Security Close to the Applications

AVIATRIX DISTRIBUTED CLOUD FIREWALL

ACE Solutions Architecture Team

NIST Tenets Covered

This module will cover two tenets of NIST Zero-Trust Architecture (ZTA)

1. Security Close to the Applications
2. Global, Dynamic and Centralized Policy Model

Related Aviatrix Features

- Aviatrix Distributed Cloud Firewall
- Network Segmentation
- Micro-Segmentation
- ThreatIQ / ThreatGuard
- GeoBlocking
- URL Filtering / Internet Egress Traffic Filtering
- Centralized Policy Engine

Tenet from NIST Publication 800-207 - Zero Trust Architecture (ZTA)

Assets and traffic moving between enterprise and non-enterprise infrastructure should have a consistent security policy and posture. Workloads should retain their security posture when moving to or from enterprise-owned infrastructure. This includes devices that move from enterprise networks to non-enterprise networks. This also includes workloads migrating from on-premises data centers to non-enterprise cloud instances.

Tenet from NIST Publication 800-207 - Zero Trust Architecture (ZTA)

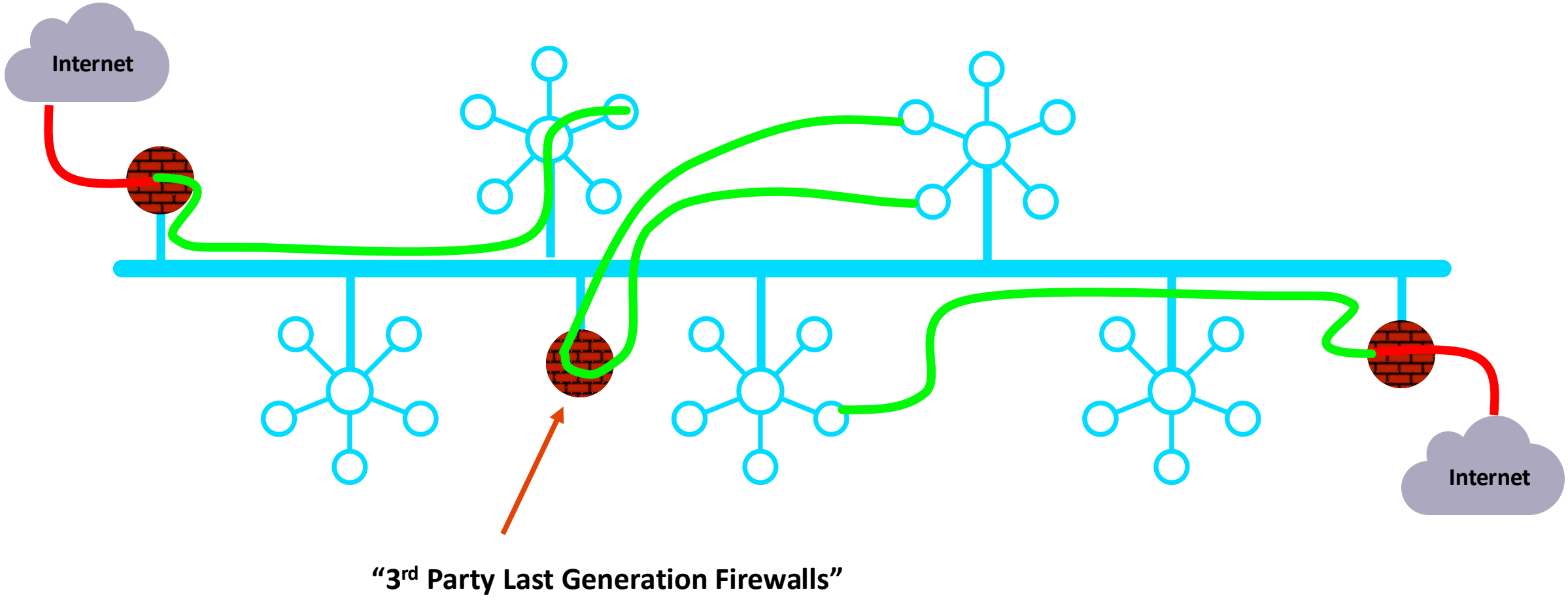
Access to resources is determined by dynamic policy—including the observable state of client identity, application/service, and the requesting asset—and may include other behavioral and environmental attributes.

Use Cases:

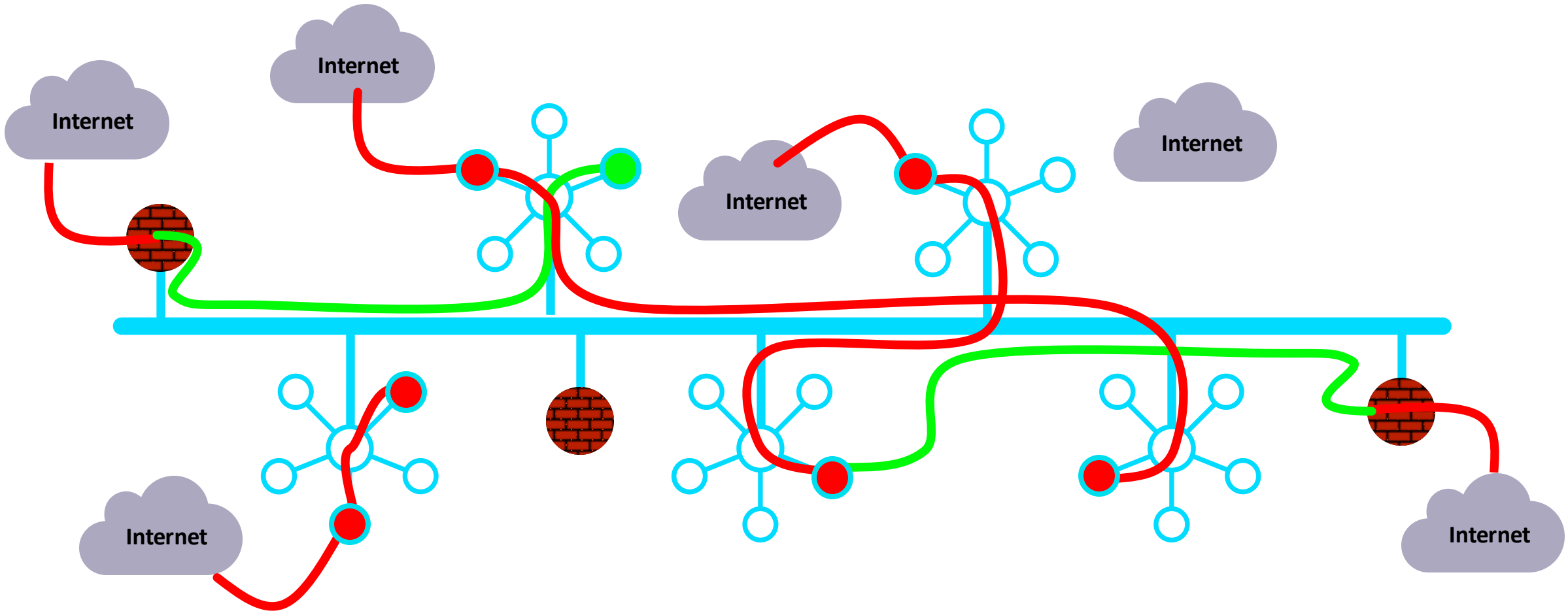
Zero Trust Network Access (Cloud Firewalling	**Secure B2B Connectivity**	**Secure High-Performance Data Connectivity for LLMS**	**Secure High-Performance Datacenter Edge**	**Cloud Visibility and Tooling**

