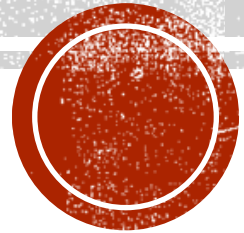
ZERO TRUST IN ACTION: AUTOMATING IDENTITY AND ACCESS CONTROL IN THE CLOUD



Shahed Kazi
Global Lead – Cloud Architecture & Strategy
Everlight Radiology

WHAT IS ZERO TRUST ARCHITECTURE?

Zero Trust Architecture (ZTA) is a cybersecurity framework based on the "**never trust, always verify**" principle, assuming no entity—inside or outside the network—is inherently secure. It eliminates implicit trust, requiring strict, continuous authentication for every user, device, and application requesting access to resources. [*Google*]

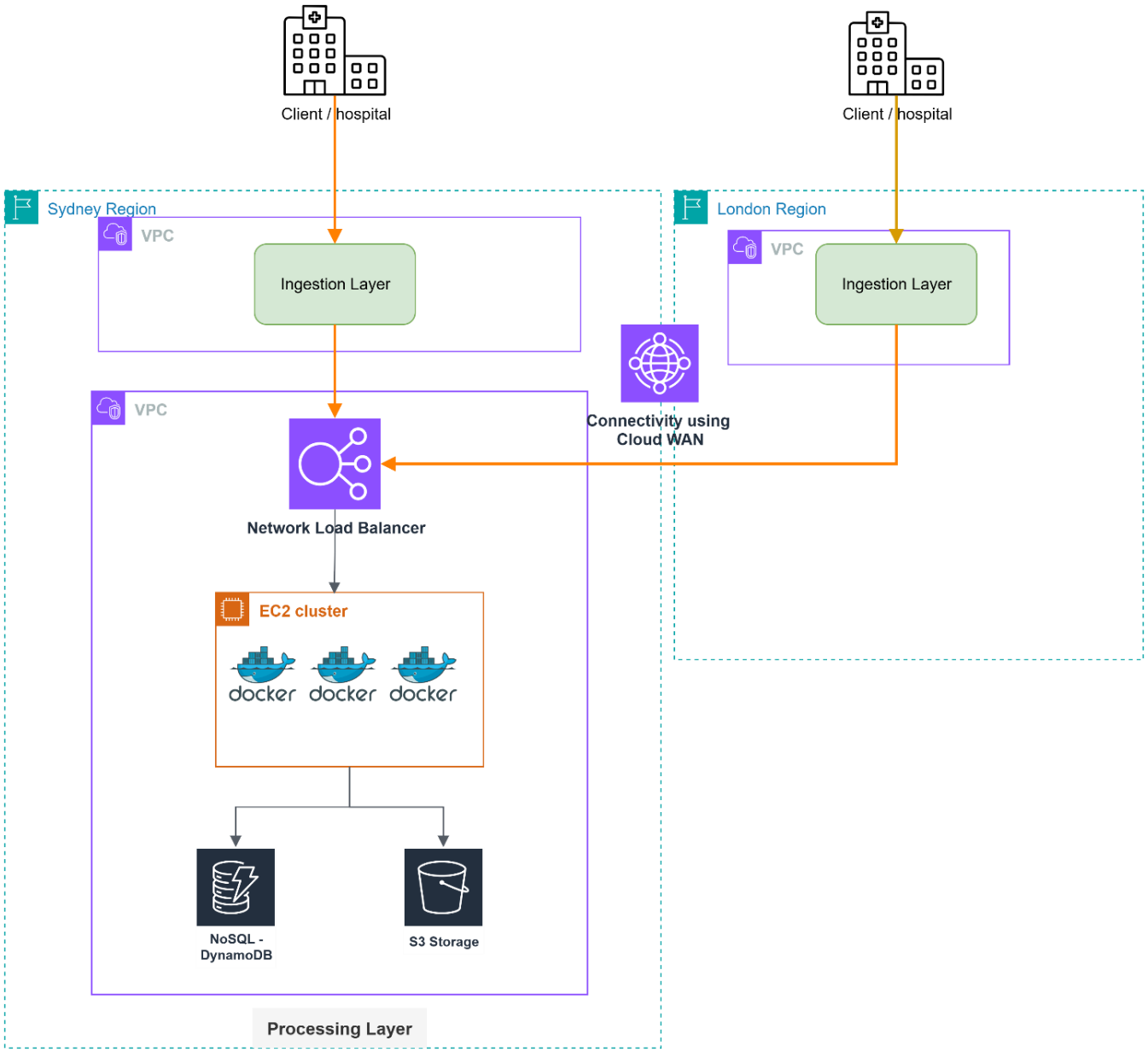
- Least Privilege
- Assume Breach
- Authorisation
- Monitoring
- Micro Segmentation
- Data Protection

AGENDA

- Design Zero-trust Architecture of a medical images archival system
- Compare Zero Trust and Zone Trust
- Data movement across Multi Region & Inter-Region
- Secure Data Storage

ABOUT DICOM

- DICOM (*Digital Imaging and Communications in Medicine*) is the international standard for handling, storing, printing, and transmitting medical imaging information.
- DICOM structure -> pixel data + metadata (patient, hospital, study data)

