

# Эксплуатация Arenadata Streaming (Kafka, NiFi)



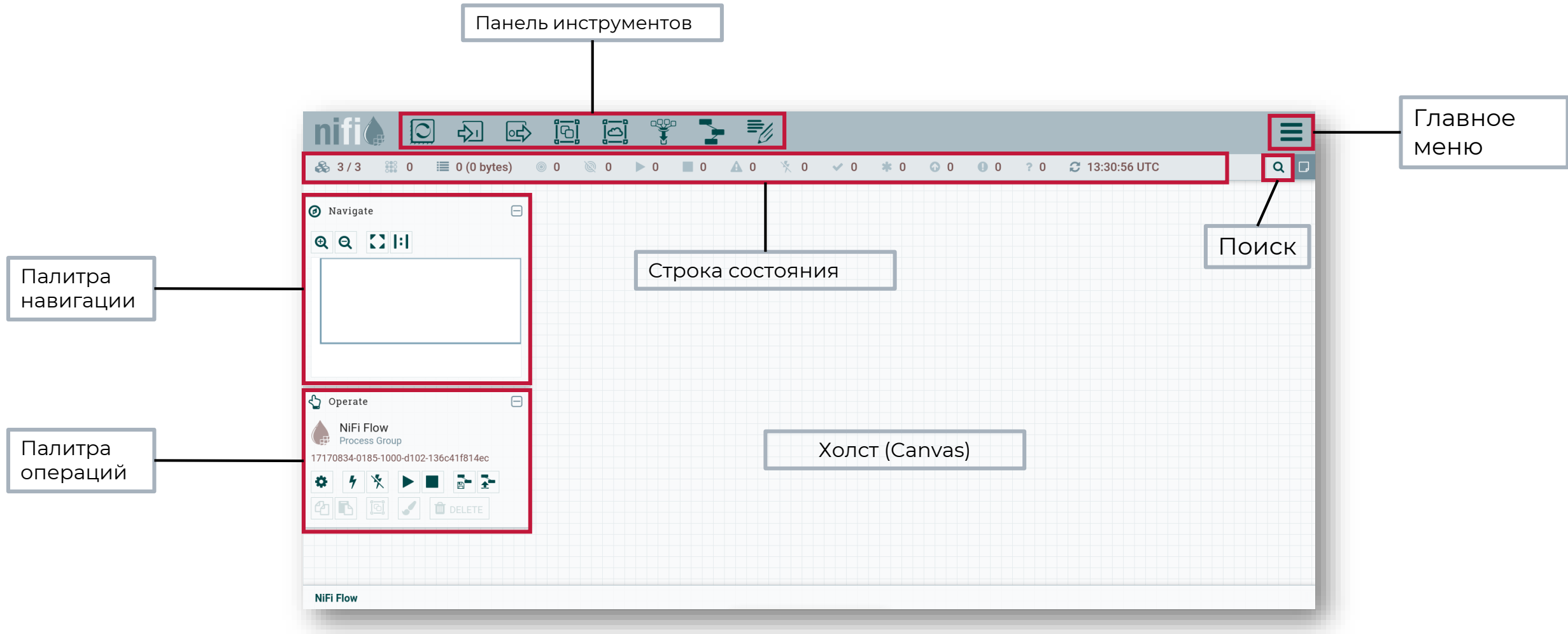
# Архитектура и инструменты Apache NiFi

# Agenda

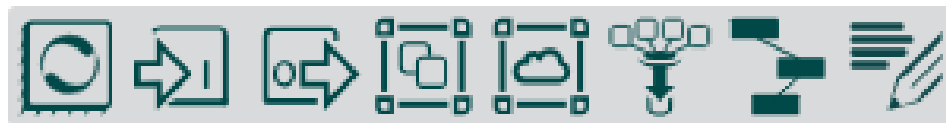
- Обзор Apache NiFi:  
*Графический интерфейс пользователя. Управление потоковой обработкой данных. Процессоры и их назначение.*
- Создание, настройка и управление процессорами, процессорными группами  
*Лабораторная работа.*
- Flow Files и атрибуты.
- Подключение источников (File, Apache Kafka, СУБД (ADB))
- FlowFile Repository, Content Repository, Provenance Repository: детальное изучение
- Производительность и оптимизация потоков данных
- Apache NiFi Registry: версионность

# Обзор Apache NiFi

# Интерфейс NiFi



# Интерфейс NiFi. Панель инструментов



— процессор. Отвечает за вход и выход данных, маршрутизацию и манипулирование ими. Существует множество различных типов процессоров.



—входной порт (Input Port) . Обеспечивает механизм передачи данных в группу процессов.



—выходной порт (Output Port) . Обеспечивает механизм передачи данных из группы процессов в пункты назначения за пределами группы процессов.



—группа процессов (Process Group). Используется для логической группировки набора компонентов, чтобы упростить понимание и обслуживание потока данных.



—группа удаленных процессов (Remote Process Group). Используется аналогично группам процессов. Однако группа удаленных процессов ссылается на удаленный экземпляр NiFi.



—воронка (Funnel). Используется для объединения данных из многих соединений в одно соединение.



—шаблон (Template). Может быть создан из разделов потока или импортироваться из других потоков данных. Шаблоны предоставляют собой крупные строительные блоки для быстрого создания сложного потока.



—метка (Label). Используется для прикрепления записей к частям потока данных.

