

### Stateful Engine (Suricata):

- Maintains full connection state.
- Performs DPI, domain filtering, signature matching.
- Detects anomalies (TCP misuse, protocol violations).
- Enforces IPS rules and domain restrictions.

Stateless controls determine whether the stateful engine evaluates a packet or whether the packet is dropped earlier.

---



## 4 — Rule Priorities, Rule Order, and Policy Hierarchies

In stateless filtering, **priority order decides everything**. Misconfigured priorities can allow or block critical traffic unexpectedly.

–

In stateful filtering, Suricata rule ordering follows an internal optimized decision tree, not a simple numeric priority.

–

At the policy level, stateless evaluation always happens **before** stateful DPI. Each policy also has **default stateless** and **default stateful** actions that determine the outcome for unmatched traffic.

---

## 5 — Handling Inbound, Outbound, and East-West Flows

Network Firewall controls three essential traffic flows:

### Inbound:

IGW → ALB/NLB → Firewall → Private Subnets

– Protects workloads from internet threats and malicious payloads.

### Outbound:

Private Subnets → Firewall → NAT → Internet

– Enforces egress restrictions, blocks malicious destinations, applies DNS filtering.

### East-West (Intra- and Inter-VPC):

Subnet-to-Subnet or VPC-to-VPC → Firewall

– Prevents lateral movement, enforces microsegmentation, detects internal threats.

– Critical for microservices, hybrid networks, and multi-VPC enterprises.

For all flows, **return traffic must follow the same path** to maintain stateful symmetry.

---

## 6 — Routing and Traffic Steering: The Most Important Architectural Component

Routing is the foundation of Network Firewall.

Traffic reaches the firewall only if routing tables send it there.

Routing models include:

– **Single VPC routing** (simple RTB → firewall → NAT/IGW)

– **TGW inline inspection** (centralized hub for many VPCs)

– **Hybrid routing** (VPN/DX → TGW → firewall)

– **Cross-Region routing** (via TGW peering)

Misconfigured routing is responsible for most failures. Correct design requires:

– firewall subnets in every AZ

- separate route tables for private, inspection, firewall, TGW attachments
  - no bypass routes around firewall
  - symmetric return paths
- 

## 7 — Transit Gateway Integration: Enterprise-Scale Security Hub

Using TGW, Network Firewall becomes a **central inspection VPC** for all VPC-to-VPC, VPC-to-on-prem, and VPC-to-internet traffic.

-

App VPC → TGW → Inspection VPC → Firewall → TGW → Destination

-

This architecture delivers predictable routing, scalable inspection, and consistent enforcement across hundreds of accounts and VPCs.

-

TGW enforces AZ-locality, so firewall subnets must exist in all active AZs.

---

## 8 — Scaling, Performance, and High Availability

Network Firewall provides:

- distributed endpoint clusters per AZ
- automatic horizontal scaling
- automatic node replacement
- deterministic flow hashing
- AZ-local high availability
- no customer responsibility for sizing or patching

Networks can process **tens of Gbps** and **millions of concurrent flows**, depending on rule complexity.

---

## 9 — Enterprise-Grade Security Architecture

A complete enterprise firewall architecture includes:

- **Inspection VPC** with firewall subnets
- **TGW** as traffic aggregation hub
- **Ingress VPC** (inbound)
- **Egress VPC** (outbound)
- **App VPCs** (workloads)
- **Firewall Manager** (governance)

- **Central logging account** (SIEM / analytics)

This design ensures mandatory inspection for all traffic paths.

---

## 10 — Deep Packet Inspection and Advanced Threat Detection

Network Firewall uses Suricata for powerful DPI, enabling:

- signature-based IPS
- domain filtering
- application protocol decoding
- TLS metadata inspection
- C2/malware detection
- lateral movement detection
- anomaly and misuse detection

Even encrypted flows can be inspected using SNI, certificate metadata, protocol fingerprints, and TLS handshake behavior.

---

## 11 — Logging, Flow Records, and Analytics Pipelines

Logging is central to firewall operations. Network Firewall generates:

- **Flow logs** (connection metadata)
- **Alert logs** (DPI signatures)
- **Custom action logs** (labels/tags)
- 

Logs are delivered to S3, CloudWatch Logs, or Kinesis Firehose → SIEM.

SOC teams use these logs for:

- threat correlation
- segmentation validation
- incident response
- forensic investigations
- compliance reporting

Enterprises centralize logs across accounts via Organizations and a dedicated logging account.

---

## 12 — Governance at Scale With Firewall Manager

Firewall Manager enforces:

- mandatory rule groups

- global domain blocklists
- outbound egress restrictions
- uniform logging
- auto-remediation of configuration drift
- firewall creation in new VPCs
- consistent policy push across Regions and accounts

This prevents misconfiguration and ensures the entire organization uses the same baseline security posture.

---

### **13 — Securing Hybrid Networks (VPN/DX)**

Hybrid traffic enters AWS through TGW or VGW and must be forced through Network Firewall to prevent:

- datacenter-to-cloud lateral movement
- malware propagation
- unauthorized database access
- exfiltration through private links

Both VPN and DX follow identical inspection flows:

On-prem → VPN/DX → TGW → Inspection VPC → Firewall → TGW → Application VPC.

---

### **14 — Integrations With VPC Lattice, Load Balancers, and App Mesh**

Application connectivity layers (Lattice, ALB/NLB, App Mesh) operate above the network-layer firewall.

Routing underneath these layers must still send traffic through Network Firewall.

This ensures that:

- microservice-to-microservice traffic is inspected
  - service-mesh encrypted flows undergo metadata-based DPI
  - inbound ALB traffic and internal ALB/NLB traffic is inspected
  - Lattice service networks cannot bypass mandatory security controls
- 

### **15 — Monitoring, Observability, and Troubleshooting**

Large environments depend on:

- CloudWatch metrics for traffic volume, drops, flow counts
- flow logs for behavioral analysis
- alert logs for threat detection
- VPC Reachability Analyzer to validate routing
- endpoint health monitoring

- packet capture (Traffic Mirroring)
- SIEM correlation
- TGW route evaluation

Most operational issues originate from routing errors or asymmetric paths.

---

## 16 — Multi-Region, DR, and Global Failover Strategies

Enterprise DR designs require:

- independent firewall stacks per Region
- replicated rule groups and policies
- redundant inspection VPCs
- Region-specific TGW integration
- cross-Region peering for multi-Region communication
- DNS failover with Route 53
- global log aggregation
- resilient egress/inbound architectures

During Region failover, active flows reset due to stateful limitations, but new flows establish normally in the secondary Region.

---

## 17 — Full End-to-End Traffic Inspection Lifecycle (Consolidated View)

Traffic flow through Network Firewall follows this pipeline:

1. Routing steers packets into firewall subnets
2. Stateless rules evaluate packets
3. If forwarded, stateful engine performs DPI
4. Suricata applies threat signatures and domain filters
5. Final verdict allow/drop/alert
6. Logs emitted to S3/CloudWatch/Firehose
7. Traffic forwarded to NAT/TGW/IGW or other VPCs
8. Return path repeats same steps, ensuring symmetry

This deterministic process applies to **all traffic** (inbound, outbound, east-west, hybrid, service-mesh, multi-Region).

---

## 18 — The Architectural Purpose of AWS Network Firewall

Across all scenarios covered in the previous questions, AWS Network Firewall consistently provides:

- network-layer control

- deep inspection
- threat detection and prevention
- segmentation enforcement
- governance consistency
- hybrid security
- routing-based enforcement
- auto-scaling
- multi-AZ high availability
- multi-Region DR capability

It becomes the **backbone** of secure network connectivity across AWS.

---

## 19 — The Universal Principle: Routing → Firewall → Routing

Every architectural model relies on the same universal pattern:

**All traffic must be routed into the firewall before reaching its destination.**

If routing is correct, security is correct.

If routing is wrong, the firewall is bypassed.

---

## 20 — Final Unified Summary

AWS Network Firewall transforms cloud security from scattered, instance-based filters into a centralized, auto-scaling, DPI-powered security layer capable of:

- protecting modern multi-VPC, multi-Region, hybrid architectures
- enforcing enterprise segmentation
- providing advanced threat detection
- supporting complex application networking (Lattice / App Mesh)
- delivering global governance through Firewall Manager
- ensuring resilient, consistent, scalable inspection for all traffic flows

It is the **core pillar** of securing large-scale AWS network topologies.

---

# 20 — Common misconceptions, pitfalls, architecture mistakes, and how to avoid them in AWS Network Firewall

---

## 1 — Misconception: “Network Firewall automatically inspects all traffic in the VPC.”

This is one of the biggest misunderstandings.

AWS Network Firewall does **not** sit inline automatically. It only inspects traffic **that routing explicitly sends to it**.

–

If a route table points traffic directly to IGW, NAT, TGW, or another subnet, the firewall is bypassed entirely.

–

### **Correct Principle:**

Firewall effectiveness is 100% dependent on **routing design**, not on the presence of the firewall itself.

Ensure all subnet route tables force inspection by forwarding traffic → firewall subnets → final destination.

---

## **2 — Misconception: “Stateful rules work even if return traffic bypasses the firewall.”**

Suricata requires symmetric flow.

If return packets do not pass through the **same firewall endpoint AZ**, the flow state is missing and packets are dropped.

–

This causes intermittent, hard-to-diagnose connectivity failures.

### **Correct Fix:**

- ensure symmetric routing
  - ensure firewall subnets exist in all TGW AZs
  - ensure forward and return path go through same inspection VPC
  - use VPC Reachability Analyzer to validate symmetry
- 

## **3 — Misconception: “The firewall decrypts TLS traffic.”**

AWS Network Firewall does **not** perform TLS decryption.

It only analyzes **TLS metadata**, including:

- SNI (server name)
- certificate issuer
- cipher suite
- TLS version
- handshake anomalies
- domain/hostname
- suspicious certificate reuse

–

DPI still works extremely well, but content inspection is metadata-driven.

---

#### 4 — Misconception: “Using NAT Gateway automatically enforces inspection on outbound traffic.”

NAT Gateways do not inspect anything.

If routing points private subnets → NAT → internet without passing through the firewall, you lose all outbound security controls.

##### **Correct Design:**

Private Subnet → Firewall Subnet → NAT → Internet.

---

## 5 — Pitfall: Missing Firewall Subnets in All AZs

---

Transit Gateway routes traffic to the **AZ-local attachment**.

If an AZ does not contain a firewall subnet, TGW silently drops traffic.

##### **Correct Pattern:**

Always deploy firewall subnets in every AZ used by:

- TGW attachments
- workload subnets
- routing domains

Missing a single AZ breaks multi-AZ traffic flows.