aviatrix

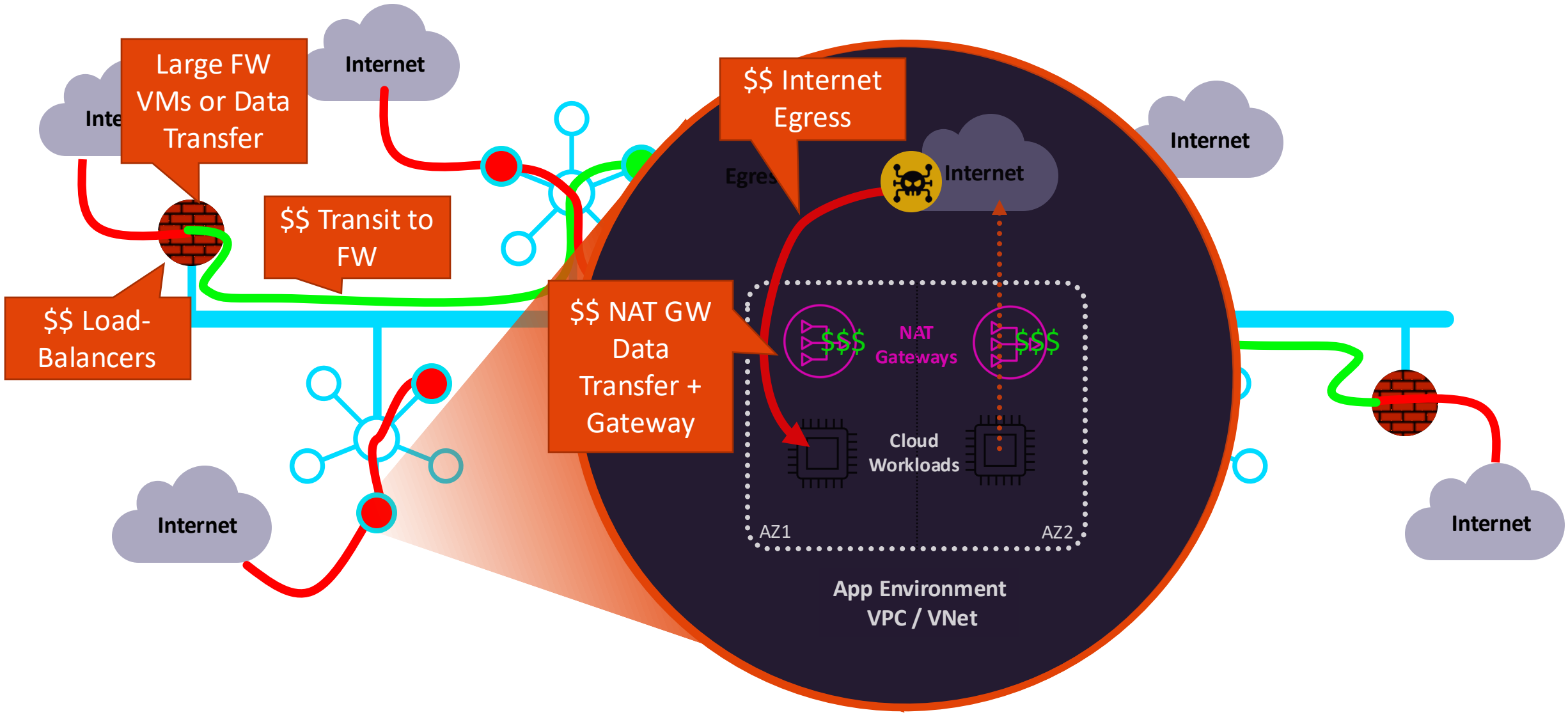
As Architected with Lift-and-Shift, Bolt-on, Data Center Era Products…