# Интерфейс NiFi. Панель инструментов


 3 / 3    0    0 (0 bytes)    0    0    0    2    2    0    0    0    0    0    0    12:53:48 UTC

 — количество узлов в кластере и количество подключенных в данный момент узлов;

 — количество потоков, которые в настоящее время активны;

 — объем данных, которые в настоящее время существуют в потоке;

 — количество групп удаленных процессов, передающих данные;


 — количество групп удаленных процессов, не передающих данные;


 — количество работающих процессоров;

 — количество остановленных процессоров;

 — количество недействительных процессоров;

 — количество отключенных процессоров;

 — количество обновленных групп процессов;

 — количество локально измененных групп процессов;













 — количество устаревших групп процессов;

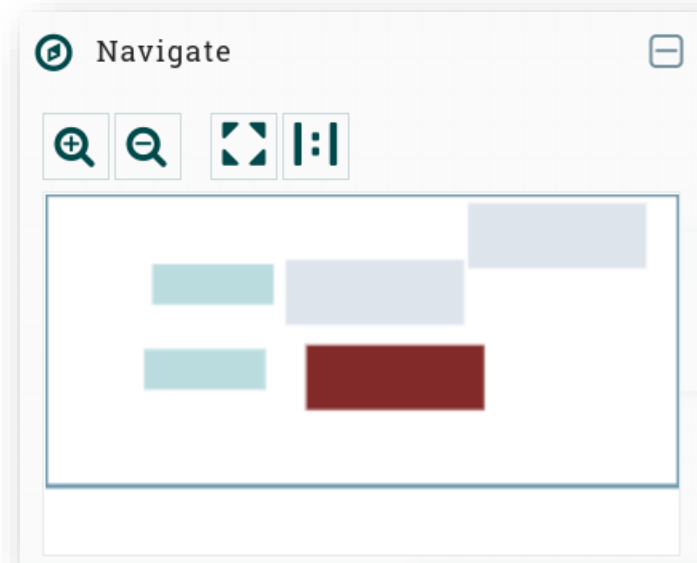
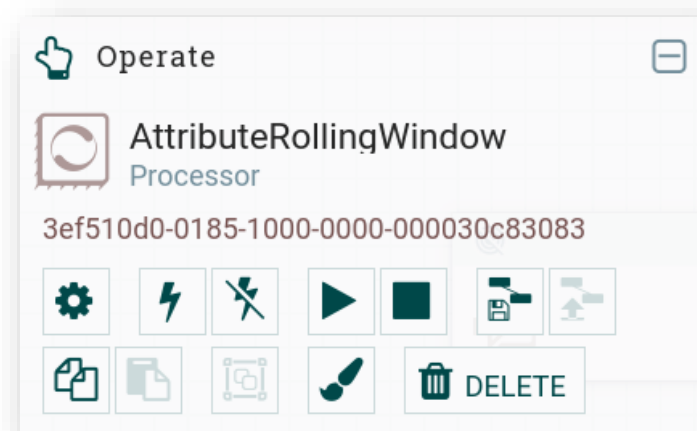
 — количество локально измененных и устаревших групп процессов;

 — количество групп процессов с ошибкой синхронизации;

 — отметка времени последнего обновления всей информации.

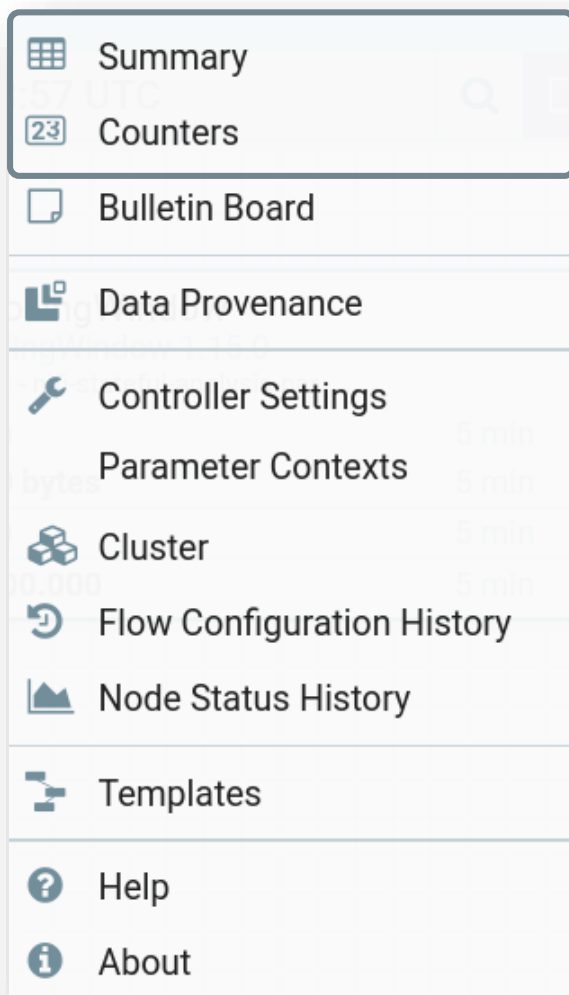
# Интерфейс NiFi. Палитры

-  — просмотр и изменение конфигурации созданного потока;
-  — включение потока (ввод в действие в схеме);
-  — отключение потока (вывод из действия в схеме);
-  — запуск потока;
-  — остановка потока;
-  — создание шаблона;
-  — загрузка созданного шаблона;
-  — копирование потока;
-  — вставка потока (где это допустимо);
-  — создание группы процессоров (где это допустимо);
-  — изменение цвета символа потока, используется для визуальной подсветки разных
-  DELETE — удаление потоков.





# Интерфейс NiFi. Главное меню



- **Summary**—предоставляет сводку о всех настроенных потоках данных и их элементах.