---

## 6 — Pitfall: Misconfigured Stateless Rule Priorities

---

Stateless rules follow strict **priority ordering**.

A broad rule (0.0.0.0/0 allow) at high priority can override intended deny rules.

Common mistakes:

- allow rules placed above deny rules
- incorrect numeric priority (lower = higher priority)
- accidental wide CIDR match
- default action set to “pass”

##### **Correct Approach:**

- place deny rules at highest priority
- use tight CIDR blocks
- review priorities regularly



- validate using traffic samples
- 

## 7 — Pitfall: Bypassing Stateful Engine Accidentally

---

If stateless rules allow traffic with `pass`, the packet bypasses stateful DPI and goes straight to routing.

Common causes:

- using `pass` instead of `forward_to_stateful`
- default action set to `pass`
- broad allow-pass rules
- unaware developers editing stateless groups

**Fix:**

- Strictly use `forward_to_stateful` for flows requiring DPI
  - Restrict `pass` rules to safe internal traffic only
  - Carefully design stateless policy defaults
- 

## 8 — Pitfall: Asymmetric Routing in TGW-Based Architectures

---

TGW → Firewall → TGW patterns often suffer from asymmetry:

Forward Path: VPC A → TGW → Firewall → TGW → VPC B

Return Path: VPC B → TGW → VPC A (bypassing Firewall)

This causes flow drops and intermittent connectivity.

**Correct Fix:**

- All TGW attachments must have matching inbound and outbound routes
  - AZ locality must be respected
  - Routing tables must be strictly symmetrical
  - Inspection VPC must be fully meshed with TGW routing
- 

## 9 — Pitfall: Wrongly Using Public Subnets for Firewall Endpoints

---

Firewall endpoints must **never** be deployed in public subnets.

Otherwise:

- IGW direct paths bypass firewall
- NAT routing loops occur
- endpoints become reachable from the internet
- no AZ-local guarantee
- TGW cannot reliably steer traffic

**Correct Design:**

Use **dedicated firewall subnets**, private with no IGW.

---

## 10 — Pitfall: Overloading Stateful Rule Groups With Too Many Custom IPS Rules

---

Suricata is powerful, but excessive custom rules can cause:

- slower flow processing
- increased CPU usage
- scaling delays
- false positives in application traffic

**Correct Pattern:**

- only add IPS rules necessary for your threat model
  - monitor performance metrics
  - use AWS-managed and vendor-managed rule groups
  - periodically review rule effectiveness
- 

## 11 — Misconception: “Terraform/CloudFormation state is enough —no need for Firewall Manager.”

---

IaC is good for resource creation, but **cannot enforce ongoing governance** across accounts.

Only Firewall Manager can:

- prevent drift
- prevent rule removal
- enforce mandatory logging
- automatically apply new rule groups
- auto-deploy firewalls in new VPCs

- maintain organizational-level compliance

IaC alone cannot enforce governance; Firewall Manager is required.

---

## 12 — Pitfall: Incorrect Hybrid Routing (VPN/DX)

---

Hybrid paths must be:

On-prem → VPN/DX → TGW → Firewall → TGW → App VPC

Common mistakes:

- routing VPN traffic directly to VPC
- bypassing inspection VPC
- asymmetric return path
- missing firewall subnets for TGW AZs

This opens the cloud to lateral movement from on-prem.

---

## 13 — Pitfall: Assuming App Mesh or Lattice Replaces Network Firewall

---

App Mesh and VPC Lattice operate at the **application layer (L7)**.

They do **not** replace network-level inspection.

They do not detect:

- malware C2
- suspicious encrypted metadata
- internal scanning
- domain-based exfiltration
- IPS signatures
- protocol misuse

Network Firewall remains mandatory.

---

## 14 — Misconception: “Network Firewall slows down the network.”

---

The distributed, multi-node, auto-scaling architecture handles large volumes.

Performance issues occur only when:

- misconfigured routes cause loops
- misconfigured stateless rules force unnecessary DPI
- incorrect flow symmetry causes drops
- huge IPS rule sets degrade compute

Fixes:

- proper routing
  - correct rule design
  - realistic IPS rule sets
  - letting AWS autoscaling handle node growth
- 

## 15 — Pitfall: Not Logging Everything

---

If you do not enable both Flow Logs + Alert Logs:

- you lose visibility
- SOC lacks forensic data
- segmentation cannot be validated
- DPI events go unnoticed
- attackers move unseen

### **Correct Practice:**

Enable all log types and centralize logs in S3 or SIEM.

---

## 16 — Pitfall: Missing Multi-Region Policy Replication

---

Policies are Region-specific.

Failure to replicate rules results in inconsistent security posture.

Fix:

- use Firewall Manager multi-Region policies
  - use IaC automation to clone rule groups
  - ensure all Regions have inspection VPCs
- 

## 17 — Pitfall: Incorrect Egress Architecture

---

Wrong:

App Subnet → NAT → Internet (no firewall)

Correct:

App Subnet → Firewall → NAT → Internet

- Enforces outbound policies
  - Blocks malicious DNS
  - Prevents exfiltration
  - Applies threat detection
- 

## 18 — Pitfall: Allowing Shadow IT to Modify Firewall Policies

---

Developers may inadvertently change rule groups or priorities.

This can break security globally.

Fix:

- only allow edits in security governance account
  - apply Firewall Manager mandatory policies
  - restrict rule editing via IAM SCPs
  - audit logs for unauthorized changes
- 

## 19 — Pitfall: Misinterpreting DPI Alerts

---

DPI alerts are **not always threats**.

Some signatures flag:

- misconfigurations
- protocol anomalies
- benign but uncommon behavior
- outdated application flows

SOC teams must:

- correlate alerts with context
  - adjust signatures
  - disable noisy irrelevant rules
  - tune policies iteratively
-

# 20 — Final Architecture and Operational Checklist (Avoid All Mistakes)

---

## Infrastructure

- Firewall subnets in every AZ
- Routing forces all traffic through firewall
- Symmetric paths ensured
- TGW inline inspection configured correctly
- No bypasses around firewall
- App → Firewall → NAT → Internet for egress
- IGW → ALB → Firewall → App for ingress

## Policies

- Deny rules at highest stateless priority
- Correct stateless default actions
- Stateful rules carefully curated
- DPI and domain filtering enabled
- Multi-Region policies consistent

## Governance

- Firewall Manager enforcing mandatory policies
- SCPs blocking unauthorized edits
- Centralized logging + SIEM
- IaC automation for repeatability

## Operations

- Monitor endpoint health
  - Use CloudWatch dashboards
  - Investigate DPI alerts
  - Validate routing with Reachability Analyzer
  - Test failover scenarios regularly
-

# Final Statement

By avoiding these misconceptions and pitfalls, AWS Network Firewall becomes a **predictable, scalable, centrally managed, deeply secure infrastructure layer** that protects all inbound, outbound, east-west, hybrid, and multi-Region traffic in your organization. It becomes the single most important network security control in enterprise AWS.