In Reality…

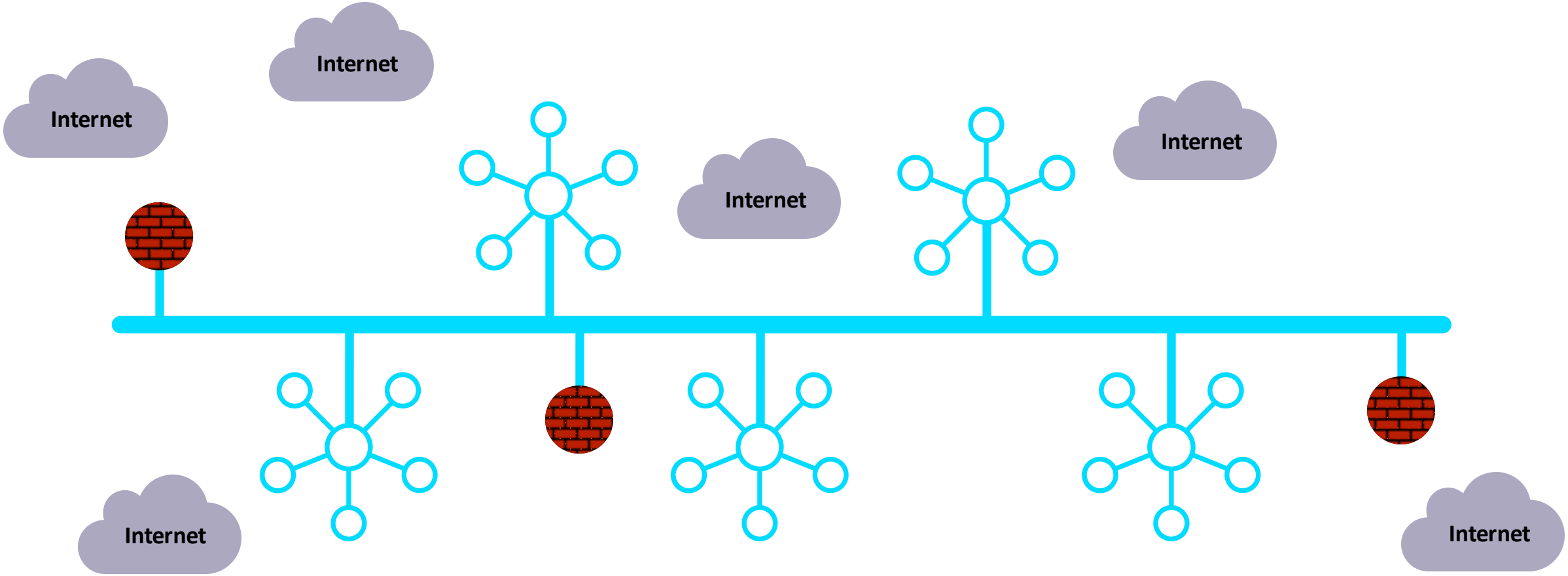

Internet
Internet
Internet
Internet
Internet
Internet
Internet

This is bad! Expensive and Lacks Enterprise-Grade Security

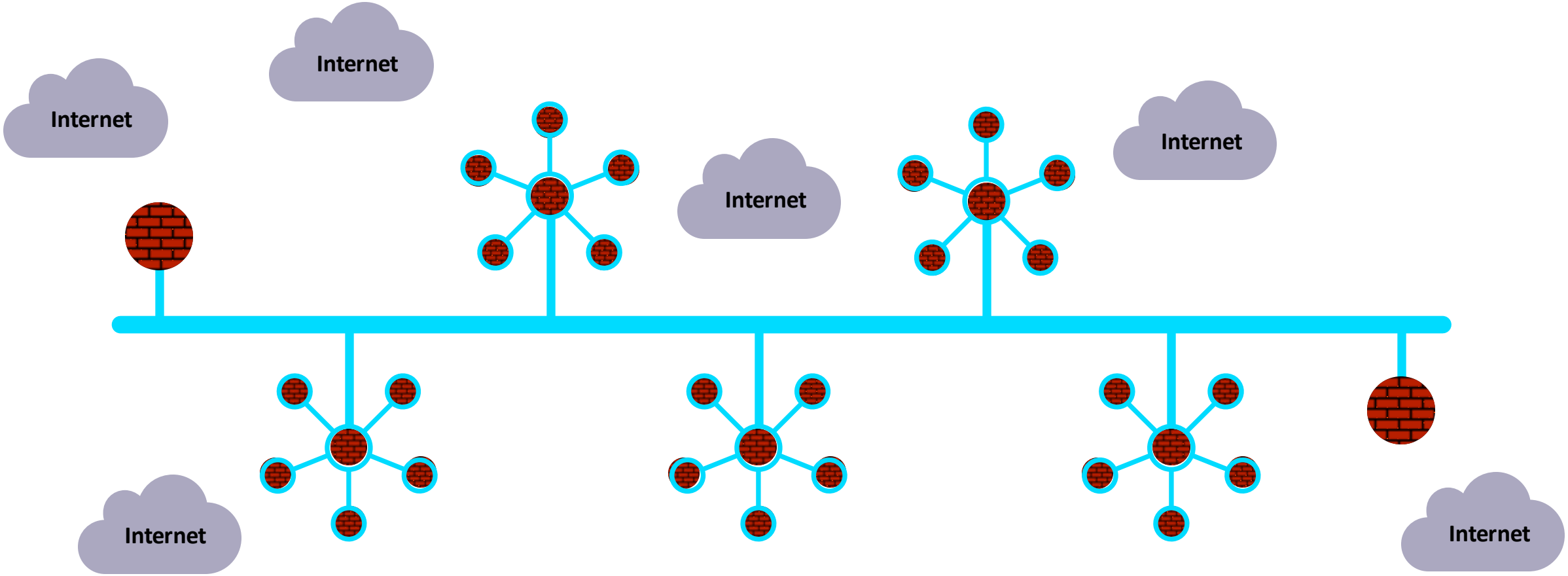


aviatrix

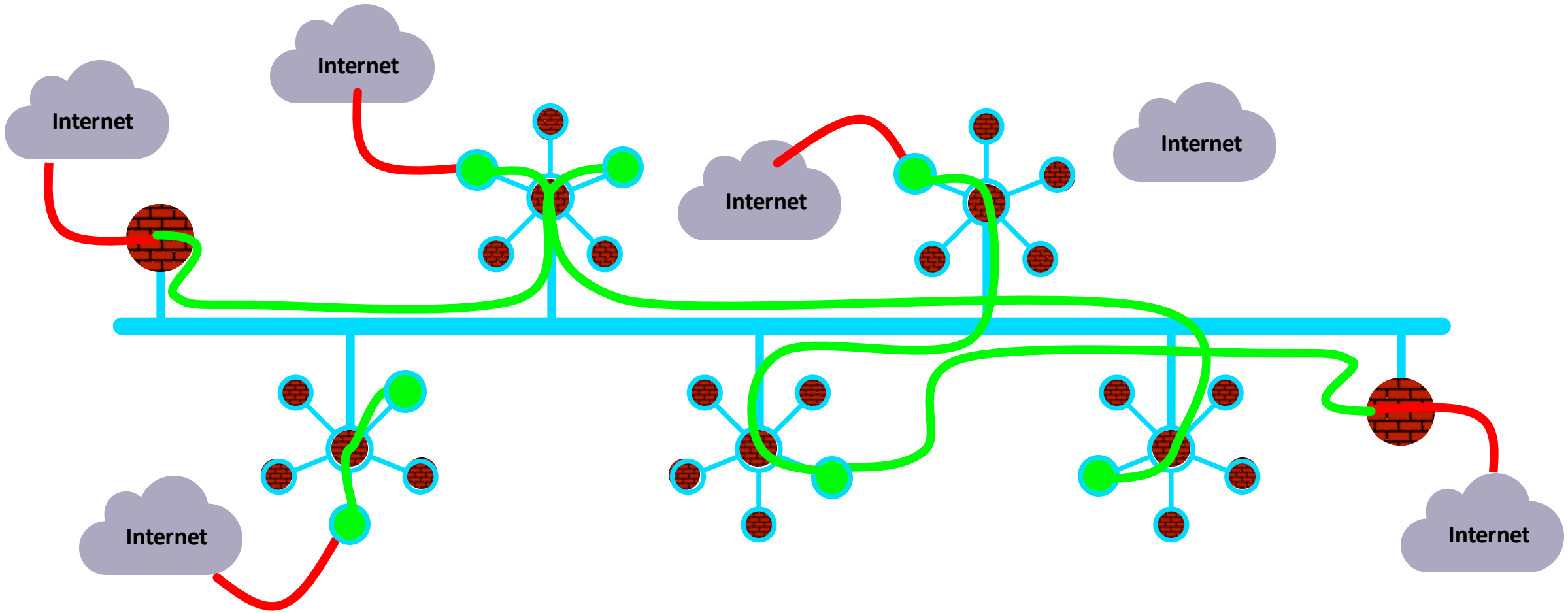
What If…


Internet
Internet
Internet
Internet
Internet
Internet
Internet

Firewalling Functions were Embedded in the Cloud Network Everywhere…



Centrally Managed, with Distributed Inspection & Enforcement…