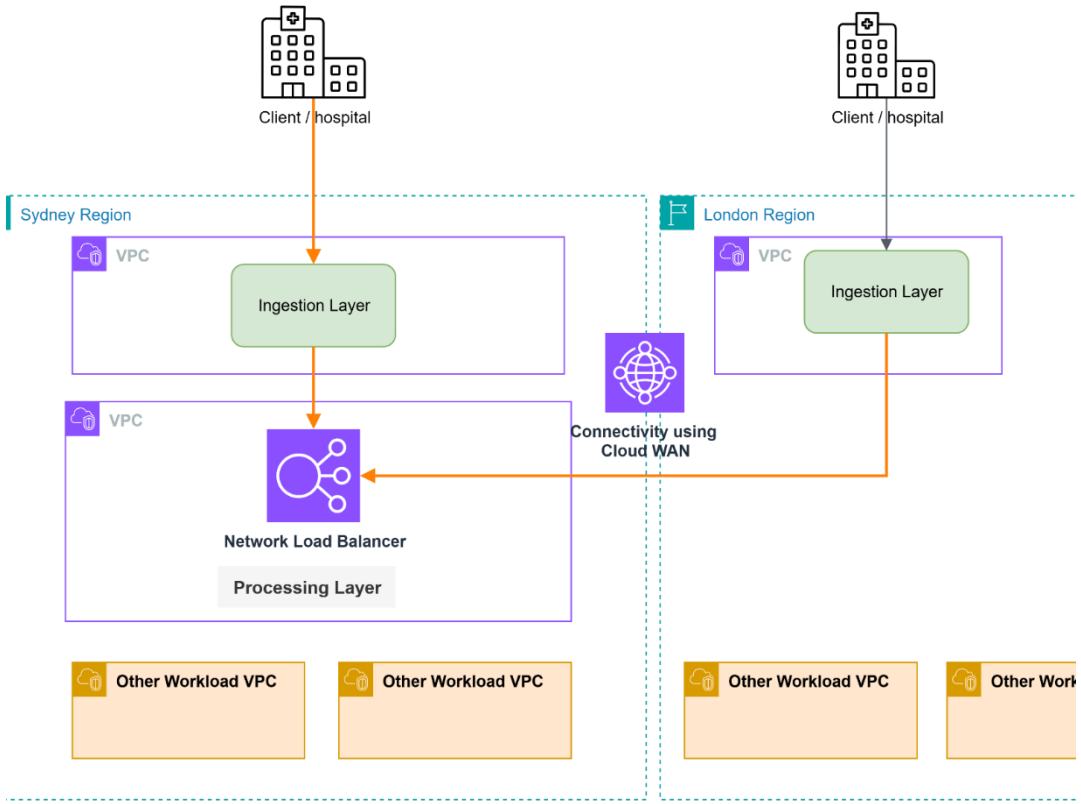


High Level Architecture of Image Archival

ARCHITECTURE OF IMAGING ARCHIVAL

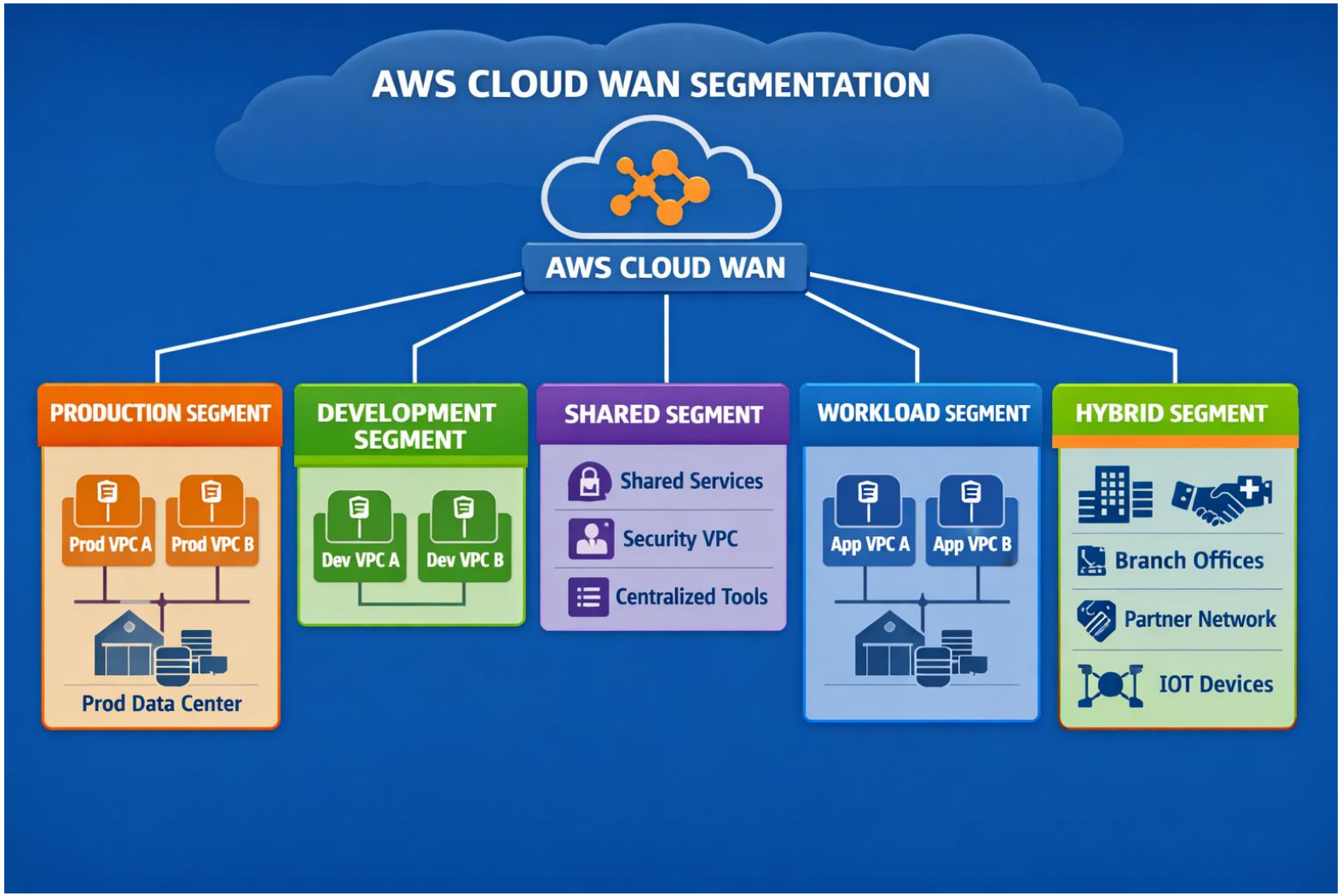
- Hospitals from multiple regions send images
- Ingestion layer forwards images to Processing layer
- Images are stored in S3 bucket
- Image metadata are stored in NoSQL database