NiFi Summary

PROCESSORS INPUT PORTS OUTPUT PORTS REMOTE PROCESS GROUPS CONNECTIONS PROCESS GROUPS

Displaying 3 of 3

Filter by name View: Single node Cluster

	Name ^	Type	Process Grou...	Run Status	In (Size) 5 m...	Read   Write ...	Out (Size) 5 m...	Tasks   Time ...	
❗	AttributeRolli...	AttributeRolli...	NiFi Flow	Invalid	0 (0 bytes)	0 bytes   0 b...	0 (0 bytes)	0   00:00:00...	→ 📊 ⚙️
❗	AttributeRolli...	AttributeRolli...	NiFi Flow	Disabled	0 (0 bytes)	0 bytes   0 b...	0 (0 bytes)	0   00:00:00...	→ 📊 ⚙️
❗	AttributeRolli...	AttributeRolli...	a	Invalid	0 (0 bytes)	0 bytes   0 b...	0 (0 bytes)	0   00:00:00...	→ 📊 ⚙️

🔄 Last updated: 08:00:53 UTC system diagnostics

- **Counters**—счетчики событий, используется для целей мониторинга.

NiFi Counters

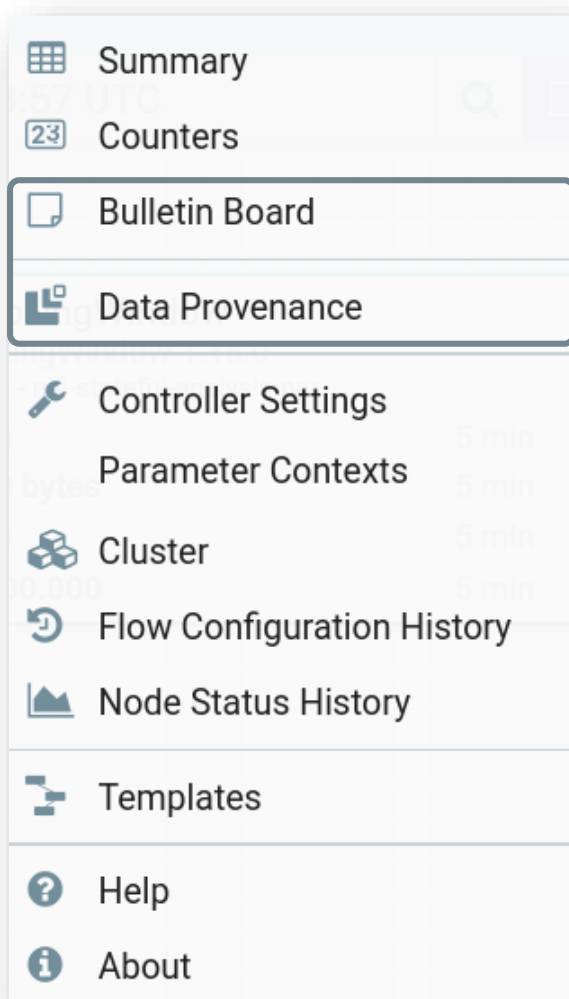
Displaying 0 of 0

Filter by name

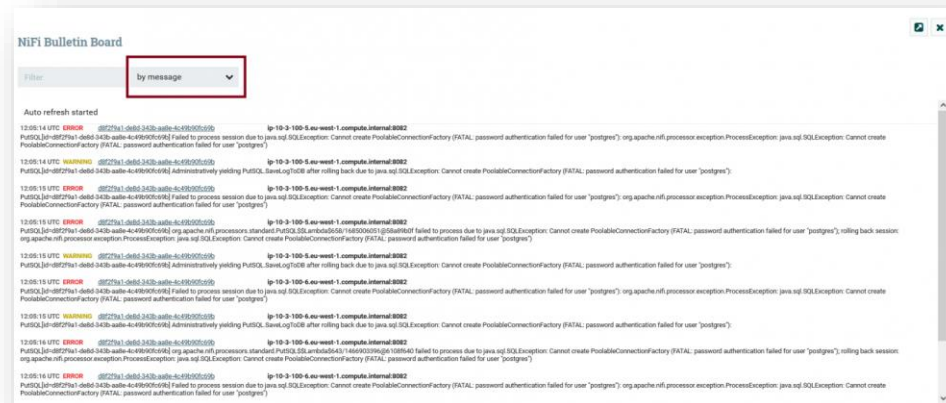
Context ^	Name	Value
-----------	------	-------

🔄 Last updated: 08:04:58 UTC

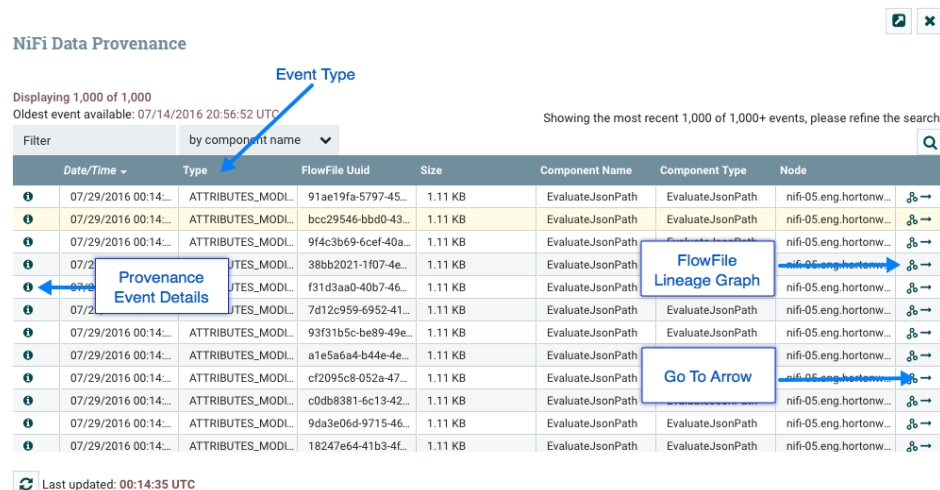
# Интерфейс NiFi. Главное меню



- **Bulletin Board**—показывает бюллетени произошедших событий, доступна фильтрация на основе компонента, сообщения и т. д.



