Tunnel view

**🔹 1. Planning and Requirements Gathering**

* Understand business use case (image processing, spatial matching, OIC output).
* Define infrastructure components (Blob Storage, Azure Files, VNet, Subnet).
* Identify all input/output formats (PGR, LDS, JPG, CSV, OIC).

**🔹 2. Azure Resource Provisioning (via ARM/Terraform/Bicep)**

**2.1. Networking**

* Create a **Resource Group**.
* Create a **VNet** with at least two **subnets**:
  + One for compute workloads (e.g., Azure VMs or Batch).
  + One for storage endpoints with **service endpoints** enabled.

**2.2. Storage**

* Deploy **Azure Blob Storage** for raw and processed image data.
* Deploy **Azure File Storage** for persistent CSV and tabular data (Frame Table, etc.).
* Enable **lifecycle policies** for cleanup.

**2.3. Compute Options (Choose based on need)**

* Azure VM Scale Sets (for Python + tools).
* Azure Batch (for processing jobs).
* Azure Container Instances or AKS (for Dockerized workflows).
* Ensure managed identity and permissions are configured via **Azure RBAC**.

**🔹 3. Code Repository & CI/CD (Azure DevOps)**

**3.1. Repo Setup**

* Create an **Azure DevOps Repo** for:
  + Python scripts (arcpy, ffmpeg, LDS parsing)
  + PowerShell/Bash scripts (automation)
  + JSON Templates (ARM if not using Terraform)

**3.2. CI Pipeline (Build)**

* Lint Python code
* Run unit tests (if applicable)
* Package scripts (e.g., into .whl or Docker image)

**3.3. CD Pipeline (Release)**

* Deploy infrastructure (Terraform/ARM)
* Deploy Python scripts to compute target
* Schedule via Azure Automation or trigger via Azure Logic Apps

**🔹 4. Processing Workflow Implementation**

**4.1. Ingest & Store Files**

* Use **SFTP** (via Azure SFTP or 3rd party) to move PGR/LDS files to Blob.
* Use **Event Grid/Azure Function** to monitor uploads.

**4.2. Video to Images**

* Use custom VM/container with ladybugProcessStream.exe.
* Convert and store .jpg files to Blob.

**4.3. Reprojection**

* Use ffmpeg for image reprojection.
* Run as scheduled jobs or Azure Batch task.

**🔹 5. Spatial Matching (GIS Data Integration)**

* Parse LDS file with **arcpy in Python** (install ArcGIS Pro/ArcPy on compute).
* Use static inputs like exam\_tracks.csv, 1FT Track, and Frame Table CSV.
* Match images and write a new table (CSV) with image path + X/Y coords.

**🔹 6. Frame Table Updates & OIC File Creation**

* Append image locations into Azure File-stored Frame Table.
* Use **Python** or **ArcGIS Pro GP Tools** (scripted) to create/update **OIC files**.
* Automate this with Azure Pipelines or **Logic App** workflows.

**🔹 7. Output Delivery**

* Store final:
  + .jpg in Blob.
  + .csv in File Storage.
  + .OIC in File Storage or serve via GIS.
* Use **Azure SQL or Synapse** to build dashboards or pipelines for reporting.

**🔹 8. Security & Monitoring**

**8.1. Security**

* Use **Private Endpoints** for Storage.
* Enable **Azure Defender for Storage**.
* Use **Managed Identity** for secure script access.

**8.2. Monitoring**

* Set up **Azure Monitor**, **Log Analytics**, and **Application Insights**.
* Track:
  + File ingestion events
  + Job success/failure
  + Storage health

**🔹 9. Cost & Performance Optimization**

* Set **Blob tiering policies** (Hot, Cool, Archive).
* Use **auto-scaling** for compute workloads.
* Schedule jobs during off-peak hours if applicable.

**🔹 10. Documentation & Handover**

* Document:
  + Infrastructure (diagrams, components)
  + CI/CD pipelines
  + Script usage & parameters
  + Troubleshooting steps
* Use Azure DevOps Wiki or Confluence for sharing.

**🔐 Step-by-Step Azure Network Flow (Detailed)**

**📌 Use Case Summary**

A hybrid workload where data is captured on-premises, processed in Azure using compute and storage services, and consumed by various applications (GIS, dashboards, analytics).