## OPTION B — Multiple Mega-Diagrams (Dimension by Dimension)

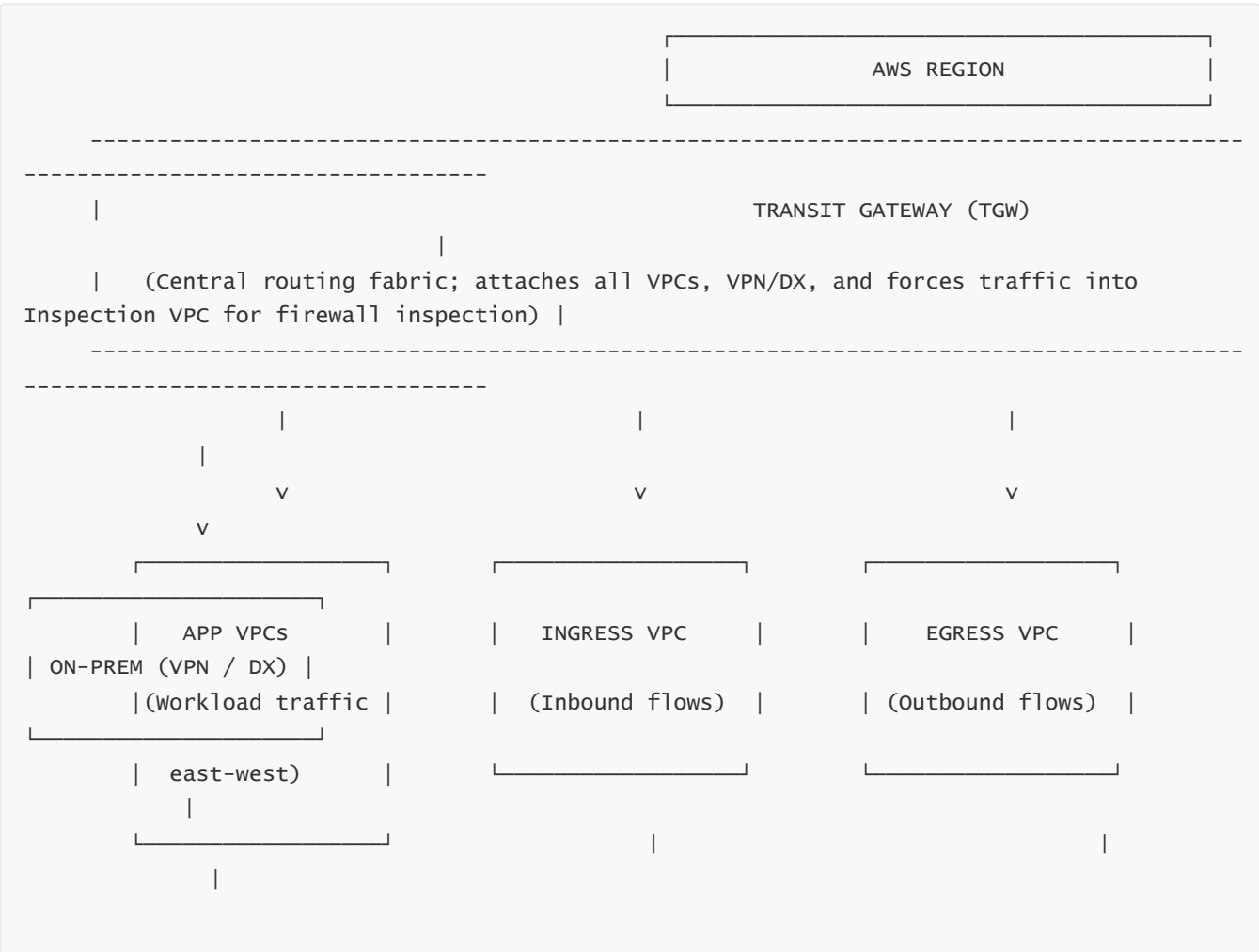
### MEGA-DIAGRAM SET BEGIN

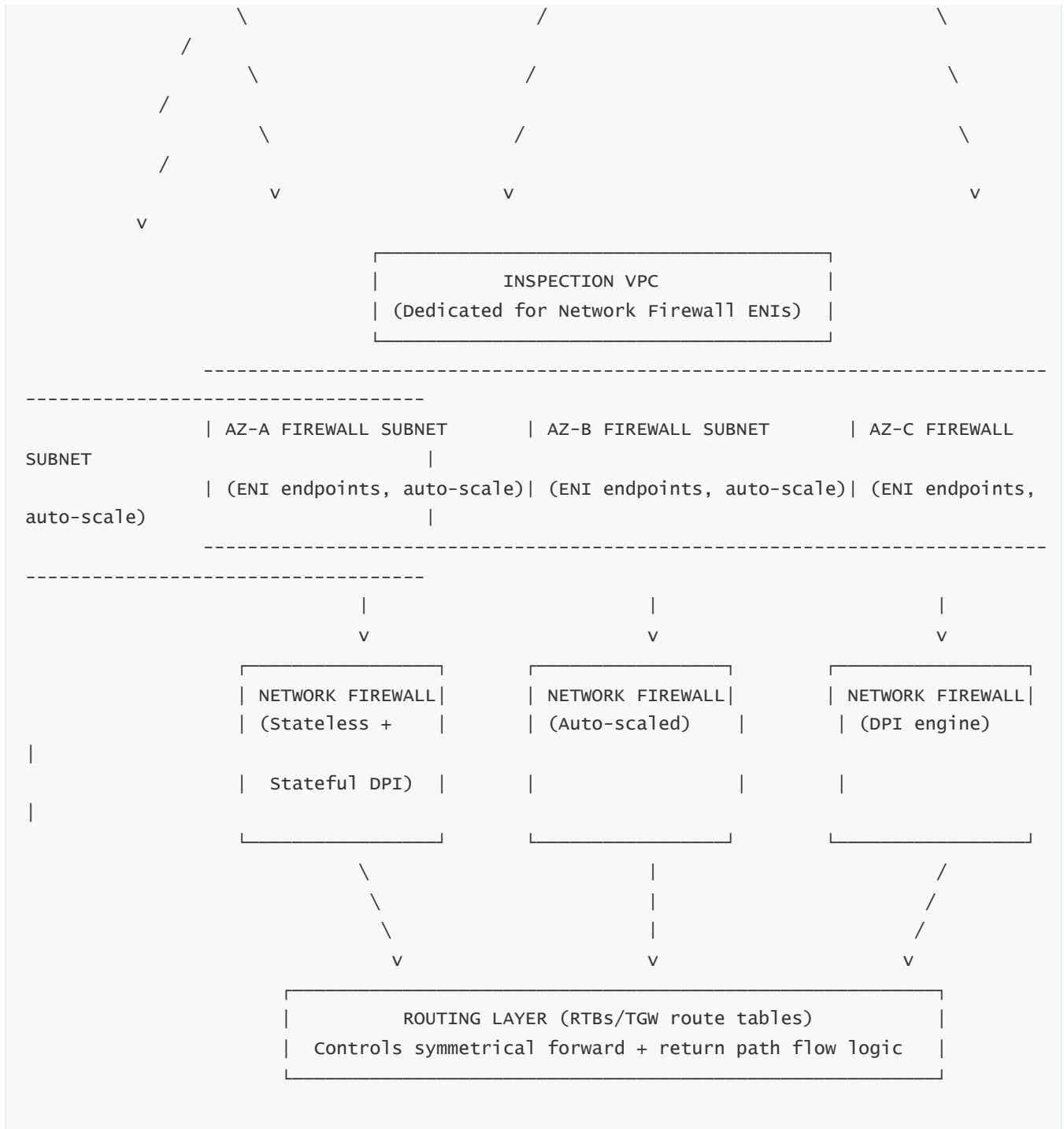
Below is **Mega-Diagram 1**.

=====

=====

## MEGA-DIAGRAM 1 — Full Deployment & Routing Architecture (VPC, TGW, Firewall)





## Explanation of Mega-Diagram 1

This diagram shows the **entire routing-centered deployment architecture** for AWS Network Firewall. The TGW sits at the center as the *routing brain* of the enterprise, aggregating all VPCs (application VPCs, ingress/inbound VPCs, egress/outbound VPCs) and on-prem VPN/DX paths. The Inspection VPC is a **dedicated security VPC** that contains firewall subnets in every AZ. Each firewall subnet contains multiple ENIs that represent **auto-scaling distributed firewall endpoints**.

Traffic from any source—application workloads, internet, or hybrid networks—enters the TGW, which forwards it to the Inspection VPC. Routing then forces traffic through the firewall endpoints, where stateless and stateful engines evaluate each packet. The routing layer then returns inspected traffic to TGW for final delivery.

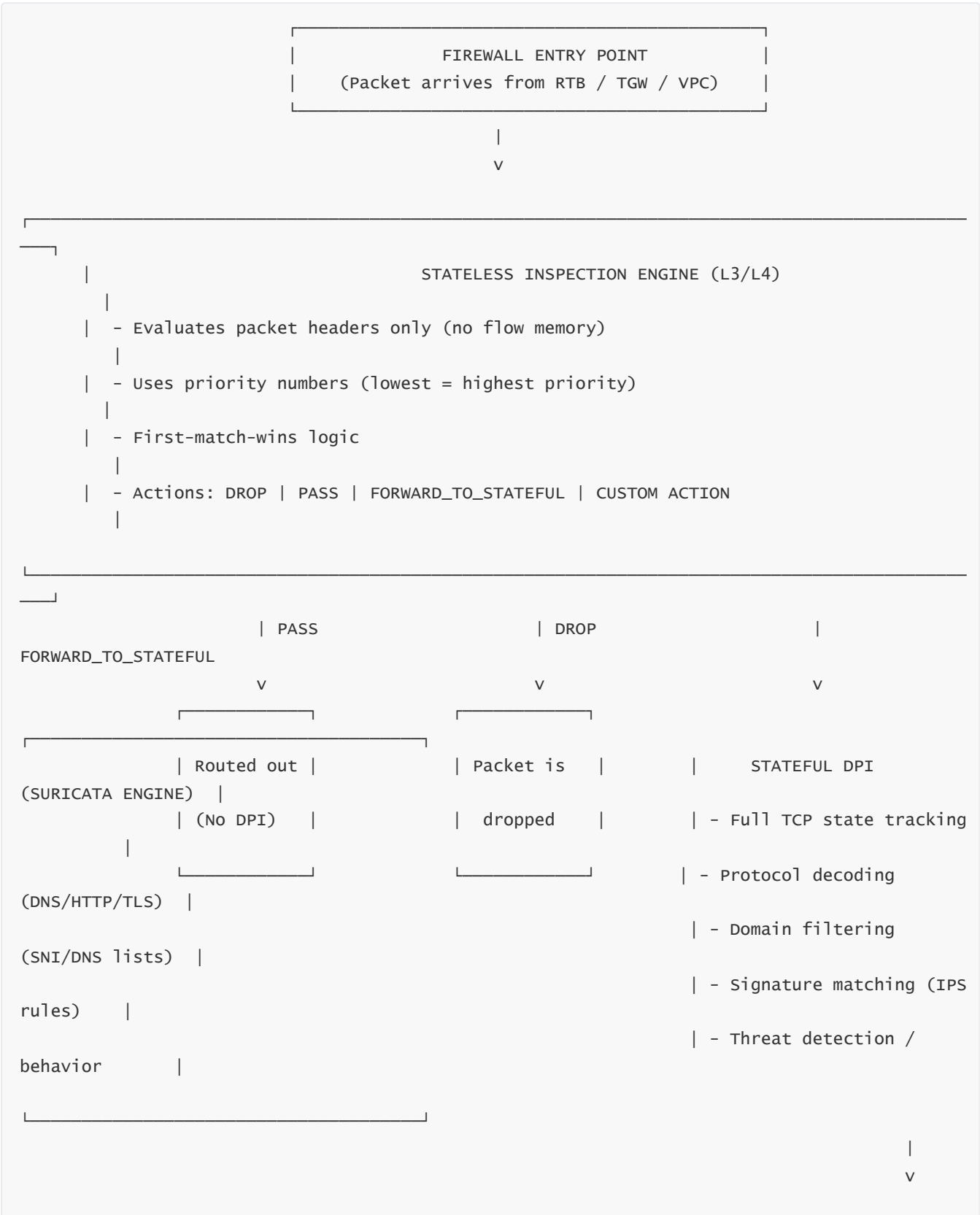
This achieves full north-south, east-west, hybrid, and inter-VPC security in a unified enterprise design.