Aviatrix Distributed Cloud Firewall

Aviatrix Controller
Global Policy Orchestrator

Terraform

Aviatrix CoPilot
Policy authoring, visibility, logging, alerting, and troubleshooting

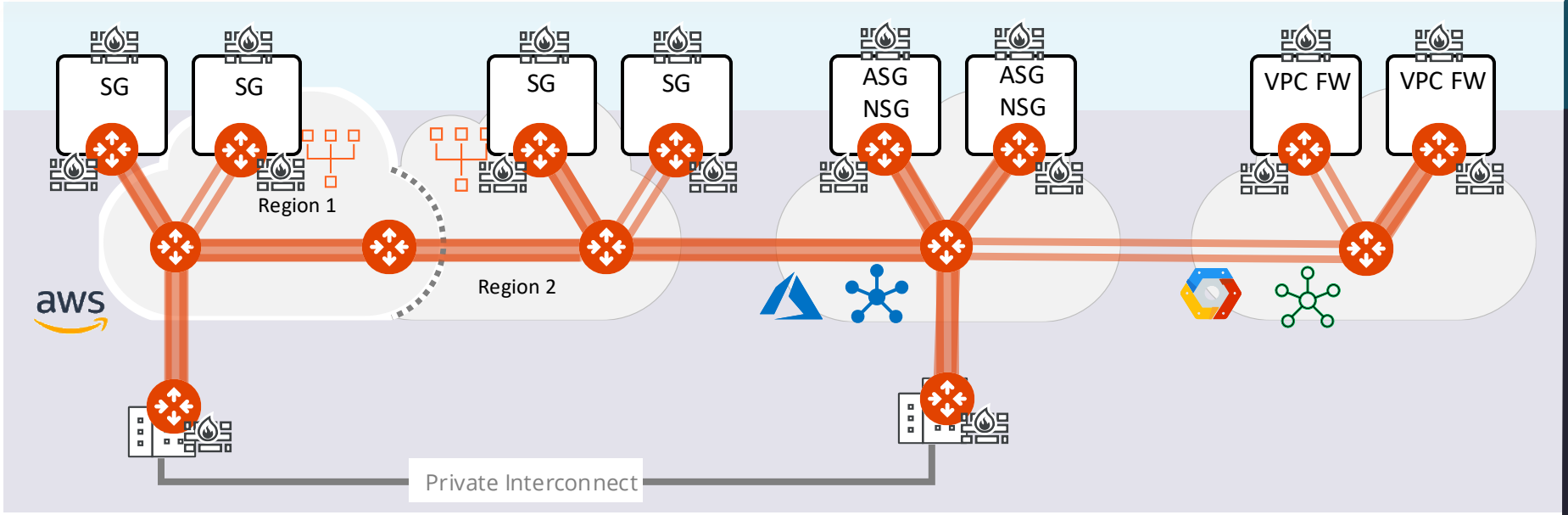
Threat Intel

Logging to SIEM

splunk>

Policy Intent Examples

- Deny workloads tagged as Prod from talking to Dev
- Allow outbound web traffic for IaaS subnet to *.microsoft.com
- Block inbound traffic from Russia
- Decrypt and inspect workloads tagged as PCI



