IPND Migration to AWS – Options for Key Components

# FTP Service:

### ***Option 1***: AWS Transfer Service

* Serverless: No management overhead
* Stored directly to S3.
* SFTP, FTPS, and FTP standards
* We can migrate file transfer workflows to AWS without changing existing authentication systems, domain, and hostnames.
* Customers can continue to exchange files without changing their applications, processes, client software configurations, or behaviour.
* Easy integration with other AWS services for processing and analytics workflows
* Service Managed authentication to authenticate SFTP users using SSH keys.

### ***Option 2***: EC2 Instances in HA across 2 AZs

* As-is. No changes required.
* Requires logic for user separation and storage locally before transferring to staging (S3)

# Key Management:

### ***Option 1***: AWS KMS and Secrets Manager.

* Seamless integration with other AWS Services.
* Minimal management overhead

### ***Option 2:*** As-is. Migrate to EC2/Database

# Staging Unit

### ***Option 1***: AWS S3

* Easy to maintain and extremely cost effective with high durability and availability
* Customer separation based on IAM access control
* Easy transition to Glacier archive using lifecycle policies

### ***Option 2***: ?

# Nodes:

### ***Option 1***: Lambda Functions

* Serverless. Can be triggered based on file upload to S3.
* No infrastructure management overhead
* Continuous scaling
* Cost optimized with millisecond metering
* Natively supports Java, Go, PowerShell, Node.js, C#, Python, and Ruby code, and provides a Runtime API which allows us to use any additional programming languages to author our functions
* AWS Lambda functions can be configured to run up to 15 minutes per execution.
* Fanout

### ***Option 1a***: AWS Batch if processing requires more than 15 mins

### ***Option 2***: Migrate procs to EC2. Refactor and consolidate the procs to support node independent processing.

### ***Option 3***: ECS/EKS Cluster for the Nodes

# Database:

### ***Option 1***: Aurora Serverless V1 (PostgreSQL compatible)

* Oracle to Amazon Aurora using DMS (Schema Conversion Tool)
* On-demand autoscaling
* Automatically starts up, scales compute capacity to match application's usage, and shuts down when it's not in use.
* Removes much of the complexity of managing DB instances and capacity.
* Seamlessly scales compute and memory capacity as needed, with no disruption to client connections
* Pay only for the database resources that we consume, on a per-second basis.
* Fault-tolerant, distributed storage system with six-way replication to protect against data loss
* Will require code refactoring

### ***Option 2***: Amazon RDS for Oracle

* Oracle to Amazon RDS for Oracle using DMS
* Minimal code impact
* Utilise existing licensing

# Client VPN

## ***Option 1***: AWS Client VPN

* Secure connections —secure TLS connection from any location using the OpenVPN client.
* Managed service — It is an AWS managed service, so it removes the operational burden of deploying and managing a third-party remote access VPN solution.
* High availability and elasticity — It automatically scales to the number of users connecting.
* Authentication — It supports client authentication using Active Directory, federated authentication, and certificate-based authentication.
* Granular control — It enables you to implement custom security controls by defining network-based access rules. These rules can be configured at the granularity of Active Directory groups. You can also implement access control using security groups.
* Ease of use — It enables you to access your AWS resources and on-premises resources using a single VPN tunnel.
* Manageability — It enables you to view connection logs, which provide details on client connection attempts. You can also manage active client connections, with the ability to terminate active client connections.
* Deep integration — It integrates with existing AWS services, including AWS Directory Service and Amazon VPC.

Option 2: VPN Appliance on EC2

* Based on current architecture but migrated to EC2
* OpenVPN Appliance