MULTI REGION NETWORK DESIGN



Network Design

- Multi-region, multi-VPC connected using Cloud WAN
- Each resource has security group



Network Segmentation

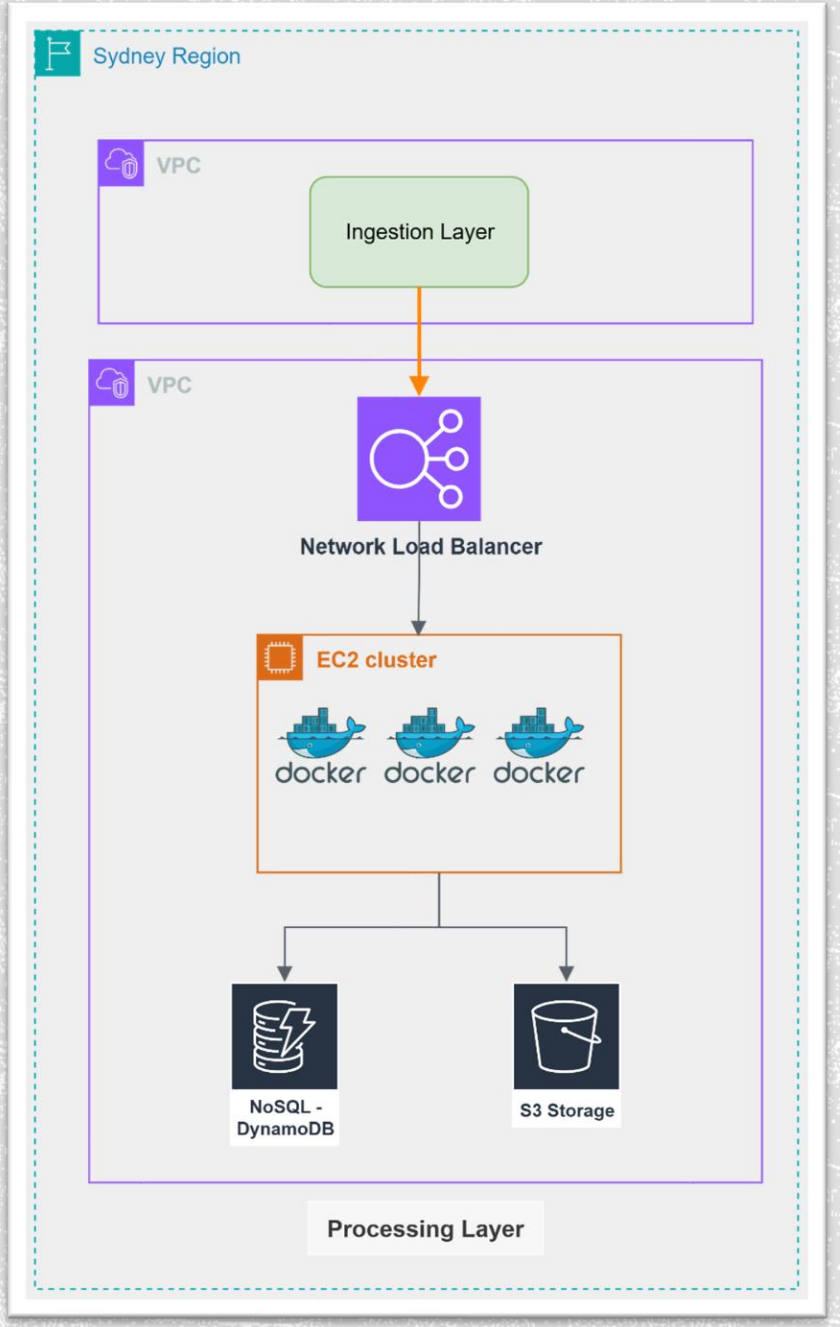
MICRO-SEGMENTATION

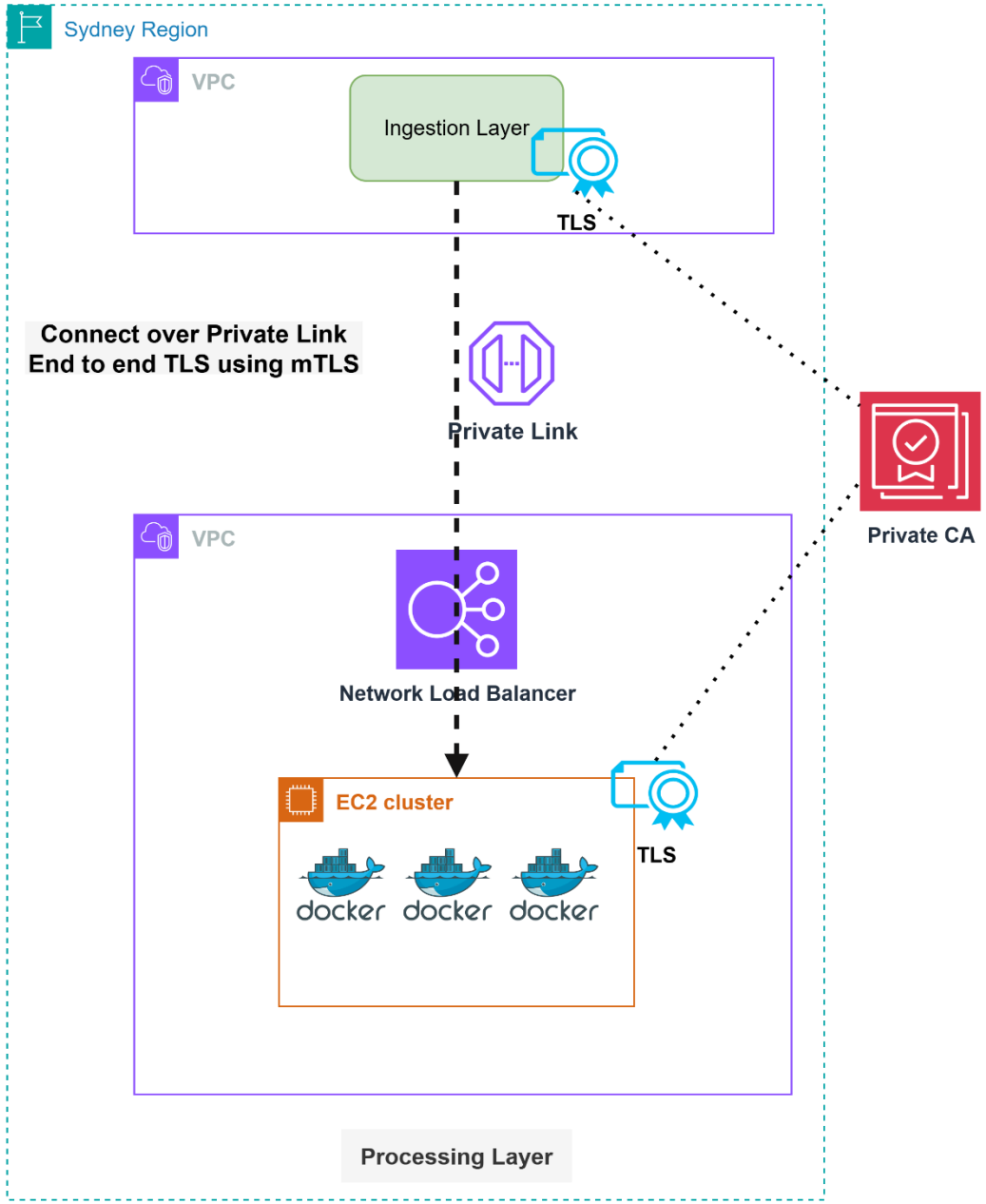
- Create smaller isolated networks – segments
- Allow/deny traffic between segments as needed
- Traffic encrypted using AES-256
- **Reduce blast radius**

WHY NETWORK TRUST BREAKS IN THE CLOUD

Current Design Issues

- Compromised server can send images to NLB
- Traffic flows over Cloud WAN
- No authorisation
- IP-based trust ≠ identity
- Zero trust assumes:
  The network is hostile. Even "internal" traffic must prove identity and authorization