- **Data Provenance**—детализированная информация для каждой части данных, которые принимает NiFi.



# Интерфейс NiFi. Главное меню

Summary

Counters

Bulletin Board

Data Provenance

Controller Settings

Parameter Contexts

Cluster

Flow Configuration History

Node Status History

Templates

Help

About

- **Controller Settings**— позволяет пользователям просматривать/изменять контроллер, включая службы контроллера управления, задачи отчетности, клиенты реестра, поставщиков параметров и узлы в кластере.

NiFi Settings

GENERAL

MANAGEMENT CONTROLLER SERVICES

REPORTING TASKS

REGISTRY CLIENTS

PARAMETER PROVIDERS

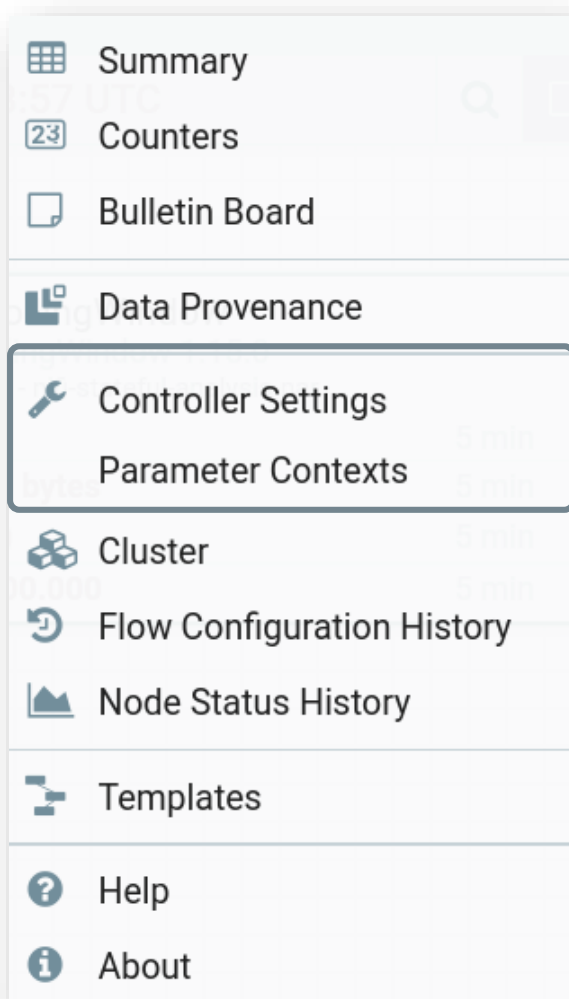
Name ^	Type	Bundle	State	Scope
ADCM Metric Reporter Service	GraphiteMetricReporterService 1.20.0	org.apache.nifi - nifi-metrics-reporting-nar	Enabled	Controller

- **Parameter Contexts**— показывает контексты параметров, глобально определенные для экземпляра NiFi.

NiFi Parameter Contexts

Name ^	Description
parameter_context_1	
parameter_context_2	
parameter_context_3	

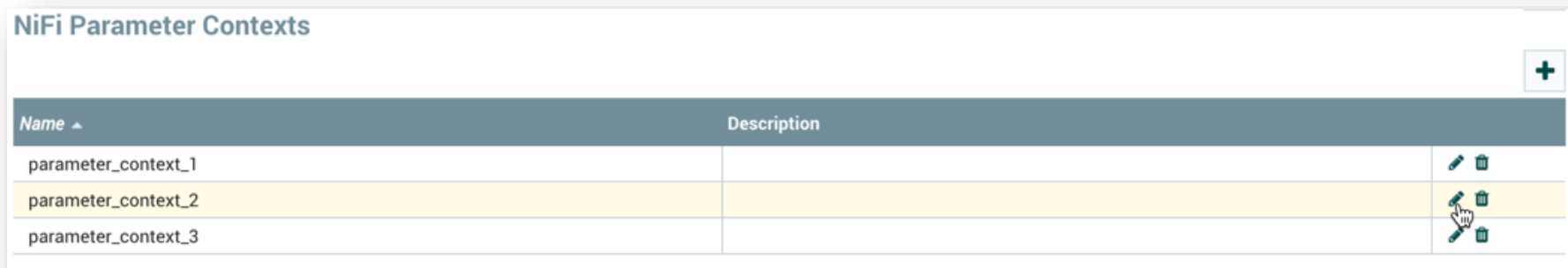
# Интерфейс NiFi. Главное меню



- **Controller Settings**— позволяет пользователям просматривать/изменять контроллер, включая службы контроллера управления, задачи отчетности, клиенты реестра, поставщиков параметров и узлы в кластере.



- **Parameter Contexts**— показывает контексты параметров, глобально определенные для экземпляра NiFi.



# Интерфейс NiFi. Главное меню

Summary

Counters

Bulletin Board

Data Provenance

Controller Settings

Parameter Contexts

Cluster

Flow Configuration History

Node Status History







Templates

Help

About






- **Cluster**—показывает параметры кластера.

