**FlowFile & Provenance Chain · Core Processors · Back-pressure & Prioritizer · Template/Version Control Registry · Site-to-Site (HTTPS) · Cluster Admin Basics**

**1. FlowFile & Provenance Chain**

**What is a FlowFile?**

* A FlowFile is the fundamental data structure in NiFi representing a single piece of data.
* Composed of:
  + **Content**: The raw data (e.g., file, JSON, CSV).
  + **Attributes**: Key-value metadata (filename, mime type, etc.).

**FlowFile Lifecycle:**

* **Created** → **Processed** → **Transferred** → **Removed**
* Managed in the FlowFile repository.

**Provenance Tracking:**

* NiFi automatically logs every event (creation, modification, transfer) a FlowFile goes through.
* Use **Provenance Viewer** to:
  + Trace data lineage.
  + Replay FlowFiles.
  + Debug flows and audit user actions.

**Use Cases:**

* Data forensics.
* Compliance auditing.
* Troubleshooting flow anomalies.

**2. Core Processors**

**What are Processors?**

* The functional units of NiFi that ingest, transform, route, or push data.

**Categories:**

* **Input**: GetFile, GetSFTP, ListenHTTP.
* **Routing**: RouteOnAttribute, RouteOnContent.
* **Transformation**: ReplaceText, UpdateAttribute.
* **Output**: PutFile, PutKafka, PutS3.

**Configurable Properties:**

* Scheduling: Timer-based, CRON, or triggered on events.
* Relationships: Success, failure, retry.
* Concurrent Tasks: Number of threads per processor.

**Best Practices:**

* Keep processors small and composable.
* Use custom naming conventions.
* Enable bulletin boards for error visibility.

**3. Back-pressure & Prioritizer**

**What is Back-pressure?**

* A control mechanism to **pause upstream processors** when downstream queue reaches a threshold.
* Prevents memory overflow and data loss.

**Configuration:**

* Queue → Right-click → Configure:
  + **Object threshold** (e.g., 10,000 FlowFiles).
  + **Size threshold** (e.g., 1 GB).

**FlowFile Prioritizer:**

* Determines processing order in the queue.
* Examples:
  + **FirstInFirstOutPrioritizer**: Oldest data first.
  + **NewestFlowFileFirstPrioritizer**: Useful for real-time use cases.
  + **Attribute-based prioritizers**.

**4. Templates & Version Control (NiFi Registry)**

**Templates:**

* Export a set of processors as a reusable blueprint.
* Good for:
  + Sharing across environments.
  + Backups and documentation.

**Version Control with Registry:**

* External service for managing flow versions.
* Enables:
  + **Check-in/Check-out of flows**
  + **Rollback to previous versions**
  + **Collaboration in teams**

**Workflow:**

1. Connect NiFi to a Registry.
2. Add process group to version control.
3. Commit changes.
4. Restore/revert from history.

**5. Site-to-Site Communication (S2S)**

**Use Case:**

Transfer FlowFiles between **two or more NiFi instances** securely and efficiently.

**Components:**

* **Remote Process Group (RPG)**: Sender configures this to connect to remote NiFi.
* **Input/Output Ports**: Defined in receiving/sending NiFi canvas.

**Protocol:**

* Typically uses **RAW TCP or HTTPS**.
* Secure communication requires:
  + TLS setup
  + Certificates on both sides
  + User/Port authorization

**Features:**

* Compression
* Batching
* Fault tolerance and retry logic

**6. Cluster Admin Basics**

**Purpose:**

Scale dataflows and ensure high availability with multiple NiFi nodes.

**Cluster Components:**

* **Nodes**: All nodes share the same flow definition.
* **Zookeeper**: Coordinates cluster state.
* **Primary Node**: Executes “primary only” processors like GetSFTP.

**Configuration (nifi.properties):**

* nifi.cluster.is.node=true
* Unique node.id per instance
* Shared zookeeper.connect.string

**Admin Tools:**

* **Cluster UI**: Monitor node health, load balance, sync status.
* **Bulletins**: Alerts for failures.
* **Status History**: Performance over time.

**Summary Table**

| **Topic** | **Key Concept** | **Real-World Use** |
| --- | --- | --- |
| FlowFile & Provenance | Data + metadata tracking with replay, audit, and debug | Compliance, debugging |
| Core Processors | Atomic units that transform and route data | ETL Pipelines |
| Back-pressure & Prioritizer | Control flow rate to avoid overload | Real-time processing |
| Templates/Registry | Version-controlled, modular flow development | CI/CD in dataflows |
| Site-to-Site (S2S) | Secure data transfer between NiFi instances | Distributed ingestion |
| Cluster Admin Basics | Scale NiFi with Zookeeper-managed nodes and central UI | High availability |

**Additional Resources**

* **NiFi Docs**: https://nifi.apache.org/docs.html
* **NiFi Registry Setup**: https://nifi.apache.org/docs/nifi-registry-docs/
* **TLS Toolkit Guide**: For setting up HTTPS in production
* **Cloudera/Hortonworks Training Materials**
* **NiFi GitHub Repos** for advanced processor examples