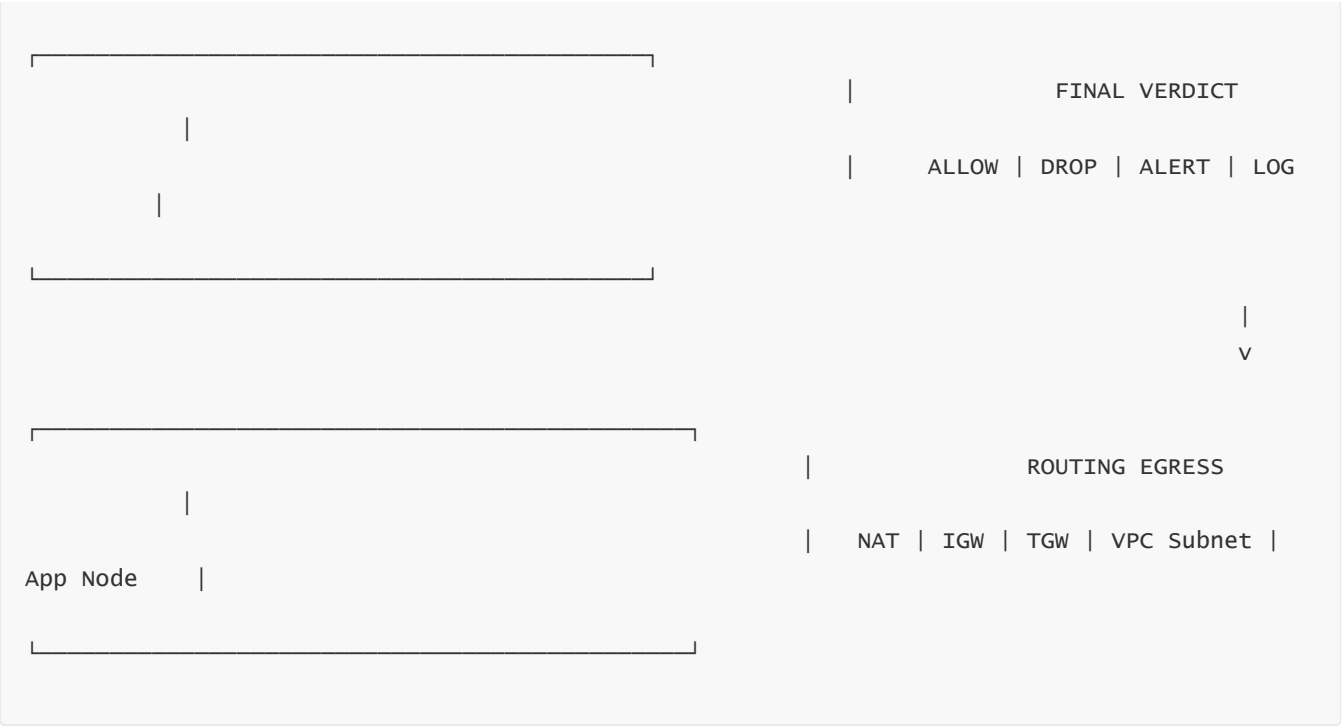


=====

=====

# MEGA-DIAGRAM 2 — Stateless + Stateful DPI Processing Pipeline (Suricata Internals)





## Explanation of Mega-Diagram 2

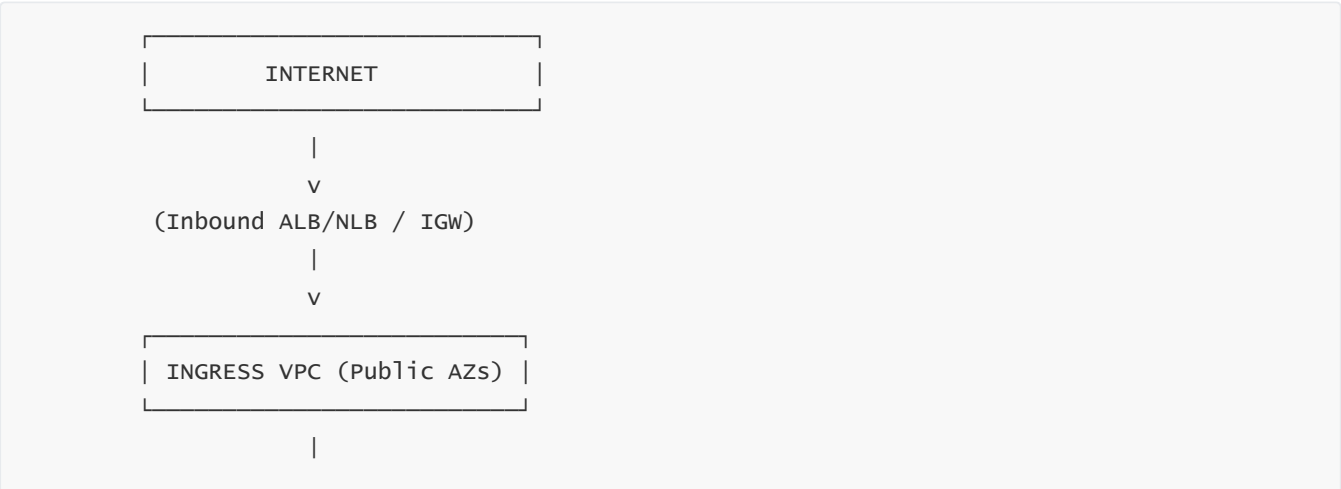
This diagram provides a complete view of the **inspection decision pipeline** inside AWS Network Firewall. Traffic enters through the stateless engine first, which decides early drops, pass-throughs, or whether deeper inspection is required. If forwarded to the stateful engine, Suricata performs full DPI, decoding application-layer protocols and applying domain filtering and IPS rules. The final verdict determines forwarding or dropping.

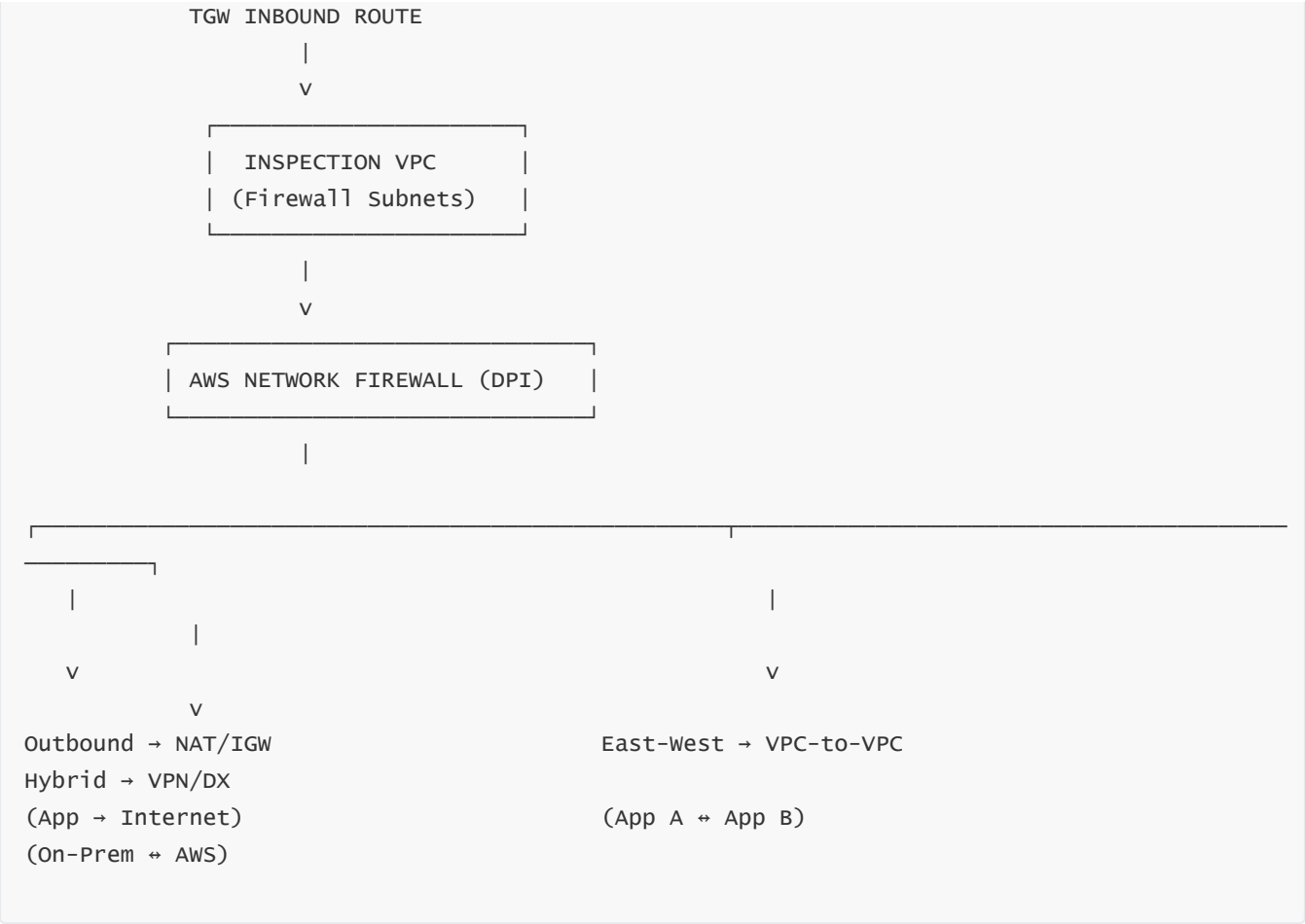
This pipeline defines how AWS Network Firewall transforms simple packet rules into enterprise-level detect/prevent capabilities.