NiFi Parameter Contexts

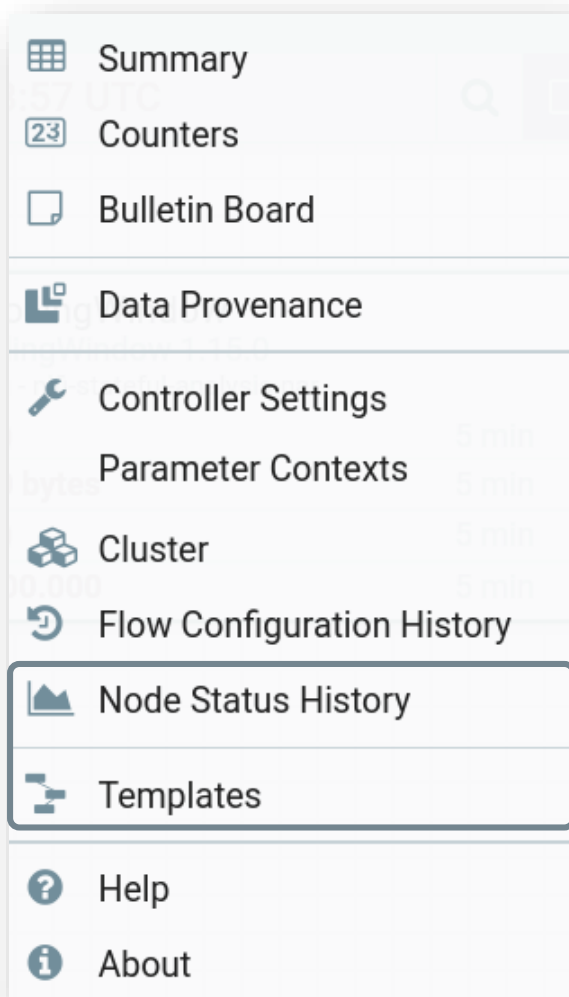
Name ▲	Description	
parameter_context_1		 
parameter_context_2		 
parameter_context_3		 

- **Flow configuration history**—показывает историю изменения конфигурации для всех ПОТОКОВ.

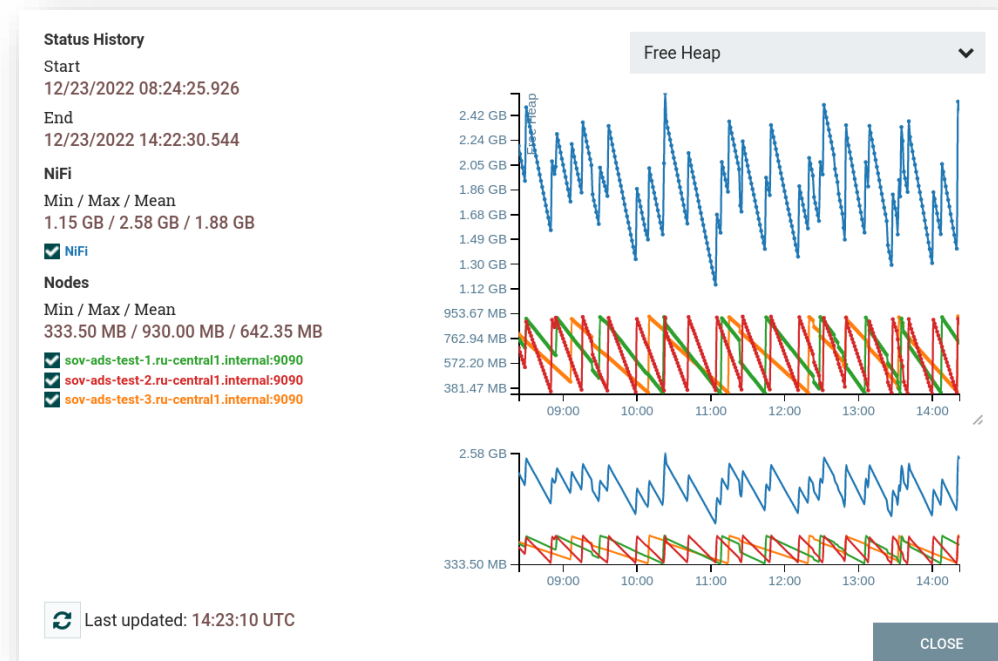
NiFi History

	Date/Time ▼	Name	Type	Operation	User
	12/26/2022 07:48:30 UTC	AttributeRollingWindow	Processor	Disable	anonymous
	12/26/2022 07:34:36 UTC	123	ProcessGroup	Enable	anonymous
	12/26/2022 07:34:34 UTC	123	ProcessGroup	Disable	anonymous
	12/26/2022 07:34:32 UTC	123	ProcessGroup	Disable	anonymous
	12/26/2022 07:34:31 UTC	123	ProcessGroup	Disable	anonymous

# Интерфейс NiFi. Главное меню



- **Nodes Status History**—показывает историю статуса всех узлов кластера.