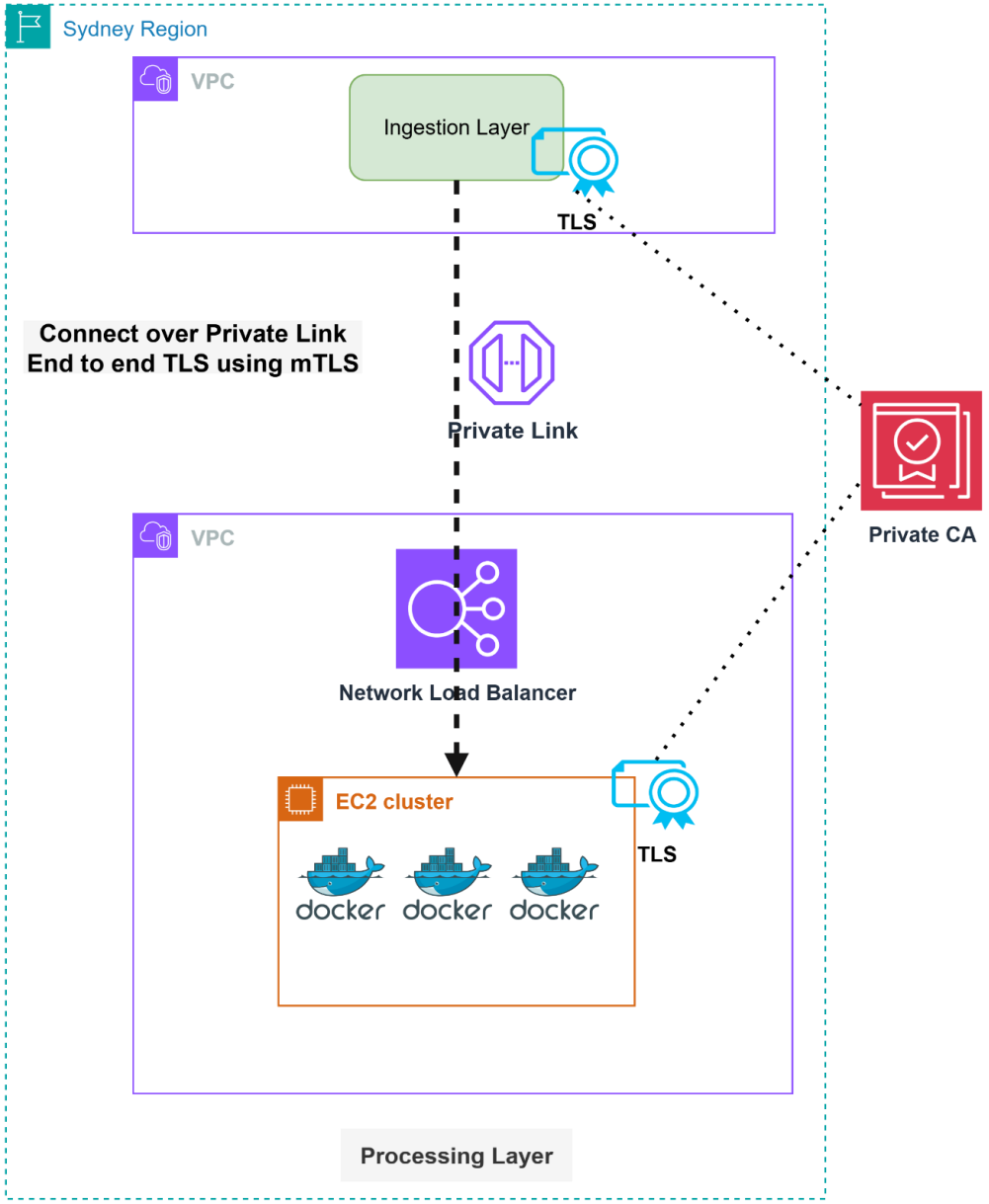


Zero Trust Load Balancer Design

NETWORK LOAD BALANCER – ZERO TRUST

- Workload Identity – use mTLS between ingestion and receiver services
- With mTLS, **each side** has
  - A private key
  - A short-lived certificate
  - A verifiable identity
- NLB performs TLS passthrough
- Receiver performs certificate check
  - Performs authorisation

Continuous Verification and Identity-Based Authentication



Cloud WAN vs Private Link

NETWORK LOAD BALANCER – CLOUD WAN VS PRIVATE LINK

- **Cloud WAN** -> trust at network layer
- **Private Link** -> application specific -> service-level trust
- One VPC **cannot scan, route to, or laterally move** inside the another VPC.

Least Privilege Network Access
Micro-Segmentation

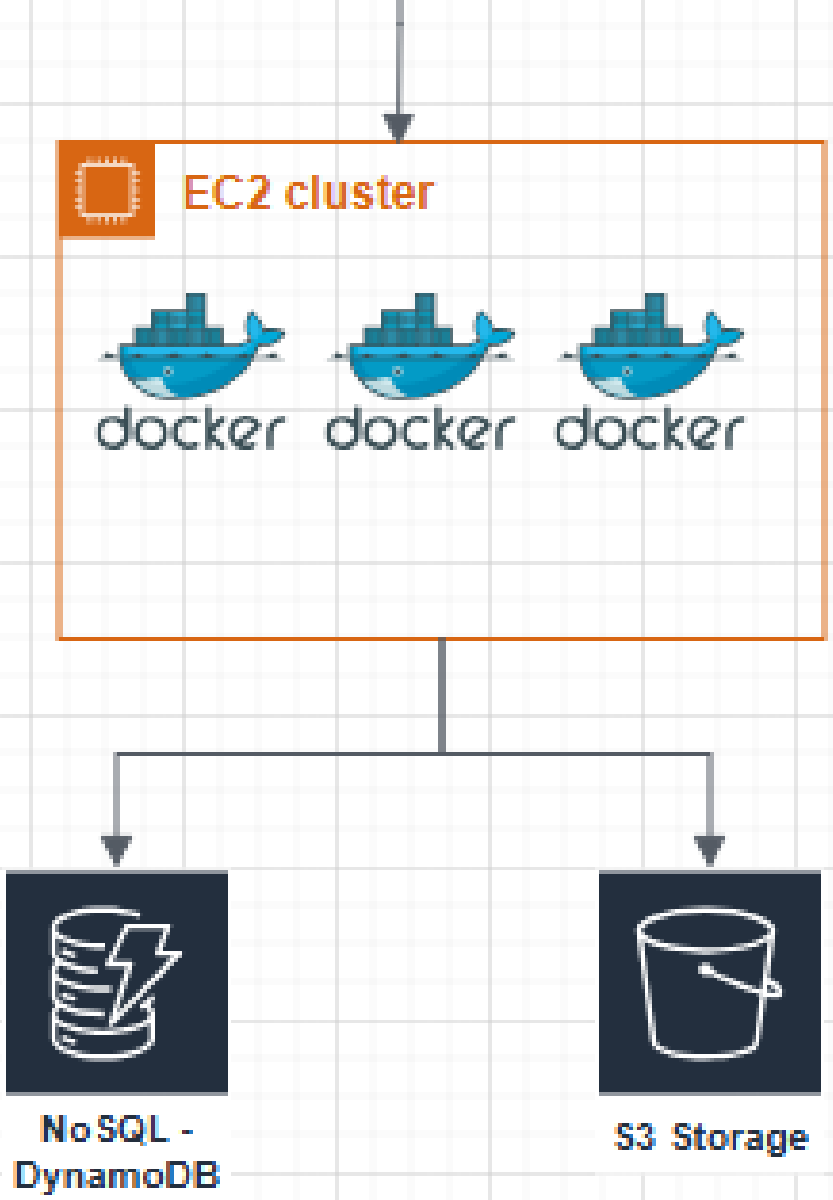
LOAD BALANCER – END TO END TLS?