**🔹 Step 1: On-Premises Data Generation**

* **Component:** Ladybug Camera System + MTA Premises
* **Function:** Captures raw video (PGR files) and telemetry logs (LDS files).
* **Security:** Encrypted file transmission over **SFTP or VPN** to Azure.

**🔹 Step 2: Secure Transfer to Azure**

* **Connection Method:**
  + **Option 1:** **SFTP via Azure Storage SFTP Endpoint** (Blob Storage with SFTP enabled)
  + **Option 2:** **ExpressRoute or Site-to-Site VPN** to Azure VNet Gateway (if large volume and private transfer needed)

**🔹 Step 3: Azure Network Architecture**

**✅ 3.1 Azure Subscription**

* Logical container for all resources.
* Enforce policies using **Azure Policy** and **Management Groups**.

**✅ 3.2 Azure Virtual Network (VNet)**

* VNet Name: vnet-imageproc-eastus
* **CIDR Range:** 10.10.0.0/16
* Provides isolated network space in Azure.

**✅ 3.3 Subnets (with roles & NSGs)**

| **Subnet Name** | **CIDR** | **Purpose** | **NSG Rules (Sample)** |
| --- | --- | --- | --- |
| subnet-compute | 10.10.1.0/24 | Host VM Scale Sets, Azure Batch | Allow 22, 3389 from Corp IP |
| subnet-storage | 10.10.2.0/24 | Connect Storage (Blob/File) via PE | Deny Internet Outbound |
| subnet-functions | 10.10.3.0/24 | Host Azure Functions or Logic Apps | Allow internal only |
| subnet-monitor | 10.10.4.0/24 | For logging/monitoring (e.g., OMS) | Allow Log Analytics |

Each subnet is associated with **Network Security Groups (NSGs)** to restrict traffic.

**✅ 3.4 Private Endpoints**

* **Blob Storage**: blobstorage-eastus uses **Private Endpoint** on subnet-storage
* **Azure Files**: file-storage-endpoint mapped privately
* This ensures **no public IP exposure**.

**🔹 Step 4: DNS and Name Resolution**

* Use **Azure Private DNS Zone**: e.g., privatelink.blob.core.windows.net
* Linked to VNet for seamless private endpoint resolution.

**🔹 Step 5: Network Flow — Processing Pipeline**

1. **Upload PGR & LDS** to Blob Storage (via private endpoint).
2. **Trigger Azure Function** or Logic App from **Event Grid** (monitors Blob).
3. Function initiates:
   * Azure VM or Container Job in subnet-compute
   * Downloads files from Blob (private access)
   * Runs ladybugProcessStream.exe & ffmpeg
4. **Processed images** stored back into Blob.
5. **Python with arcpy** running on VM/container parses LDS and CSVs, performs spatial joins.
6. **Updated CSV and OIC files** are saved to **Azure File Storage** (via Private Endpoint).

**🔹 Step 6: Data Access/Consumption**

* **Azure SQL Database or Power BI Gateway** connects securely via **Private Link/Subnet** for dashboard consumption.
* GIS Tools (like ArcGIS Pro):
  + Either run on-premises and connect via VPN
  + Or deployed in Azure VMs within subnet-compute

**🔹 Step 7: Monitoring and Logging**

* **Azure Monitor**, **Log Analytics Agent** installed on VMs and resources.
* VNet-integrated **Diagnostic Settings** send logs to:
  + Log Analytics workspace
  + Storage accounts
  + Event Hub (if needed for SIEM)

**🔹 Step 8: Security Controls**

* **NSGs** on each subnet: restrict inbound/outbound traffic.
* **Azure Firewall or 3rd-party NVA** (optional for east-west traffic inspection).
* **Microsoft Defender for Cloud**: enables threat protection.
* **Service Endpoint Policies** (if used instead of Private Link).

**🔹 Step 9: Identity & Access**

* All compute and automation tools use **Managed Identity**.
* **RBAC** used to control access:
  + VM: Reader
  + Function App: Blob Contributor
  + Logic App: Storage File Data SMB Share Contributor
* **Key Vault with Private Endpoint** for storing secrets.