=====

=====

## MEGA-DIAGRAM 3 — Traffic Flow: Inbound, Outbound, East-West, Hybrid





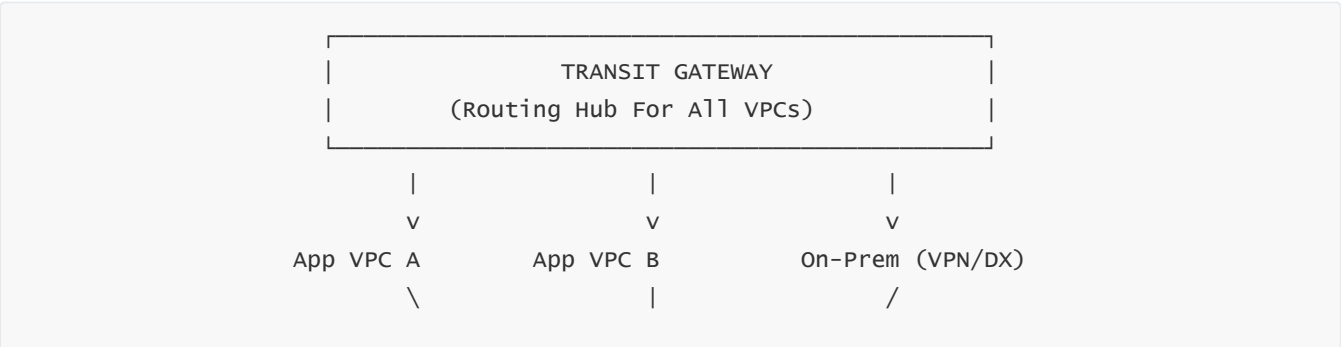
### Explanation of Mega-Diagram 3

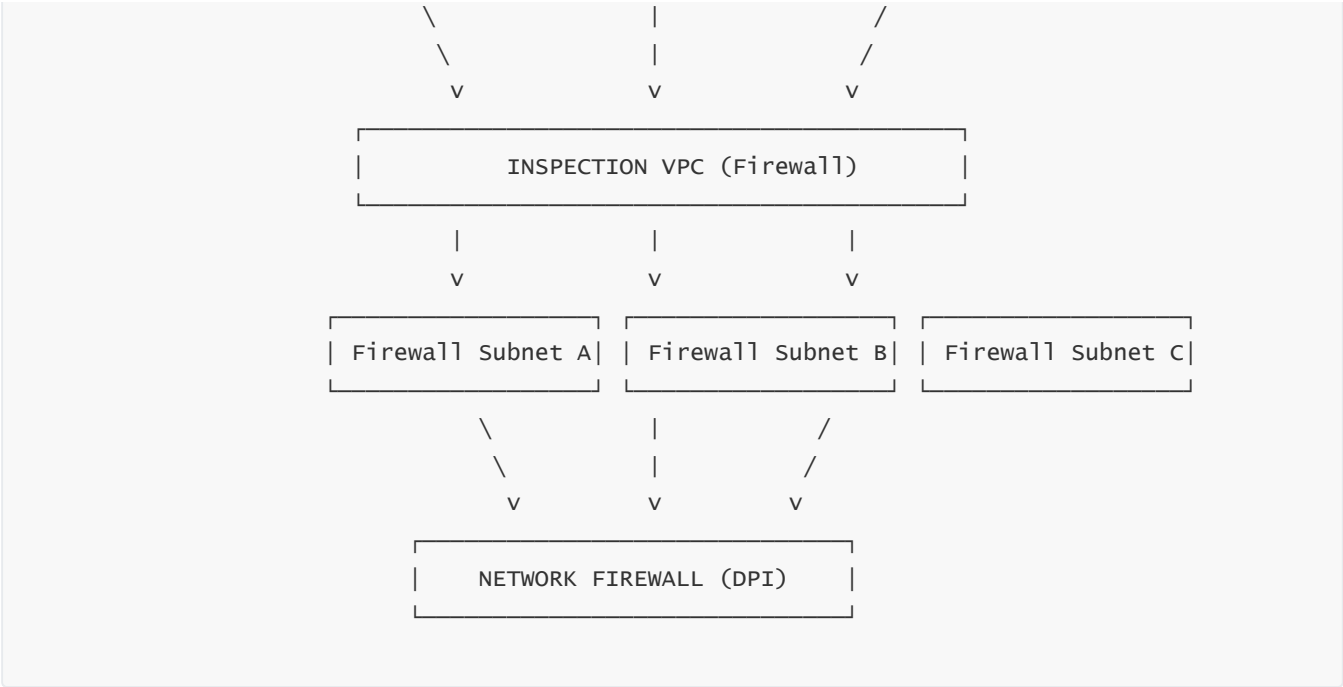
This diagram consolidates *all major traffic directions* into AWS Network Firewall. Inbound flows from the internet must pass through Ingress VPC → TGW → Firewall → Application VPCs. Outbound flows pass from app subnets → firewall → NAT → internet. East-west flows always pass through the TGW inline inspection path. Hybrid flows via VPN or Direct Connect must traverse the firewall before mixing with AWS workloads. This ensures that all traffic, regardless of source or destination, undergoes consistent inspection.

=====

=====

### MEGA-DIAGRAM 4 — TGW Inline Inspection (Enterprise Mandatory Path)





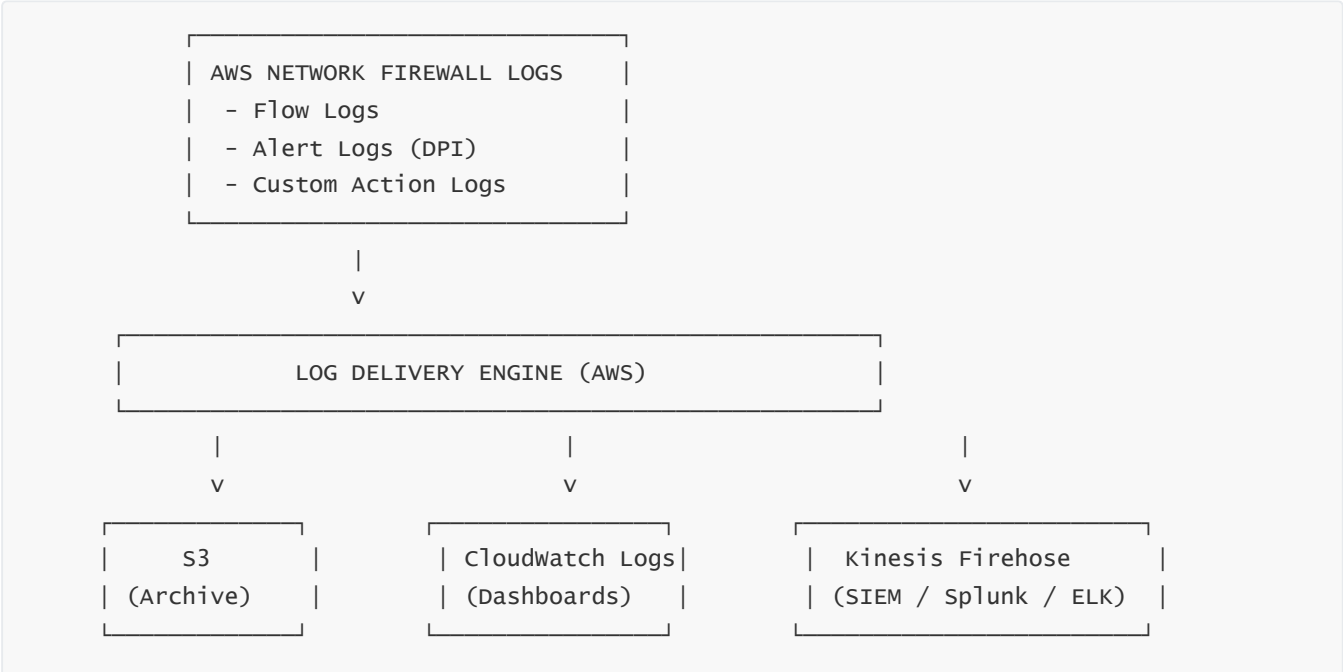
Explanation

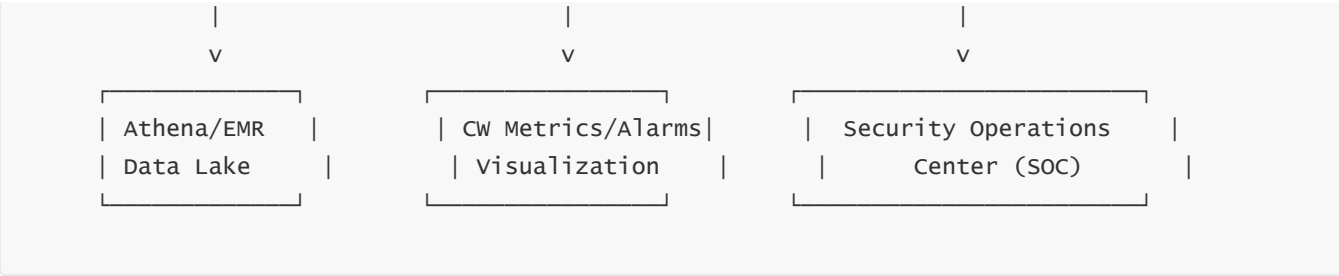
TGW is the backbone of enterprise-scale security. All VPC attachments route traffic to the Inspection VPC. The firewall subnets contain distributed endpoint nodes in each AZ. Traffic always returns to TGW after inspection. This inline service insertion pattern ensures no VPC can communicate with any other VPC, or with on-prem, without passing through the firewall.

=====

=====

MEGA-DIAGRAM 5 — Logging, SIEM, Analytics Pipeline



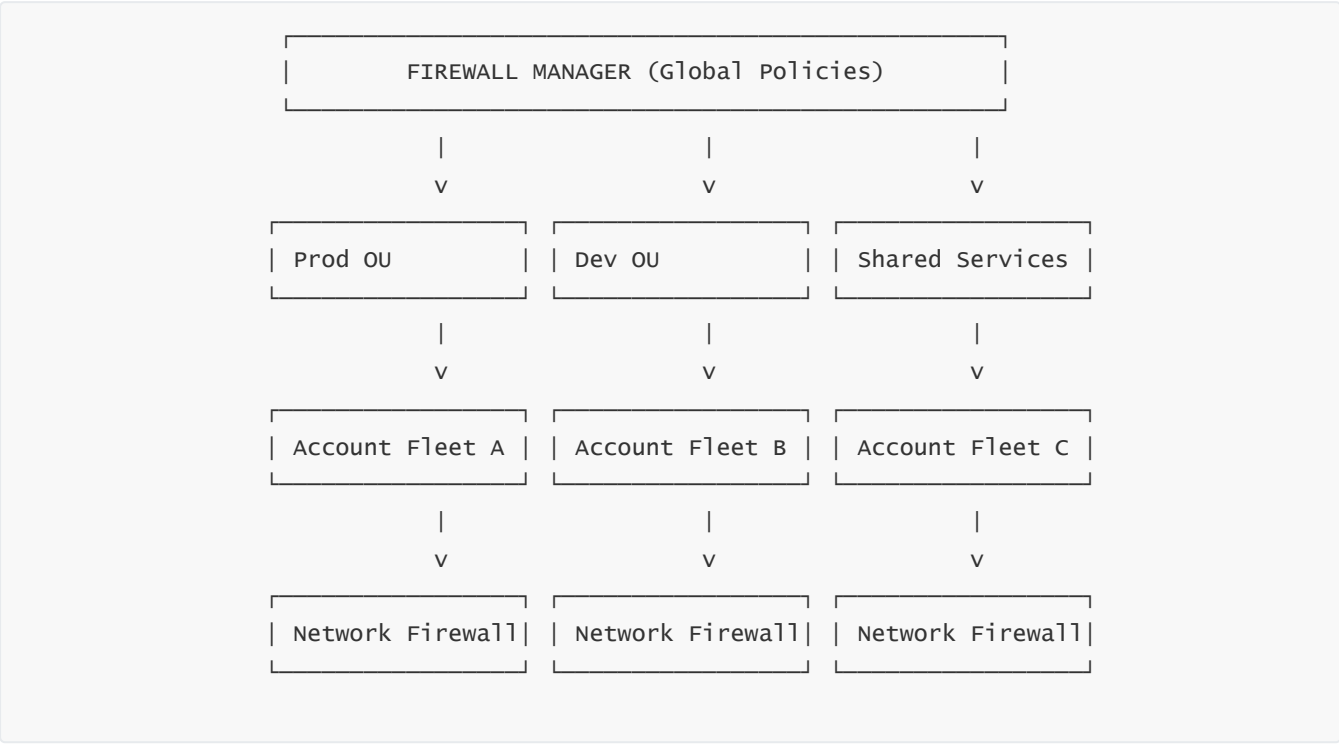


**Explanation**

Firewall logs feed S3 for long-term storage, CloudWatch for real-time dashboards, and Kinesis Firehose for SIEM ingestion. This creates a multilayer analytics pipeline for compliance, threat detection, and incident response.