- Load Balancers perform TLS termination
  - Help save compute resources downstream
- Bypass end to end TLS and Zero Trust
- Setting up mTLS can be costly

Assume Breach

NETWORK LOAD BALANCER – ZONE TRUST BEST PRACTICE

- Private subnets
- Security groups (only NLB → receiver)
- No public IPs
- No wildcard routes
- PrivateLink



Processing Layer

PROCESSING LAYER - DOCKER CLUSTER

- The processing layer – docker cluster
  - Stabilises the study
  - Obtains the metadata
  - Saves metadata in NoSQL database
  - Saves images in Object storage

WRITE TO NOSQL & S3 - IAM

```
IAM POLICY

s3:PutObject
Resource: BUCKET_NAME

dynamoDB:PutItem
Resource: TABLE_NAME
```

- Allow writing to S3 and NoSQL database
- Any service that can assume this role will get permissions to S3 and NoSQL

Identity-Centric Security
Least Privilege Access

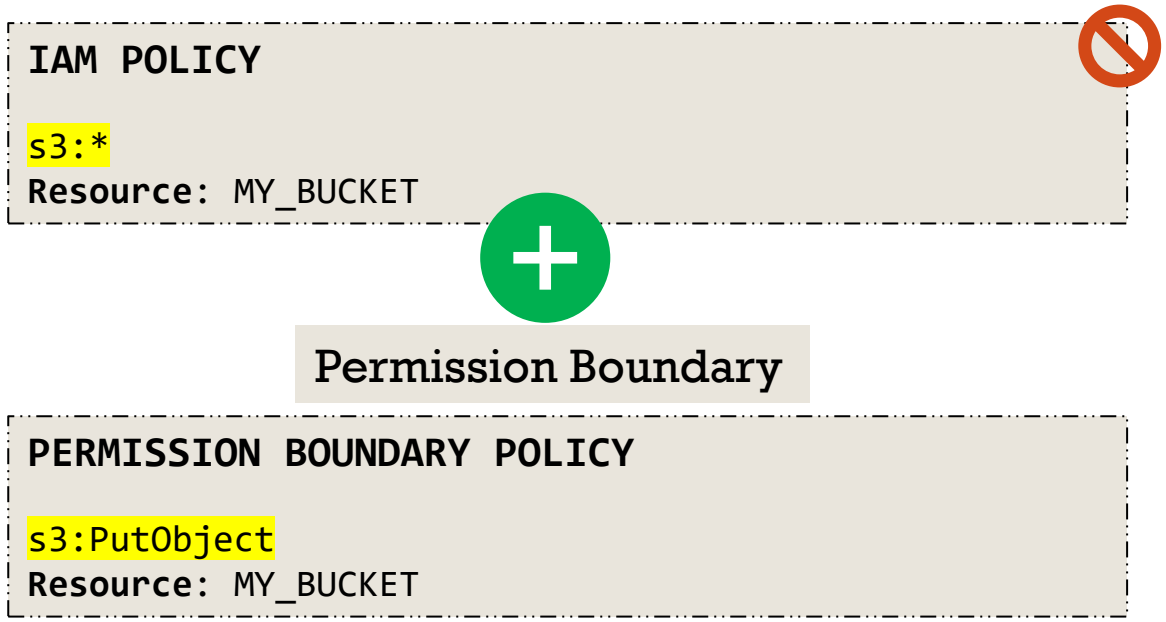
WRITE TO S3 & NOSQL - IAM CONDITIONS

```
IAM POLICY

s3:PutObject
Resource: BUCKET_NAME
Condition: Secure_Transport
Condition: Source_VPC_Endpoint
```

```
IAM POLICY

dynamodb:PutItem
Resource: TABLE_NAME
Condition: Source_VPC_Endpoint
```

- Add condition to allow traffic over TLS only -> S3 allow HTTP by default
- Add condition to allow from a specific VPC endpoint only
- Disable access from internet

IAM PERMISSION BOUNDARY



- Add permission boundary to restrict the maximum permission the role can assume
- Protects against accidental or intentional or mis-aligned IAM

Least Privilege
Blast Radius

IAM ROLE – TRUST POLICY

```
IAM POLICY

sts:AssumeRole
Service: CONTAINER_TASK
Condition: ACCOUNT_ID_1
Condition: CLUSTER_NAME
```

- Workload Identity
- Explicit verification

Explicit Verification (Always Verify)

STORAGE / NOSQL – RESOURCE POLICY

- Resource protect itself — decides who can access it
- Restrict which identities can access
- Deny access even if IAM allows it

Principle of Least Privilege
Explicit Verification (Always Verify)