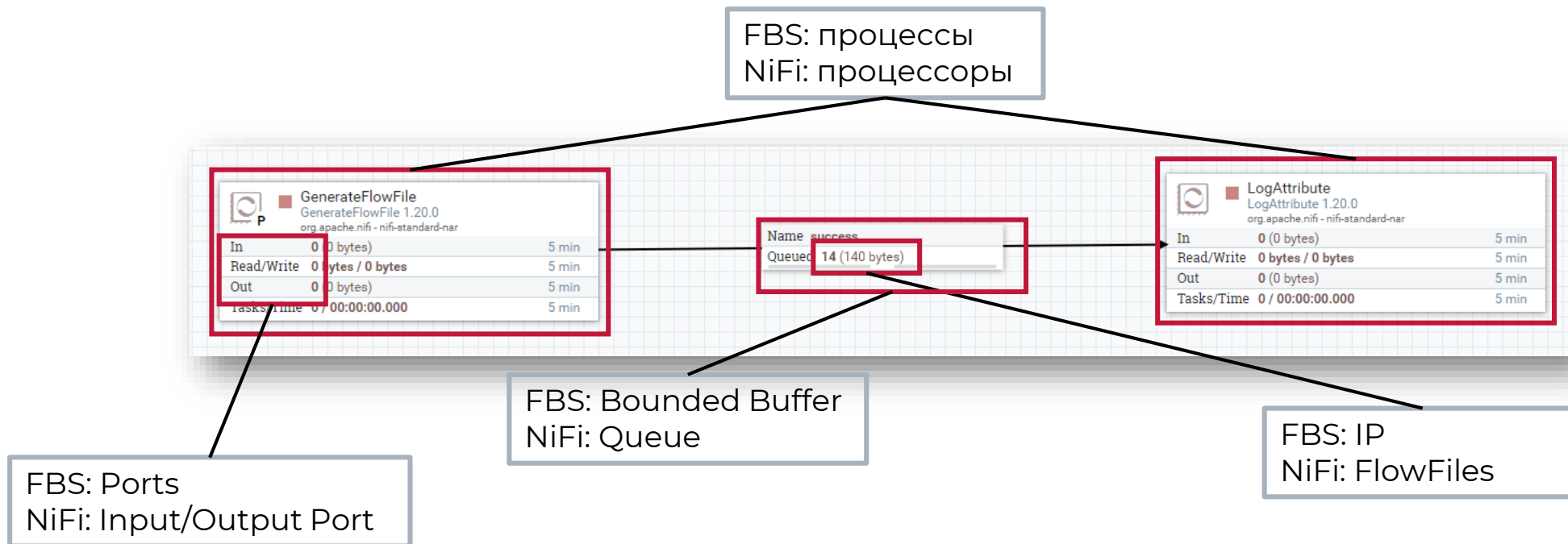
- **Templates**—показывает созданные шаблоны.

The screenshot shows the 'NiFi Templates' page. It displays a table with the following data:

Date/Time	Name	Description	Process Group
12/16/2022 09:53:34 UTC	simple-mi...	Empty str...	17170834-...

# Flow Based Programming (FBS) & NiFi

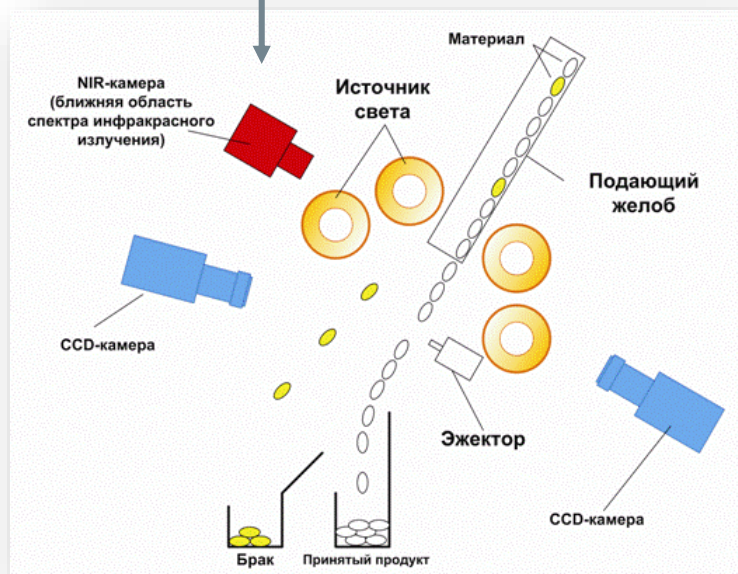
- Flow Based Programming – это парадигма программирования, которая определяет приложения как сеть «черных ящиков» (**процессов**), которые обмениваются данными (**IP – Information Packets**) по predetermined соединениям посредством передачи сообщений, где соединения (**Bounded Buffer**) указываются извне для процессов. Эти процессы «черного ящика» можно бесконечно повторно связывать для формирования различных приложений без необходимости внутреннего изменения.





# Процессоры и их назначение

Processors  
(перемещает  
посылки и  
сортирует.  
Например, фото  
сепаратор)



FlowFiles

Content FlowFile

Attributes  
FlowFile



# Процессоры и их назначение

Индикатор состояния показывает текущее состояние процессора.

- Возможны следующие варианты:
  - Running (работает): процессор в данный момент работает;
  - Stopped (остановлен): процессор валиден и включен, но не работает;
  - Invalid (невалидный): процессор включен, но в настоящее время не может быть запущен. Всплывающая подсказка указывает почему процессор невалидный;
  - Disabled (отключен): процессор не работает и не может быть запущен, пока он не будет включен. Этот статус не указывает, валидный ли процессор.

## Индикатор ошибки

- Процессор регистрирует какое-либо событие, он создает бюллетень.
- Можно настроить – какие бюллетени должны отображаться в пользовательском интерфейсе.
- Значение по умолчанию – WARN, в пользовательском интерфейсе будут отображаться только предупреждения и ошибки

Индикатор состояния процессора

Имя процессора

Индикатор ошибки (Bulletin Board)

Тип процессора

PutSQL.SaveLogToDB		
PutSQL 1.9.0.1.0.0.0-90		
org.apache.nifi - nifi-standard-nar		
In	0 (0 bytes)	5 min
Read/Write	0 bytes / 0 bytes	5 min
Out	0 (0 bytes)	5 min
Tasks/Time	0 / 00:00:00.271	5 min

Статистика (1 раз в 5 минут)

Активные задачи

RouteOnContent		
RouteOnContent 1.9.0.1.0.0.0-90		
org.apache.nifi - nifi-standard-nar		
In	14 (4.2 KB)	5 min
Read/Write	4.2 KB / 0 bytes	5 min
Out	14 (4.2 KB)	5 min
Tasks/Time	14 / 00:00:00.074	5 min