**MEGA-DIAGRAM 6 — Governance Using Firewall Manager**



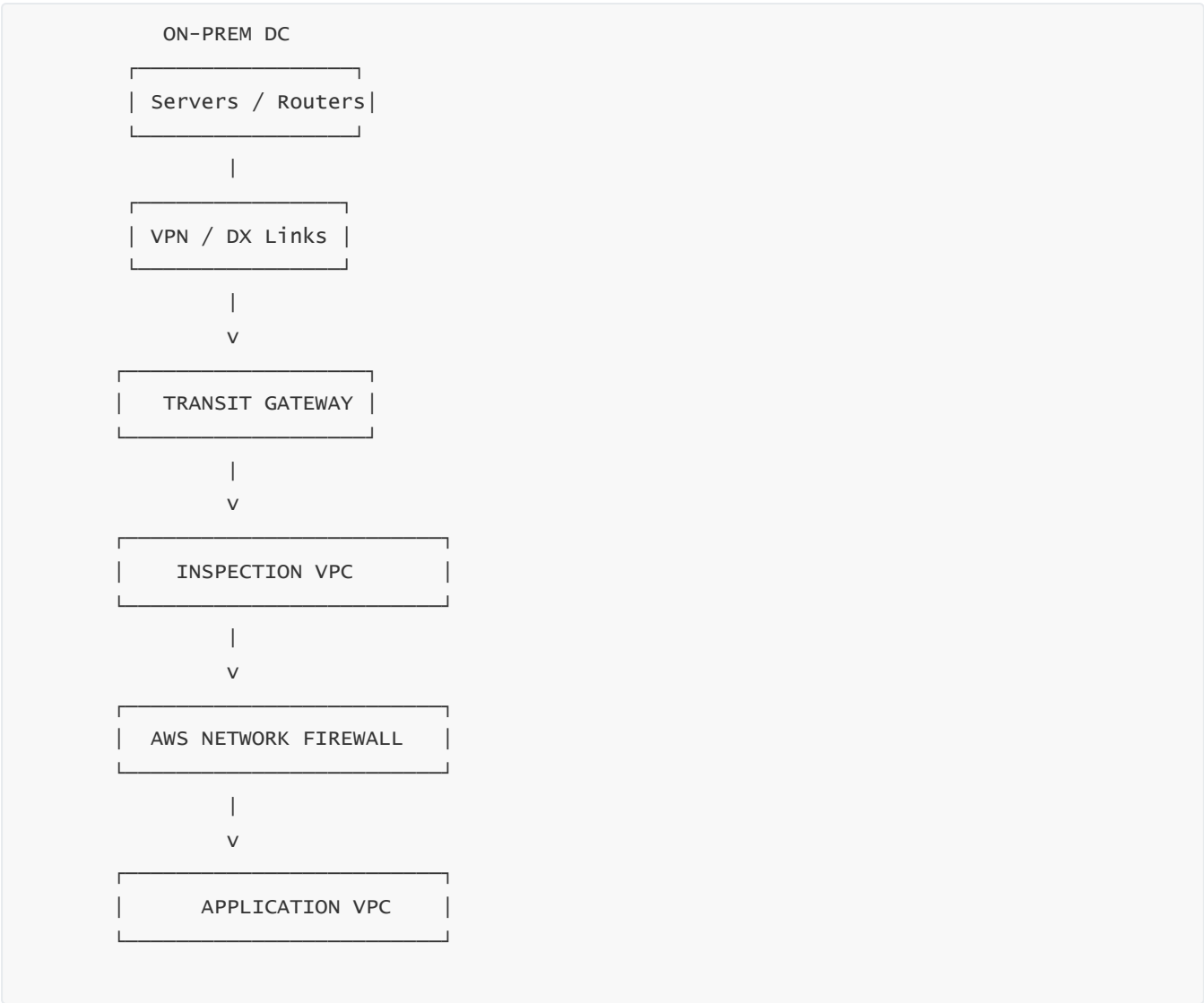
**Explanation**

Firewall Manager enforces mandatory rule groups, mandatory logging, and uniform policies across all accounts and Regions. It auto-remediates drift and automatically deploys firewalls into new VPCs.

=====

=====

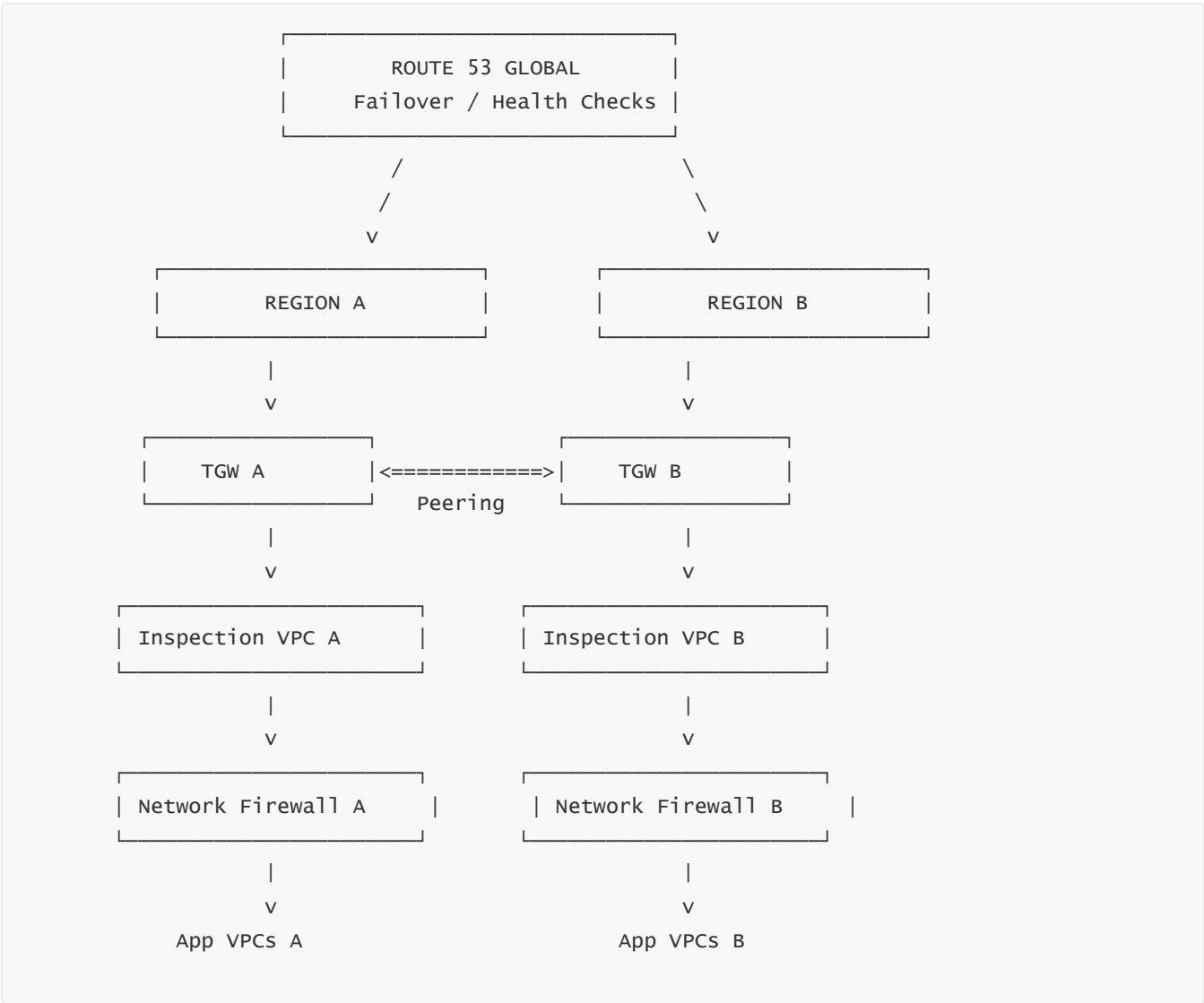
# MEGA-DIAGRAM 7 — Hybrid Network Security (VPN/DX to AWS Firewall)



## Explanation

All hybrid (on-prem → AWS) traffic enters via TGW and must be routed through the firewall before reaching any application VPC.