**🧭 Final Diagram Key Components:**

* **VNet**: vnet-imageproc-eastus
* **Subnets**: compute, storage, functions, monitoring
* **Private Endpoints**: Blob, File, SQL
* **NSGs**: tightly scoped to IP/port/application
* **VPN Gateway/ExpressRoute**: optional secure connection to on-prem
* **DNS Zone**: private zone linked to VNet
* **Monitoring**: Log Analytics, Alerts, Dashboards
* **Security**: Defender for Cloud, Key Vault, Role-Based Access Control

**🔢 Step-by-Step Workflow with Numbering**

**MTA Premises (On-Premise Infrastructure)**

**➊ Step 1: Onboard Data Capture**

* Ladybug Camera on the train/bus captures:
  + Raw video data (PGR files)
  + Log metadata (LDS files)

**➋ Step 2: Upload Files to Azure**

* Data is securely transferred from the MTA premises to Azure via:
  + **VPN Gateway**
  + or **ExpressRoute** (if used for private connection)

**Azure Virtual Network (Cloud Infrastructure)**

**➌ Step 3: Store in Azure Blob Storage**

* Uploaded PGR and LDS files are received in Azure via a **VM Scale Set**
* Files are then written to **Azure Blob Storage**
* Secure connection through **Private Endpoint**
* DNS resolution via **Private DNS Zone**

**➍ Step 4: Data Processing**

* Another **VM Scale Set** processes the LDS files:
  + Runs **Python scripts with ArcPy** to parse LDS and match frames
  + Outputs new Frame Table with X/Y data and matching image metadata

**Data Output and Analytics**

**➎ Step 5: Output Generation**

* From the processed data, system generates:
  + .JPG files (from .PGR conversion)
  + .CSV files with spatial/image metadata
  + .OIC files (Oriented Imagery Catalogs)

**➏ Step 6: Consumption**

* Final output is sent to:
  + **GIS Systems** (ArcGIS Pro, web maps, etc.)
  + **Dashboards** (for Executive/Operational visibility)

**Optional Step (if used in Hybrid Mode)**

**➐ Step 7: Off-Board Image Processing**

* In certain cases, image processing (conversion, spatial match) may be done using local compute before uploading to Azure.
* This occurs **off-board** and integrates via secure tunnel (VPN/ExpressRoute).

**✅ Component Mapping Recap:**

| **Step** | **Component** | **Purpose** |
| --- | --- | --- |
| 1 | Onboard Camera | Capture PGR/LDS data |
| 2 | VPN Gateway / ExpressRoute | Secure connection to Azure |
| 3 | VM Scale Set + Private Endpoint | Upload to Blob securely |
| 4 | VM Scale Set (ArcPy) | Process LDS / match with tracks |
| 5 | Blob Storage / Azure Files | Store JPG, CSV, OIC |
| 6 | GIS Systems / Dashboards | Visualize and consume output |
| 7 | Off-Board Laptop (optional) | Local processing before cloud transfer |

Last option

**Step-by-Step Numbering Based on the Diagram**

**📍 MTA Premises (On-Prem Data Source)**

**➊ Step 1: Data Capture**

* The **Onboard Camera** captures:
  + **PGR files** (video data)
  + **LDS files** (metadata logs)
  + Existing data sources

**➋ Step 2: Secure Upload to Azure**

* Data is transferred **securely via VPN or ExpressRoute** through a **VPN Gateway**.

**📍 Azure Virtual Network (Cloud Infrastructure)**

**➌ Step 3: Data Ingestion via VM Scale Set**

* A **VM Scale Set** running in a secure **Subnet** receives the data.

**➍ Step 4: Store Files in Blob Storage**

* The VM writes the files to **Azure Blob Storage** (through a Private Endpoint).
* This step ensures data is securely stored and isolated in the **Azure Virtual Network**.

**➎ Step 5: Parse LDS Data for Spatial Matching**

* Another **VM Scale Set** processes the **LDS files**:
  + Extracts spatial information
  + Generates frame match metadata

**📍 Outputs (Post-Processing)**

**➏ Step 6: Output Generation**

* Final outputs generated:
  + .JPG images (from raw video)
  + .CSV files (with X/Y coordinates and frame references)
* Stored back in Blob/File storage or served through APIs

**➐ Step 7: Consumption by End Systems**

* **GIS Systems**, **dashboards**, or other apps access the processed outputs securely via APIs or data connections.