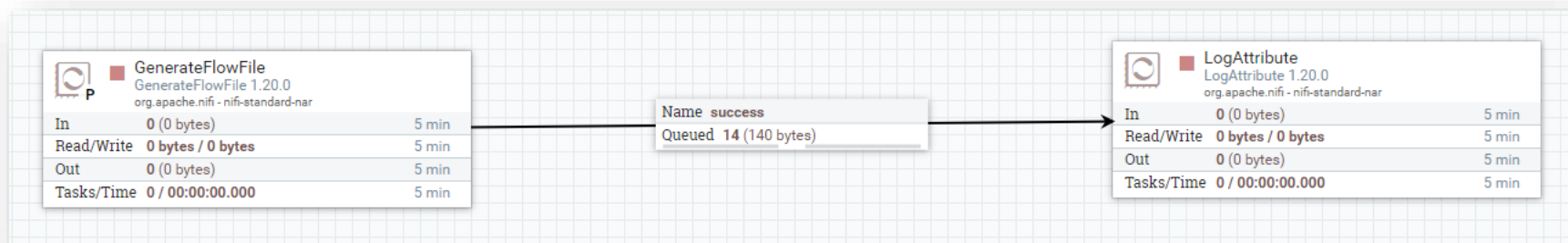
Статистика (1 раз в 5 минут)

# Процессоры. Классификация

Data Ingestion Processors	Процессоры используются для приема данных. В основном это отправная точка любого потока данных в Apache NiFi. Примеры — GetFile, GetHTTP, GetFTP, GetKAFKA и т. д.
Routing and Mediation Processors	Процессоры маршрутизации используются для маршрутизации файлов потока к различным процессорам или потокам данных в соответствии с информацией в атрибутах или содержимом этих файлов потока. Эти процессоры также отвечают за управление потоками данных NiFi. Примеры, — RouteOnAttribute, RouteOnContent, ControlRate, RouteText и т. д.
Database Access Processors	Процессоры этой категории доступа к СУБД способны выбирать или вставлять данные или выполнять и подготавливать другие операторы SQL из базы данных. Эти процессоры в основном используют настройки контроллера пула подключений к данным Apache NiFi. Примеры, — ExecuteSQL, PutSQL, PutDatabaseRecord, ListDatabaseTables и т. д.
Attribute Extraction Processors	Процессоры отвечают за извлечение, анализ и изменение атрибутов потокового файла в потоке данных NiFi. Примеры, — UpdateAttribute, EvaluateJSONPath, ExtractText, AttributesToJSON и т. д.
System Interaction Processors	Процессоры используются для запуска процессов или команд в любой операционной системе. Эти процессоры также запускают сценарии на многих языках для взаимодействия с различными системами. Примеры — ExecuteScript, ExecuteProcess, ExecuteGroovyScript, ExecuteStreamCommand и т. д.
Data Transformation Processors	Процессоры способны изменять содержимое потоковых файлов. Их можно использовать для полной замены данных файла потока, который обычно используется, когда пользователю необходимо отправить файл потока в качестве тела HTTP для вызова процессора HTTP. Примеры — replaceText, JoltTransformJSON и т. д.
Sending Data Processors	Процессоры отправки данных обычно являются конечными процессорами в потоке данных. Эти процессоры отвечают за хранение или отправку данных на сервер назначения. После успешного сохранения или отправки данных эти процессоры УДАЛЯЮТ файл потока с отношением успеха. Примеры — PutEmail, PutKafka, PutSFTP, PutFile, PutFTP и т. д.
Splitting and Aggregation Processors	Эти процессоры используются для разделения и объединения содержимого, присутствующего в потоковом файле. Примеры — SplitText, SplitJson, SplitXml, MergeContent, SplitContent и т. д.
HTTP Processors	Эти процессоры обрабатывают вызовы HTTP и HTTPS. Примеры — InvokeHTTP, PostHTTP, ListenHTTP и т. д.
AWS Processors	Процессоры AWS отвечают за взаимодействие с системой веб-сервисов Amazon. Примеры — GetSQS, PutSNS, PutS3Object, FetchS3Object и т. д.
Other	...

# Лабораторная работа

- Создать Flow:



- Добавить новый атрибут и изменить поле Custom Text.
- Запустить Flow и посмотреть очередь.

# Создание, настройка и управление процессорами, процессорными группами

# Настройки процессора. Settings

- *Name*— имя процессора. Имя процессора по умолчанию совпадает с типом процессора. Рядом с именем процессора находится флажок *Enabled*, указывающий включен ли процессор.
- *Id*—уникальный идентификатор процессора.
- *Type*—тип процессора.
- *Bundle*— пакет NAR.
- *Yield Duration*— период времени без ответа от удаленной службы, по истечении которого процессор должен "уступить", что предотвратит запланированное выполнение процессора в течение некоторого периода времени.
- *Penalty Duration*— период времени, в течении которого предотвращается запланированное выполнение процессора в случае ошибки приема или передачи файлов в удаленную службу.
- *Bulletin Level*— самый низкий уровень бюллетеня, который должен отображаться в пользовательском интерфейсе.