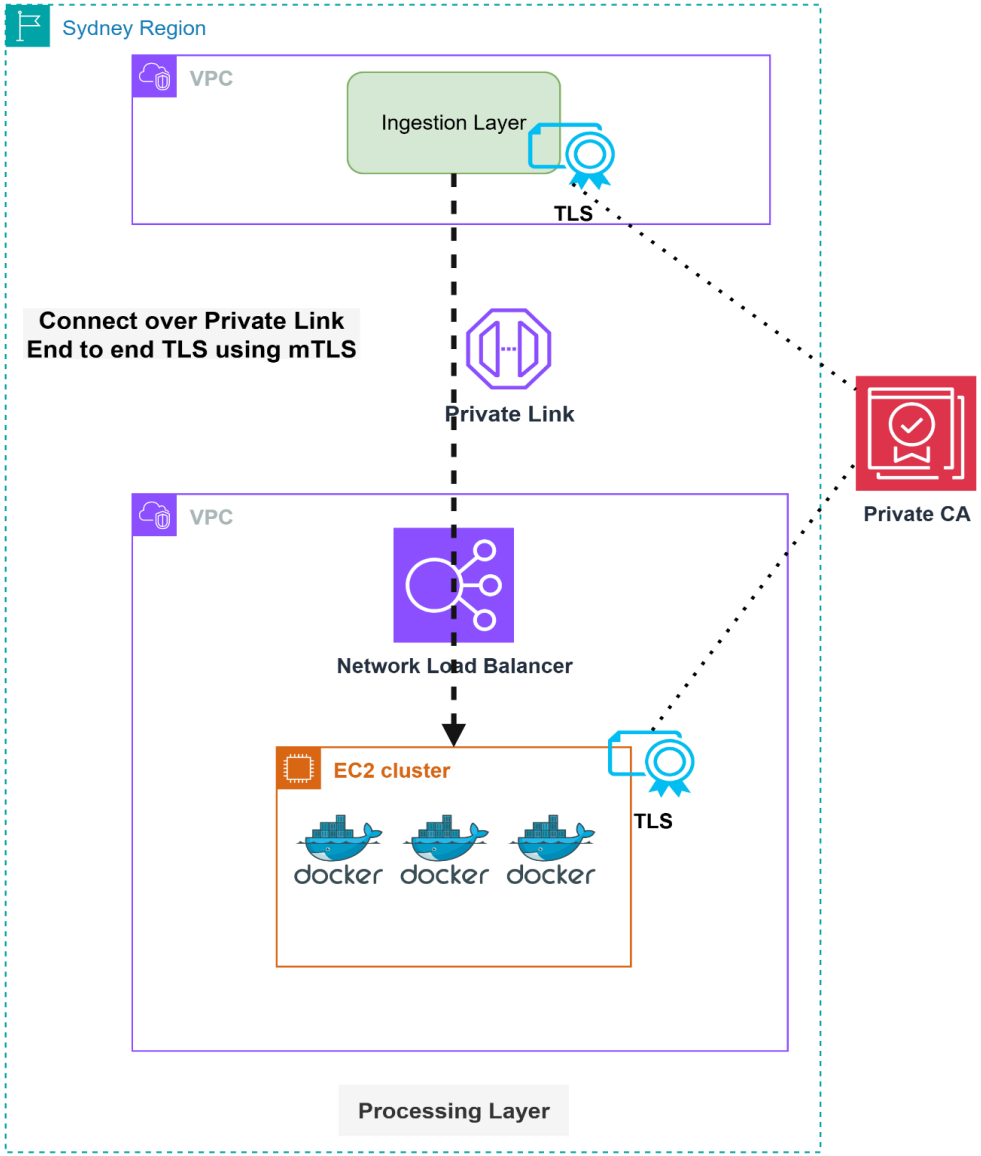
```
RESOURCE POLICY

dynamodb:WriteItem
Resource: MY_TABLE
Principal: MY_IAM_ROLE
```

STORAGE- DATA LOSS PREVENTION (DLP)

- Enable versioning – avoid overwriting / data tampering
- Require MFA delete
- Enable Object Lock – CSP disables deletion for a specified time

Data Security Pillar
Principle of Least Privilege
Assume Breach

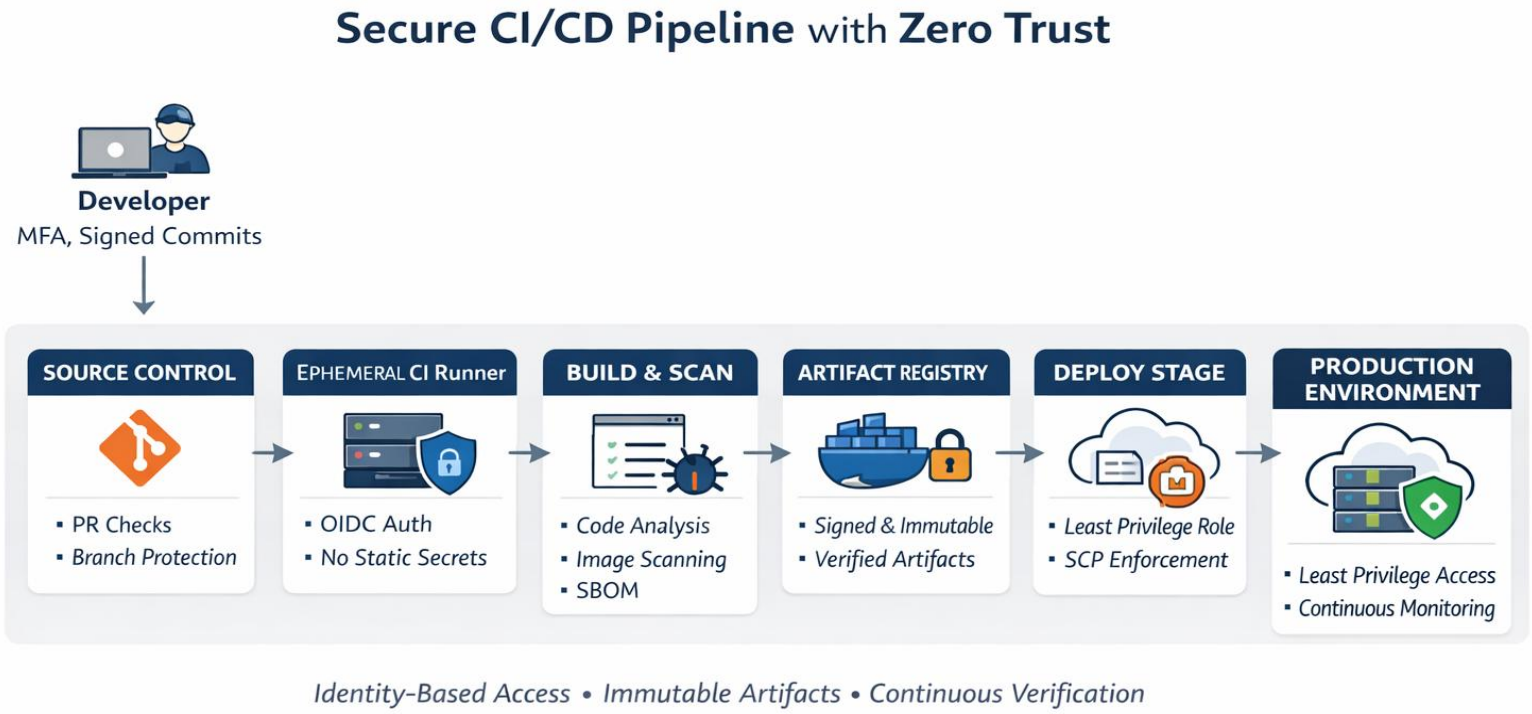
"Zero Trust isn't just about *access* —
it's about assuming data *will* be touched, copied, or deleted and designing controls anyway."