Distributed Stateful L4 Security

Cloud Native Security Group Orchestration

- East-West Micro-Segmentation

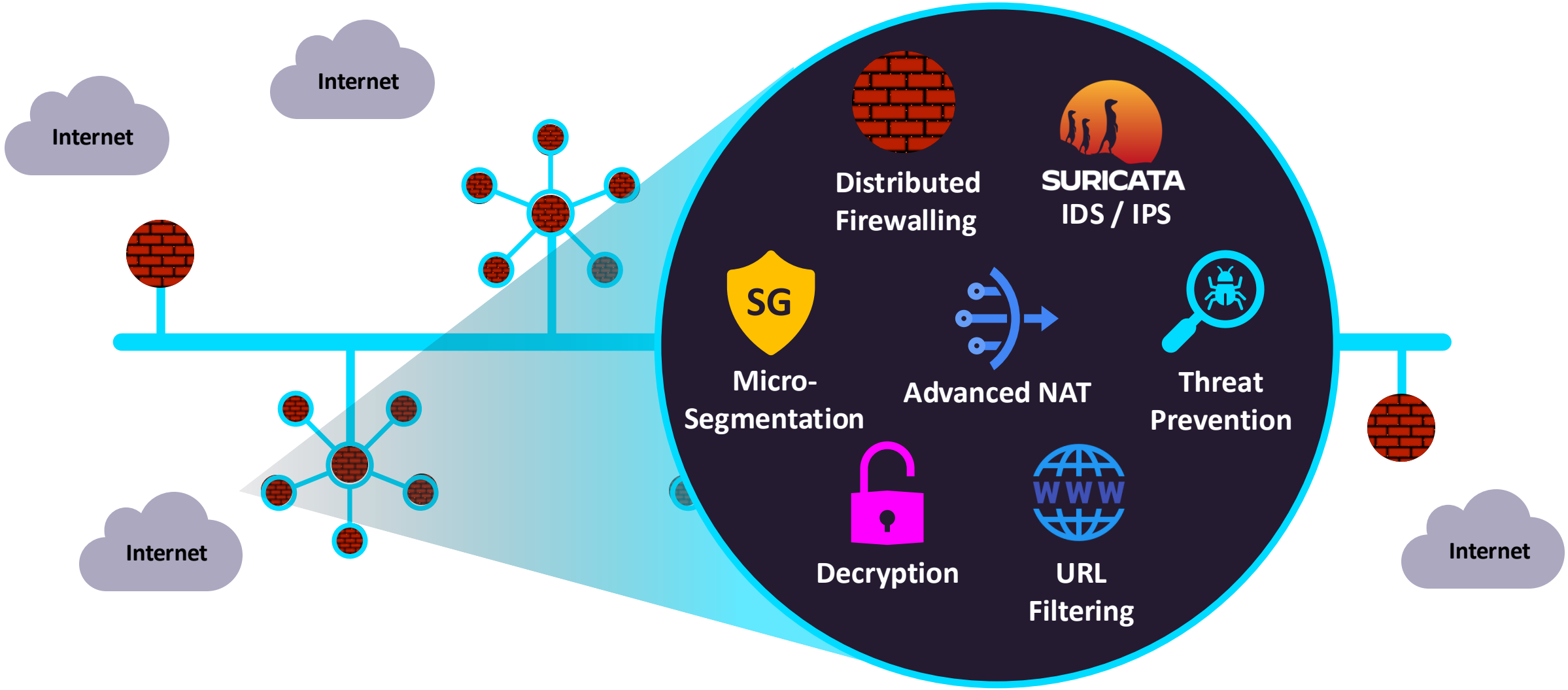
Distributed Stateful L4/L7 Security

Visibility and Enforcement in the Aviatrix Data Plane

- East-West L4 at scale
- Egress FQDN filtering
- Malicious IP/Threat Blocking
- Geo-blocking
- SSL Fingerprinting
- Decryption + IPS/IDS (Private Preview)
- Public Subnet Filtering
- Network Behavior Analytics