# MEGA-DIAGRAM 8 — Multi-Region Disaster Recovery Architecture

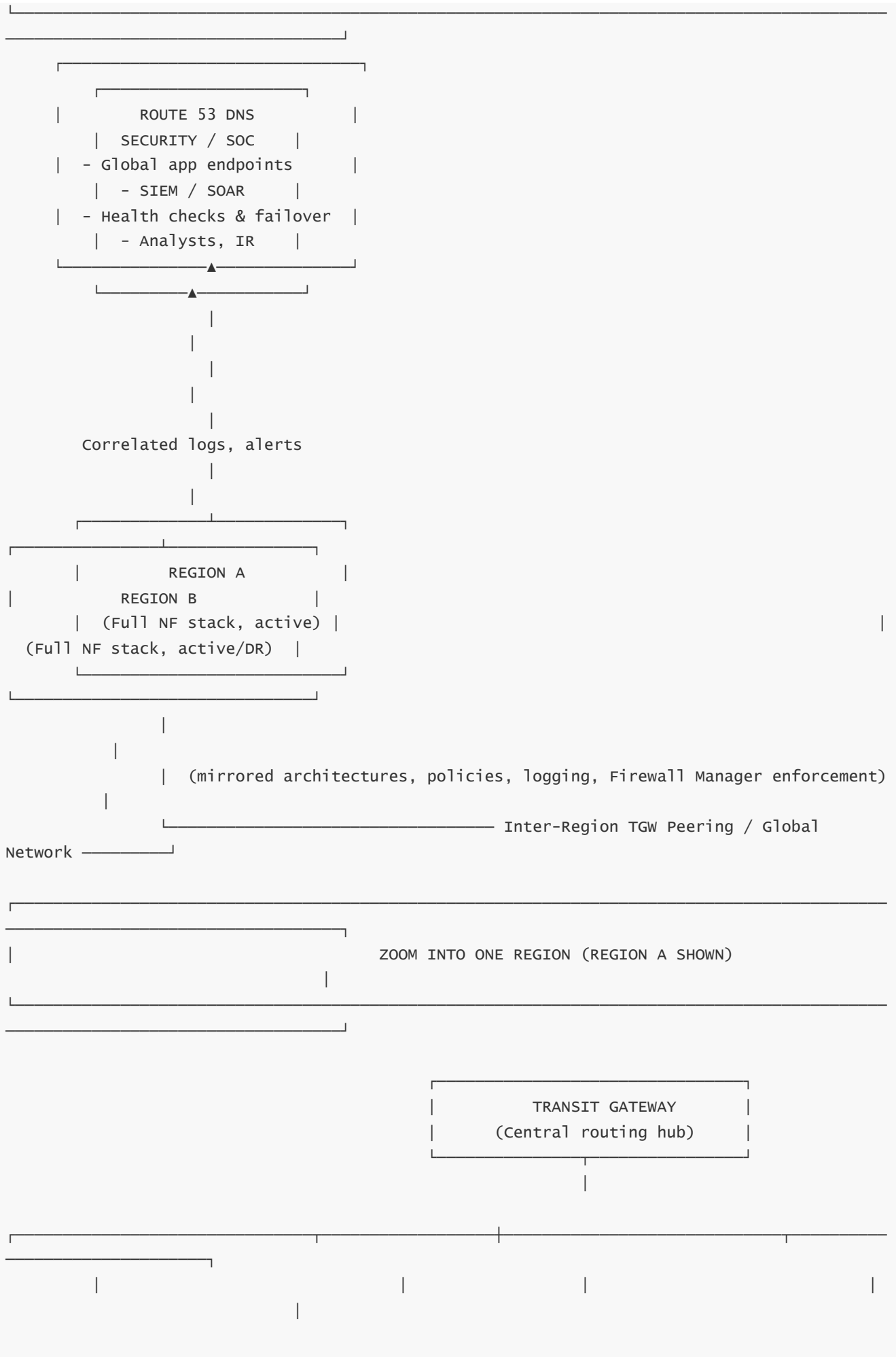


## Explanation

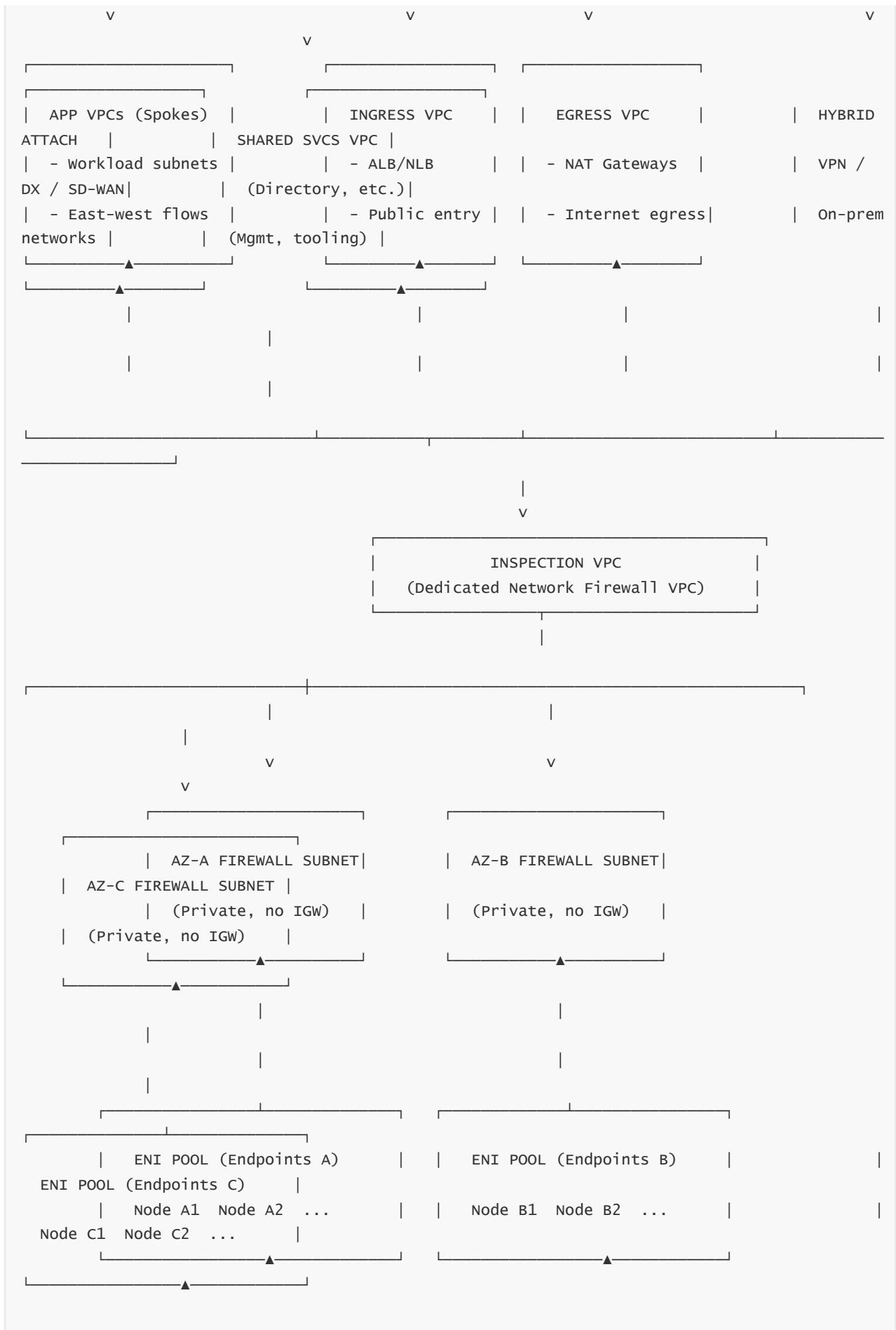
Each Region has its own full firewall stack. TGW peering enables cross-Region flows. Route 53 failover switches application + inspection paths automatically.

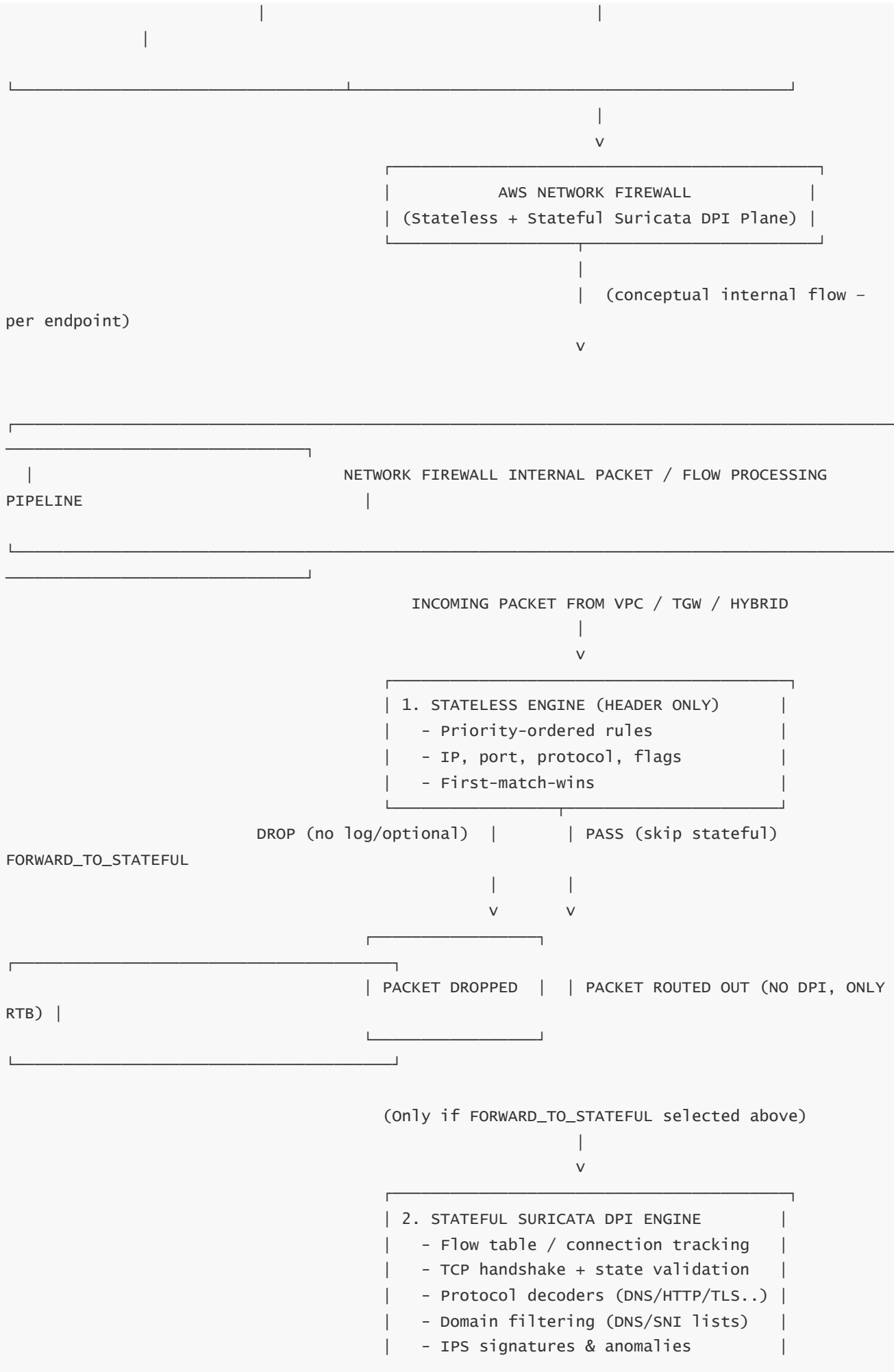
GLOBAL AWS NETWORK FIREWALL BLUEPRINT  
(Multi-Region, Multi-VPC, Hybrid, DPI, Logging,  
Governance)

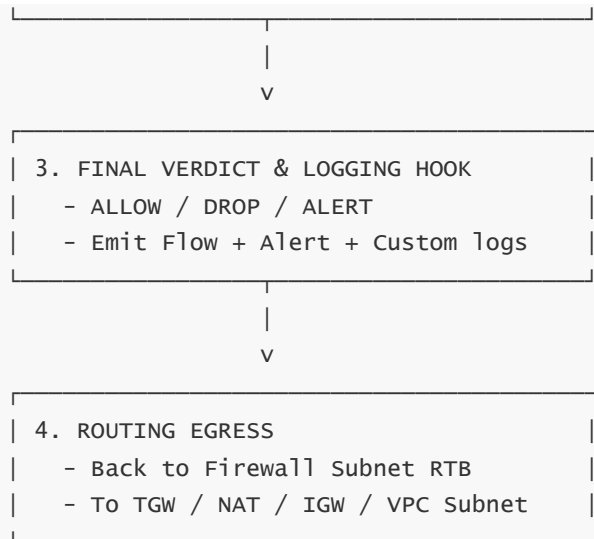
GLOBAL ENTRY & MULTI-REGION LAYER











DIRECTIONAL TRAFFIC PATHS (NORTH-SOUTH, EAST-WEST, HYBRID) THROUGH SAME FABRIC

INBOUND (Internet → App)  
EAST-WEST (VPC↔VPC & Subnet↔Subnet)