TRUST YOUR DOCKER IMAGE?

- Docker tags are mutable meaning
  - Insecure code will be trusted by our infrastructure
- Turn off tag mutability
- Turn on docker image scanning
- No direct access to docker repo
- Reduced Attack Surface: Attackers cannot replace trusted, audited, and running images with malicious versions.

Reduce Attack Surface
Assume Breach
Verified Integrity



If pipeline isn't Zero Trust, your runtime controls don't matter — attackers ship code, not packets

CI CD PIPELINE

- OIDC for authentication
- Signed commits
- Code analysis (S/DAST)
- MFA

Identity + Integrity

- IAM roles activities
- Resource modification
- Data access patterns
- Real-Time Threat Detection
- Alert / Notify
- Detect Anomalies

Continuous monitoring moves security from a point-in-time check to ongoing verification

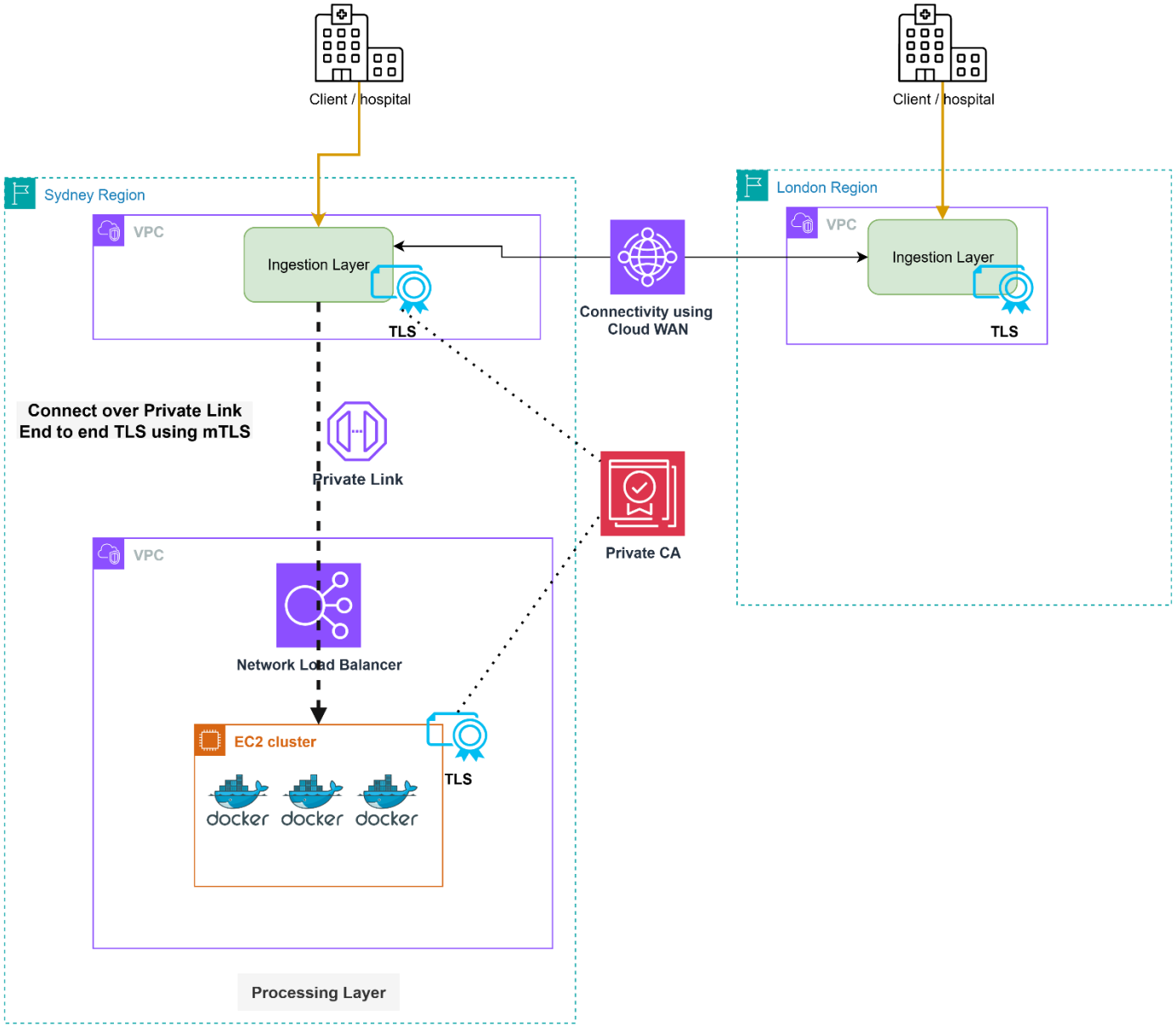
SCENARIO: CICD PIPELINE COMPROMISED

- Modify Resources -> *Policy blocks it*
- Ship Code –> *Signed Commits, Reject Unsigned builds*
- Monitoring -> *Alerts for anomaly*

Assume Breach ✓

Monitoring & Alerting ✓

Zero Trust don't stop attack – it stops the damage



Imaging Archival Architecture

DID WE ACHIEVE ZERO TRUST?

- No implicit trust (network, identity, or pipeline)
- Identity verified at every hop
- Least privilege enforced by policy
- Resources protect themselves
- Continuous verification & visibility
- Zero Trust is a journey, not a state