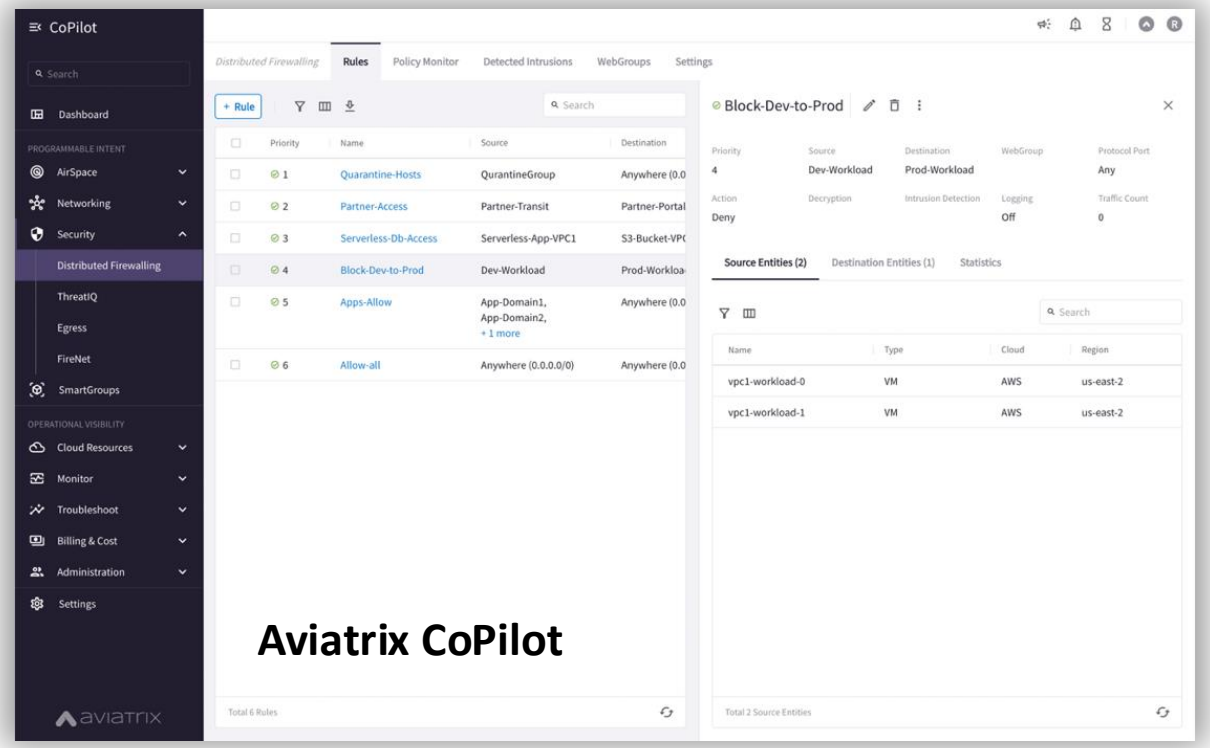
aviatrix

And, What If it was more than just firewalling…



Centralized Policy Creation Looked Like One Big Firewall…

Centralized Policy Creation → **Distributed Enforcement**



Aviatrix CoPilot

IDS / IPS

NSG

Micro-Segmentation

Threat Prevention

Advanced NAT

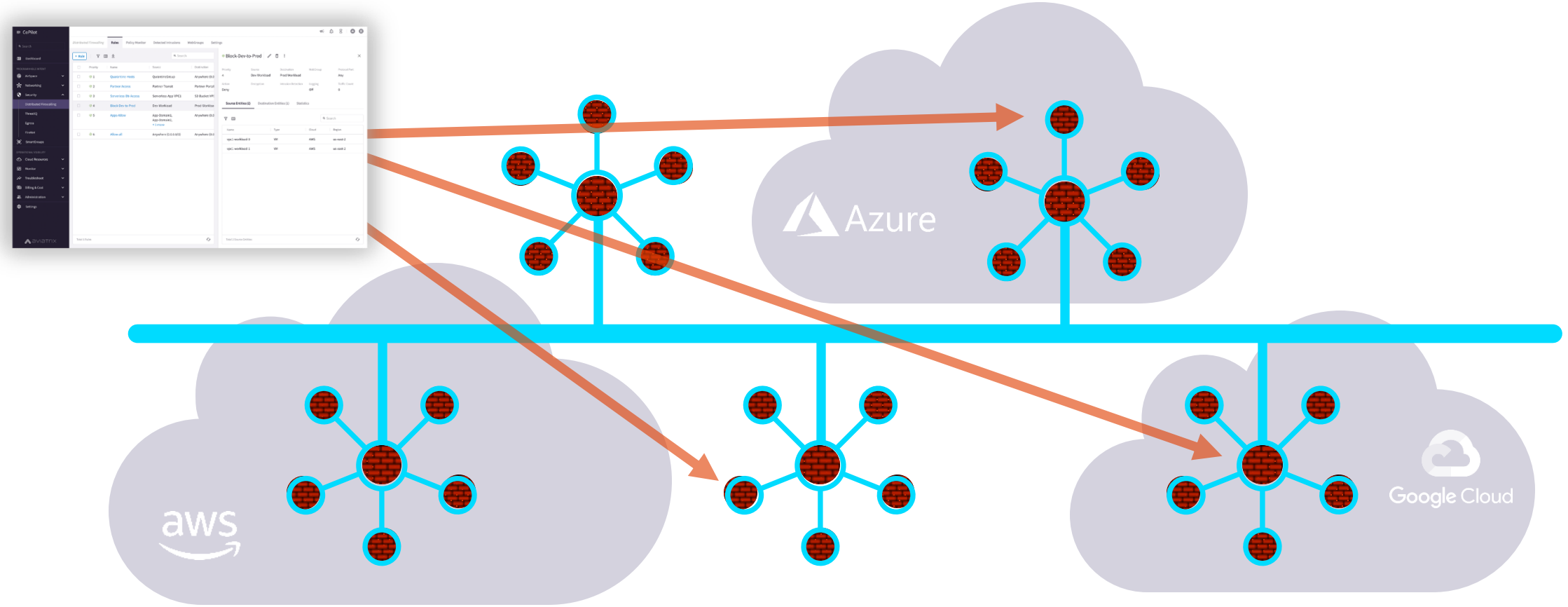
Distributed Firewalling

Decryption

WWW

URL Filtering

A Distributed Cloud Firewall…



Where and How Policies Are Enforced Is Abstracted…

Centralized Vs Distributed Firewalling Architecture

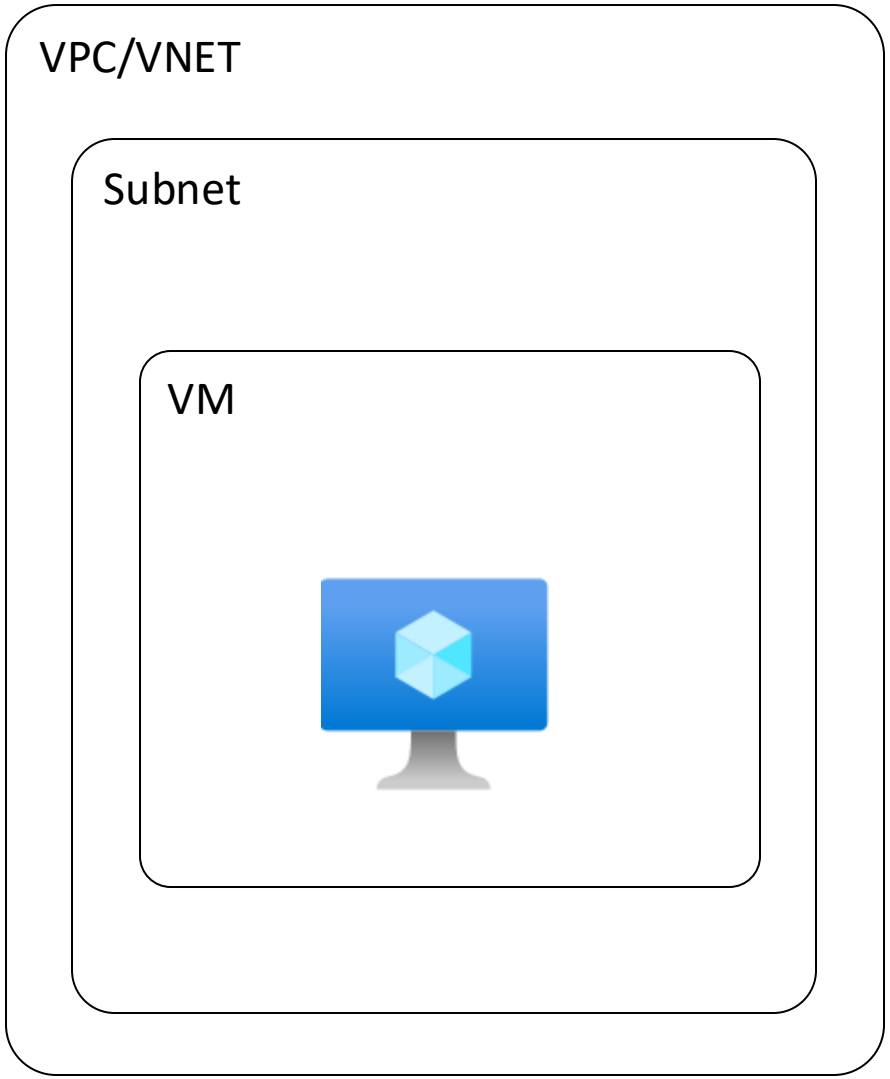
Aspect	Cloud Centralized Firewall Model	Aviatrix Distributed Cloud Firewall
Blast Radius (Fault Tolerance and Resilience)	Single point of failure risks, although mitigated by redundant systems; still, failure can affect entire network traffic.	Enhanced fault tolerance by distributing firewalls, reducing impact of localized failures and increasing overall network resilience.
Performance	Potential latency issues due to traffic bottlenecks through a centralized point; performance can degrade under high load.	Optimized for low latency by distributing firewall capabilities close to application workloads, improving overall network performance.
Cost	Maintaining centralized Firewall architectures is increasingly expensive. Cloud providers and firewall vendors profit from costly data processing charges, oversized VMs, and pricey licenses.	DCF's cloud-native design enables organizations to sidestep the costly data processing fees, oversized virtual machines, and exorbitant licensing often encountered with traditional centralized models.
Deployment Speed	Longer deployment times due to hardware installations and configurations.	Quick deployment and provisioning, leveraging cloud-native tools for rapid scaling and implementation
Noisy Neighbor	Since all network traffic is funneled through a limited set of centralized appliances, high traffic from one tenant or application can degrade the performance of others sharing the same resources	Effectively mitigates the "Noisy Neighbor" issue through its decentralized design

SmartGroup Design Best Practices

IMPORTANT THINGS TO NOTE

- Use Dynamic SmartGroups wherever possible.
- Take advantage of the hierarchy in SmartGroup types to provide governance. Tags that an application owner should not be able to over-ride (such as PCI, Prod/Dev, etc.) should leverage VNET/VPC type smartgroups. RBAC can be applied to these tags via the CSP and inherited by the workloads. This enables a blend of developer controlled policy and security-controlled policies.
- Currently VM-type smartgroups only work for IaaS/VMs. To capture PaaS you should use CIDR or Subnet-type SmatGroups.