OUTBOUND (App → Internet)

Internet → Ingress VPC (ALB)  
App VPC A → TGW → Inspection VPC  
→ TGW → Inspection VPC  
→ Network Firewall → TGW → App VPC B  
→ Network Firewall  
(also intra-VPC east-west via RTB → FW)  
→ TGW → App VPC private subnets  
(all symmetric, same AZ, same endpoint)

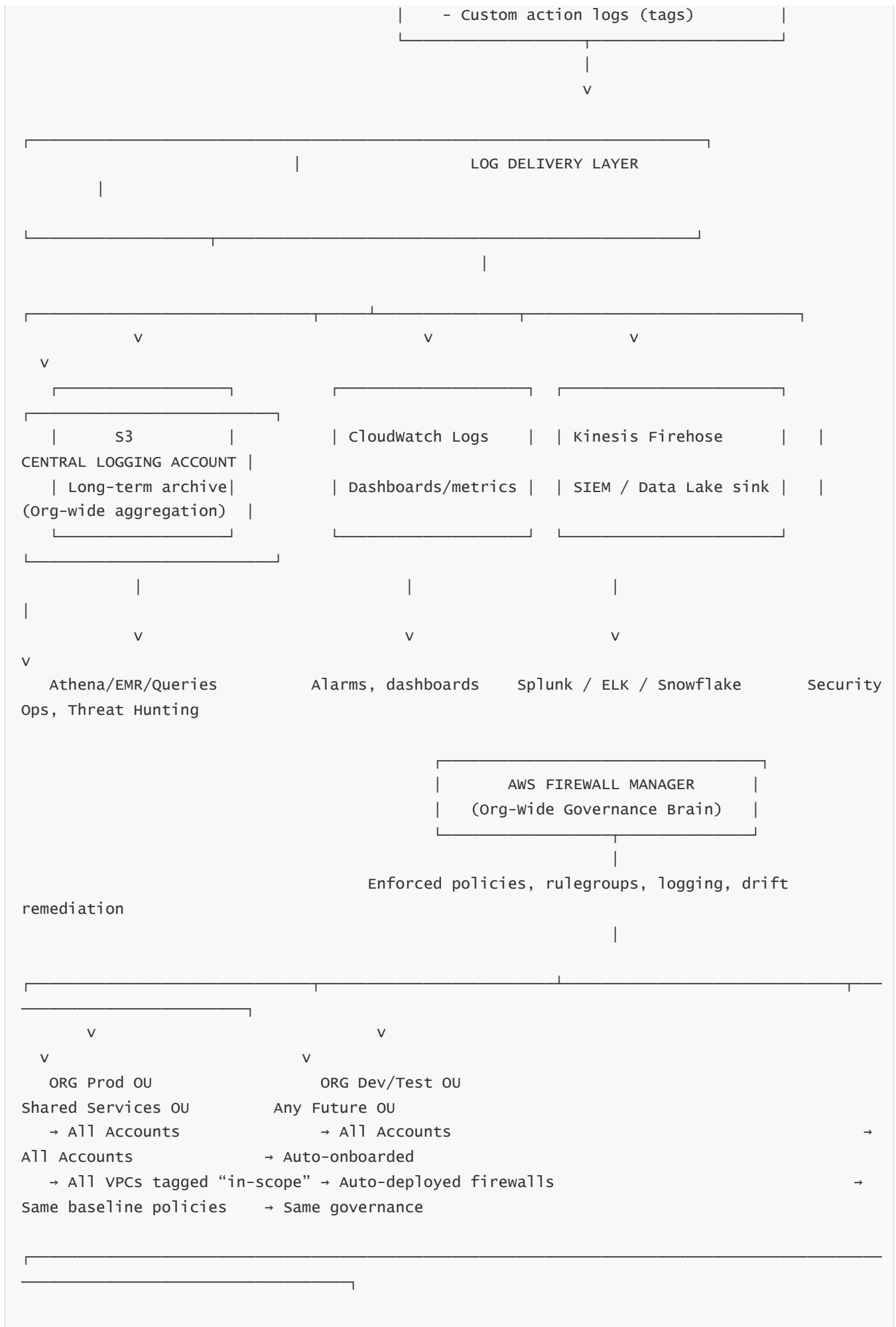
App Subnet → VPC Route Table  
→ Firewall Subnet → Network Firewall  
→ NAT in Egress VPC → IGW  
→ Internet

HYBRID (On-Prem ↔ AWS)

On-Prem DC → VPN/DX → TGW → Inspection VPC → Network Firewall → TGW → Application VPC  
Return path follows the exact reverse stepping through the same firewall fabric for stateful symmetry.

LOGGING, ANALYTICS, AND GOVERNANCE (SHARED ACROSS ALL REGIONS)

AWS NETWORK FIREWALL LOGS  
- Flow logs (behavior)  
- Alert logs (DPI / Suricata)



On-premises networks in multiple physical data centers connect through VPN and Direct Connect into TGW in each Region.

Each Region runs its own Inspection VPC and Network Firewall stack, with rule groups and policies replicated via IaC and Firewall Manager.

Route 53 and TGW peering work together with application failover to ensure that when workloads move between Regions, traffic still passes through a local

firewall fabric before reaching any workloads, maintaining identical security posture in all active Regions.

Now a single narrative tying everything together:

– At the very top of this unified blueprint we see the global layer where Route 53 and your SOC or security operations center coordinate multi-Region traffic and incident response. Route 53 uses health checks and DNS failover to steer user requests into the currently healthy Region, while your SOC ingests logs and alerts from every Region and every Network Firewall deployment into a global SIEM and investigation workflow. This guarantees that security operations and application routing are aligned at the global level.

– The next major layer zooms into a single Region such as Region A. Inside this Region, the Transit Gateway acts as the central routing hub for every VPC and every hybrid connection. Application VPCs, the Ingress VPC, the Egress VPC, hybrid VPN or Direct Connect attachments, and shared-services VPCs all attach to the TGW. No VPC talks directly to another VPC or to on-prem; everything rides through the TGW core, which then sends security-sensitive traffic into the Inspection VPC.

– The Inspection VPC is the heart of all network security in that Region. It contains only private subnets, specifically designed as firewall subnets in each Availability Zone. These firewall subnets never have an Internet Gateway; they are internal choke points. In each firewall subnet, AWS automatically creates and scales a pool of endpoint ENIs that represent multiple invisible firewall nodes. These nodes are distributed per AZ to provide local high availability and horizontal scalability, and all traffic that must be inspected is routed through these endpoints.

– Inside each endpoint, the Network Firewall inspection pipeline takes over. The first stage is the stateless engine, which evaluates packet headers only, according to priority-ordered rules. This engine makes extremely fast decisions about whether a packet should be dropped immediately, passed onward with no further inspection, or forwarded into the stateful DPI engine. Because the stateless layer does not maintain flow state, it can handle large volumes of simple allow/deny decisions and acts as the first guardrail for unwanted protocols, ports, and sources.

– When packets are forwarded to the stateful engine, Suricata performs deep packet inspection. At this stage the firewall tracks flows, validates TCP handshakes and connection state, decodes higher-layer protocols such as DNS, HTTP metadata, SMB, TLS handshakes, and more, and compares traffic against IPS signatures and domain-filtering lists. Here the firewall detects malware callbacks, C2 traffic, protocol abuse, and internal lateral-movement behavior. The engine then produces a final verdict, which may allow the packet, drop it, or simply alert while still forwarding, and at the same time emits detailed logs capturing flow and threat information.

– After inspection, the packet returns to the routing layer. If it is outbound traffic, the firewall subnet route table sends it to the Egress VPC NAT gateways and then out to the internet. If it is inbound traffic, it is routed through TGW from the Ingress VPC to the appropriate application VPC private subnets. If it is east-west inter-VPC traffic, TGW receives the packet from the firewall and forwards it to the destination VPC. If it is hybrid traffic, TGW directs it towards the VPN or Direct Connect attachment for return to on-premises. In every case, the design ensures that the forward path and return path both traverse the same inspection fabric in the same AZ, which is mandatory for stateful consistency.

– The diagram then consolidates the directional paths logically. Inbound flows always traverse Internet → Ingress VPC → TGW → Inspection VPC → Firewall → TGW → private subnets. Outbound flows always traverse Application Subnets → Firewall Subnets → Network Firewall → NAT in the Egress VPC → Internet. East-west inter-VPC flows always pass through TGW and the Inspection VPC. Hybrid flows always run from on-prem through VPN or Direct Connect into TGW, then into the Inspection VPC and through Network Firewall before ever reaching any AWS workload subnets.

– On the observability side, the blueprint shows Network Firewall logs feeding three destinations in parallel: S3 for long-term archival and offline analytics, CloudWatch Logs for real-time metrics, alarms, and dashboards, and Kinesis Firehose for streaming into SIEM and data-lake platforms. All of these feeds are centralized into a dedicated logging account at the organization level, so that security teams gain a unified view across all accounts and Regions. From there, query tools such as Athena and EMR, plus dashboards and SIEM correlation engines, provide full forensic, compliance, and threat-hunting capability.

– Governance is handled by AWS Firewall Manager at the organization level. The diagram shows Firewall Manager sitting above all organizational units and accounts, pushing mandatory firewall policies, rule groups, and logging requirements down into every Region and account. Firewall Manager ensures that every in-scope VPC has a firewall, that mandatory stateless and stateful rule groups are applied, that logging is enabled and pointed to the correct destinations, and that configuration drift is automatically remediated. This prevents locally managed teams from weakening security controls or bypassing inspection paths.

– Finally, the lower part of the blueprint extends the model to hybrid and multi-Region designs. On-premises data centers connect through VPN or Direct Connect, but those connections always terminate at TGW and are routed through the Inspection VPC and Network Firewall before reaching any application VPCs. Multiple Regions each implement the same Inspection VPC and Network Firewall pattern, with Transit Gateways in each Region connected over inter-Region peering. Route 53, together with TGW routing and replicated firewall policies, gives you a failover model where both applications and inspection layers are active and ready in every Region, ensuring that traffic is never allowed to bypass security during DR events or cross-Region migrations.

– Put together, this single mega-diagram is the complete mental model for AWS Network Firewall as an enterprise-grade, multi-Region, hybrid, DPI-powered security fabric. It shows how routing, inspection, logging, and governance all fit together to create a consistent, scalable security perimeter for every packet entering, leaving, or traversing your AWS environment.