**Lesson Summary: Gateway Team Architecture and Microservices**

**Team Overview**

* **Team Name:** Big Team
* **Key Members:** Multiple speakers discussing various aspects of the team's architecture and services.

**Framework and Services**

* **Microservices:** The team handles multiple microservices running in JVM (Java or Scala), including back-end services, agent dispatcher, and gateways.
* **Current Work:** Combining two separate microservices into one for efficiency.

**Agents and Domains**

* **Agents:** TV agent, home agent, sports agent, entertainment agent (new), CX agent (Xfinity assistant).
* **Domains:** Different agents handle specific domains like TV, home devices, sports, and entertainment.

**Framework Components**

* **Shared Services:** Logging, tracing, and metrics configuration.
* **Framework:** A Java library used by various services to maintain consistency in implementation.

**Technology Stack**

* **Java Versions:** Older services use Java 11 with Spring Boot 2; new services are upgrading to Java 21 with Spring Boot 3.
* **Upgrade Reason:** Spring Boot 2 has reached end-of-support; Spring Boot 3 requires Java 17+, thus the move to Java 21.

**Ingestion and Data Services**

* **Ingestion Services:** Framework and Gateway handle data ingestion, transforming data from MongoDB to Elasticsearch and eventually to RDS (Postgres).
* **Data Pipelines:** Full feed (daily updates) and incremental feed (real-time updates based on MongoDB changes).

**API and Data Processing**

* **API Types:** HTTP API for PTT (Push-to-Talk) and WebSocket API for hands-free services like Sky's voice control.
* **Backend Processing:** ASR (Automatic Speech Recognition) and ARS (Availability Routing Service) for handling audio inputs and content availability.
* **Entities:** Programs, episodes, stations, persons, bands; entity data is stored and resolved using a search engine.

**New Initiatives**

* **Neon Domain:** A new entertainment domain combining data from multiple sources, including TV, sports, and international partners.
* **Control Plane/Data Plane:** Replacing Elasticsearch with RDS for better data handling and scalability.

**Key Infrastructure Components**

* **Database Transition:** Moving from Elasticsearch to Postgres (RDS) to improve data integrity and performance.
* **Polling Mechanism:** A legacy system being phased out in favor of a more efficient data update process via the control plane.
* **Data Centers and Cloud:** North American services are moving from a hybrid model to primarily on-premises data centers; European services continue using cloud infrastructure.

**Communication and Code Repository**

* **Internal Communication:** Slack for team communication; various channels for different topics.
* **Code Repository:** GitHub (cloud) for code storage and version control.

**Onboarding and Documentation**

* **New Member Onboarding:** Access to Slack, GitHub, and necessary permissions; documentation available on the team's wiki and internal channels.

**Action Items for New Joiners**

1. **Access Setup:** Ensure access to Slack, GitHub, and internal documentation.
2. **Code Checkout:** Follow the readme to download and set up the application.
3. **Documentation Review:** Familiarize yourself with the team's wiki and architecture documents.
4. **Permission Requests:** Request necessary permissions for servers and repositories.

This summary provides an overview of the architecture, technology stack, and processes discussed in the lesson, along with action items for new team members. If there are specific sections or additional details needed, please let me know!