- SmartGroups could take several minutes to update upon tag changes or initial provisioning. For actions that have to happen at boot time, consider leveraging SmartGroups that are of the subnet/VNET type.

Dynamic SmartGroup Hierarchy



Tags at the VPC/VNET level and possibly the subnet level can be used for Guardrail policies since application owners don't have IAM privileges to edit these tags. Examples include Lifecycle, Data Classification, etc.

SmartGroups at the VPC/VNET and subnet level can be used to apply policy to PaaS.

Tags at the workload level will likely be used to defined "Application" and "Role" within the application.

aviatrix

Policy as Code Workflow

- Security/Firewall Admin creates a SmartGroup, Webgroup and Rule for an app team or VPC.
- Notes reason for policy change in Git Commit for future auditing.
- Allows Developers to submit Pull requests for Webgroup

- Developer redeploys an app – no action necessary as long as it has the same tags
- Developer needs to add a FQDN to their Internet Egress - submits a pull request for their Webgroup

- Security/Firewall Admin reviews change and merges.
- IAC Pipeline automatically deploys the new policy.

Full change log tracked in Git for easy auditability

Policy Design Best Practices

- Aviatrix Native Firewalling has an implicit, partial Deny, but it is recommended to not rely on implicit policy. Instead, all policy should be explicitly authored for understandability.
- The first policy written should be and explicit catch-all called “Global-Catch-All”. For brownfield it is recommended that this has an action of “allow”. The catch all should have a high priority number and be the last policy in the list.
- Rules are evaluated in order and should have a naming convention that indicates their intent.
- **Internet Egress Policies**
  - For Internet Traffic, use the “Public Internet” pre-defined Smartgroup introduced in 7.1 as the destination, NOT the 0.0.0.0/0 SmartGroup.
  - Internet Policies (especially if they contain webgroups) should be near the bottom of the list.
  - It’s possible to have a single base rule for HTTP and HTTPS and use inspection sub-rules to define more specific policies with FQDN/URL filtering (introduced in Egress 2.0)
  - It’s recommended that source not be a “VM Type” SmartGroup as the delay in recognizing a new VM could restrict bootstrapping. Instead, use a subnet, VNET, or CIDR type SmartGroup.
- Each VPC/VNET will likely have their own catch all as you gradually introduce firewalling policy and enforce. VNET catch-alls should be a pair rules, one for Ingress, and one for Egress.
- Logging. Best practice is to log Deny rules and Internet Egress rules. All other allows will be captured by FlowIQ.

Example Hierarchical Section Design

Section (1-1999): Guardrail Exceptions and Global Allow Policies

Priority	Name	Src	Dst	Port/Proto	Action	Log
1	GR-E-App1-Prod-Dev-DB	App1-Prod-Web	App1-Dev-DB	TCP/3306	Allow	No

Section (2000-3999): Guardrails

1002	GR-Deny-Prod-Dev	Prod,Dev	Prod,Dev	Any	Deny	Yes

Section (5000-29999): Application Policies

10000	App1-Policy-1	Internet	App1-LB	TCP/443	Allow	No
10001	App1-Policy-2	App1-LB	App1-Web	TCP/80	Allow	No
10002	App1-Policy-3	App1-Web	App1-DB	TCP/3306	Allow	No

Section (30000-34999): Explicit, Targeted Catch Alls

90000	Deny-Catch-All-Ingress	Any	VNET1, VNET2	Any	Deny	Yes
90001	Deny-Catch-All-Egress	VNET1, VNET2	Any	Any	Deny	Yes

Section (35000-35999): Internet Egress Policies

2000	Egress-HTTP	Private-Networks	Public Internet	TCP/80	Allow and Inspect	Yes
2001	Egress-HTTPS	Private-Networks	Public Internet	TCP/443	Allow, Inspect, Decrypt	Yes

Section (50000): Explicit Allow - Global Catch All

100000	Global-Catch-All	Any	Any	Any	Allow	No

3rd Party Firewall Service Insertion (Aviatrix FireNet)

Centralized model

Use as necessary

Aviatrix FireNet For 3rd Party FW Service Insertion/Chaining

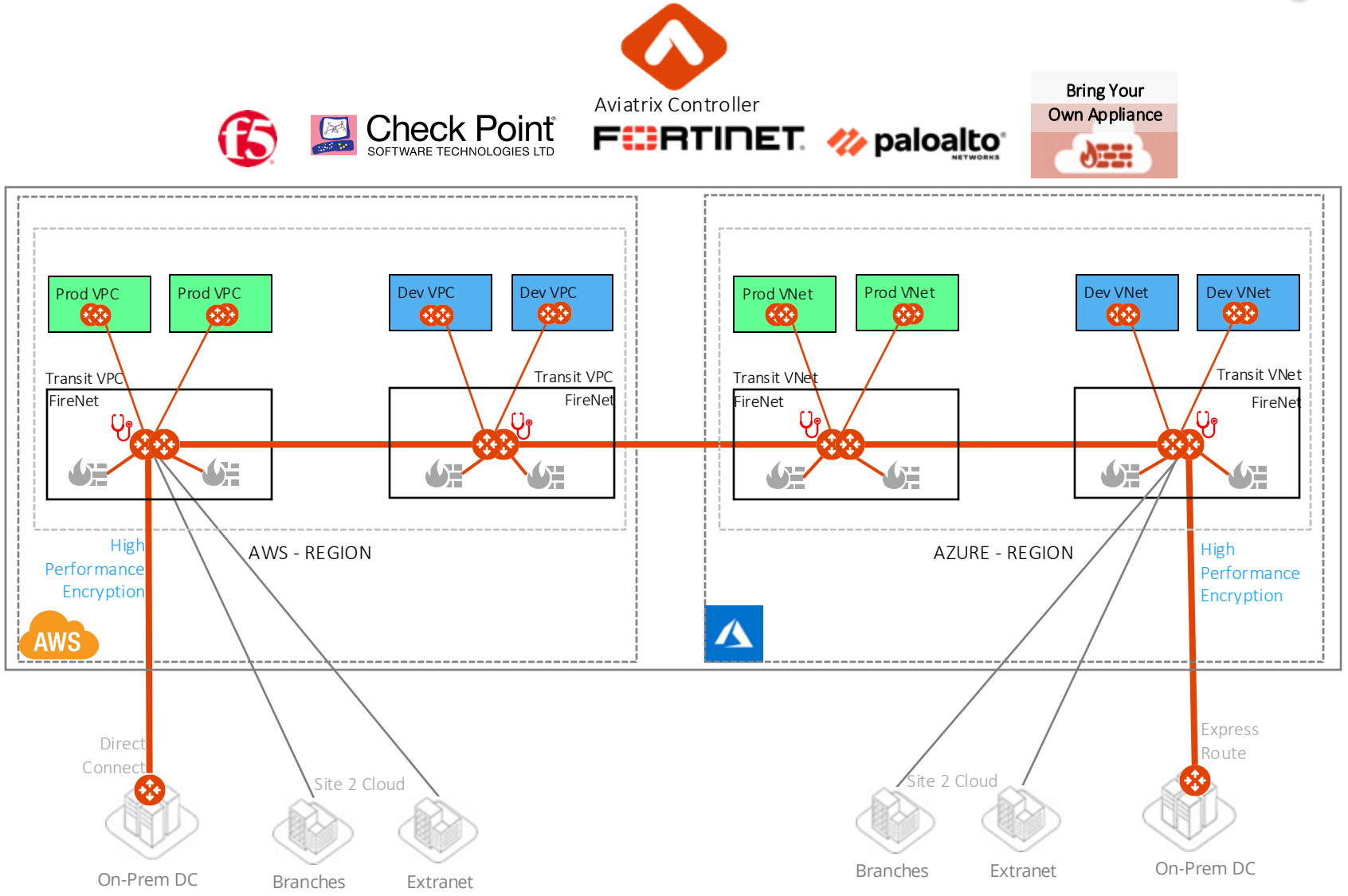
Firewall Service Insertion
- E-W / Egress / Ingress / all traffic
- High Performance Encryption (HPE)
- Active / Active – Across AZs
- No IPsec / No BGP / No SNAT required

Automated Control and Management
- Repeatable architecture across regions/clouds
- Centralized firewall deployment
- Vendor API integration
- UDR and VPC Route propagation

Improved Failure Detection and Failover
- Health Check monitoring

Forwarding Algorithm Options
- Intelligent traffic steering and firewalling based on traffic type
- 5-tuple and 2-tuple

Firewall Bootstrap Support
- Firewall zero-touch deployment capability in Azure and AWS



aviatrix

Aviatrix Certified Engineer (ACE)
https://aviatrix.com/ACE

COMMUNITY
https://community.aviatrix.com