Configure Processor | RouteOnAttribute 1.16.0

Stopped

SETTINGS SCHEDULING PROPERTIES RELATIONSHIPS COMMENTS

Name  
RouteOnAttribute ☒ Enabled

Id  
532c2699-017f-1000-c2ad-df176546d571

Type  
RouteOnAttribute 1.16.0

Bundle  
org.apache.nifi - nifi-standard-nar

Penalty Duration <sup>?</sup> 30 sec Yield Duration <sup>?</sup> 1 sec

Bulletin Level <sup>?</sup> WARN

CANCEL APPLY

# Настройки процессора. *Scheduling*

- *Scheduling Strategy*— стратегия планирования. Возможные варианты планирования компонентов:
- *Timer driven*— процессор будет запускаться через регулярные промежутки времени. Интервал запуска процессора определяется параметром *Run Schedule*.
- *CRON driven*— при использовании режима планирования, управляемого CRON, процессор планируется запускать периодически, аналогично режиму планирования, управляемому таймером. Однако режим, управляемый CRON, обеспечивает значительно большую гибкость за счет увеличения сложности конфигурации. Значение планирования, управляемое CRON, представляет собой строку из шести обязательных полей и одного необязательного поля, каждое из которых разделено пробелом.
- *Concurrent Tasks*— определяет, сколько потоков будет использовать процессор. Другими словами, это определяет, сколько FlowFiles должно обрабатываться этим процессором одновременно.
- *Run Schedule*— определяет, как часто процессор должен запускаться по расписанию. Допустимые значения для этого поля зависят от выбранной стратегии планирования. Значение по умолчанию 0 секунд означает, что процессор должен запускаться как можно чаще, пока у него есть данные для обработки.
- *Execution*— используется для определения, на каком узле (узлах) будет запланировано выполнение процессора.

Configure Processor | ConsumeKafka\_1\_0 1.18.0

Stopped

SETTINGS SCHEDULING PROPERTIES RELATIONSHIPS COMMENTS

Scheduling Strategy ⓘ  
Timer driven ▼

Concurrent Tasks ⓘ 1 Run Schedule ⓘ 0 sec

Execution ⓘ  
All nodes ▼

CANCEL APPLY

# Настройки процессора. Properties

- Предоставляет механизм для настройки поведения конкретного процессора.
- Свойства по умолчанию отсутствуют.
- Каждый тип процессора имеет свои свойства.

Configure Processor | ConsumeKafka\_1\_0 1.18.0

Stopped

SETTINGS SCHEDULING PROPERTIES RELATIONSHIPS COMMENTS

Required field

Property	Value
Kafka Brokers	localhost:9092
Security Protocol	PLAINTEXT
Kerberos Service Name	No value set
Kerberos Credentials Service	No value set
Kerberos Principal	No value set
Kerberos Keytab	No value set
SSL Context Service	No value set
Topic Name(s)	topic-test
Topic Name Format	names
Honor Transactions	true
Group ID	1
Offset Reset	latest

CANCEL APPLY

# Настройки процессора. Relationships

- *Automatically Terminate / Retry Relationships*:
- *Automatically Terminate*—чтобы процессор считался действительным и способным работать, каждое отношение, определяемое процессором, должно быть либо подключено к нижестоящему компоненту, либо автоматически завершено. Если отношение завершается автоматически, любой FlowFile, маршрутизируемый к этому отношению, будет удален из потока, а его обработка будет считаться завершенной. Любая связь, которая уже связана с нижестоящим компонентом, не может быть автоматически завершена. Отношение должно быть сначала удалено из любого соединения, которое его использует. Кроме того, для любого отношения, которое выбрано для автоматического завершения, статус автоматического завершения будет очищен (отключен), если отношение будет добавлено к соединению.
- *Automatically Retry*—пользователи также могут настроить, следует ли повторять FlowFiles, маршрутизируемые к данной связи.
- *Number of Retry Attempts*—для отношений, настроенных на повторную попытку, это число указывает, сколько раз FlowFile будет пытаться повторно обработать, прежде чем он будет направлен в другое место.
- *Retry Back Off Policy*—когда FlowFile необходимо повторить, пользователь может настроить политику отсрочки с двумя параметрами:
  - *Penalize*—повторные попытки будут происходить вовремя, но процессор продолжит обработку других FlowFiles.
  - *Yield*—никакая другая обработка FlowFile не будет выполняться до тех пор, пока не будут предприняты все повторные попытки.
- *Retry Maximum Back Off Period*—первоначальные повторные попытки основаны на значениях *Penalty Duration* и *Yield Duration*, указанных на вкладке *SETTINGS*. Время продолжительности многократно удваивается для каждой последующей попытки повтора. Это число указывает максимально допустимый период времени до следующей повторной попытки.
- *Bulletin Level*—самый низкий уровень бюллетеня, который должен отображаться в пользовательском интерфейсе.

The screenshot shows the 'Configure Processor' dialog box for 'ConsumeKafka\_1\_0\_1.18.0'. The 'Relationships' tab is selected. It contains settings for 'Automatically Terminate / Retry Relationships', 'Number of Retry Attempts', 'Retry Back Off Policy', and 'Retry Maximum Back Off Period'. The 'Automatically Terminate / Retry Relationships' section has a 'success' status and checkboxes for 'terminate' (unchecked) and 'retry' (checked). The 'Number of Retry Attempts' is set to 10. The 'Retry Back Off Policy' is set to 'Penalize' (selected) and 'Yield' (unselected). The 'Retry Maximum Back Off Period' is set to 10 mins. There are 'CANCEL' and 'APPLY' buttons at the bottom right.

Configure Processor | ConsumeKafka\_1\_0\_1.18.0

Stopped

SETTINGS SCHEDULING PROPERTIES RELATIONSHIPS COMMENTS

Automatically Terminate / Retry Relationships ⓘ

success

☐ terminate ☒ retry

FlowFiles received from Kafka. Depending on demarcation strategy it is a flow file per message or a bundle of messages grouped by topic and partition.

Number Of Retry Attempts ⓘ

10

Retry Back Off Policy ⓘ

☒ Penalize ☐ Yield

Retry Maximum Back Off Period ⓘ

10 mins

CANCEL APPLY



# Настройка групп процессов

- *Process Group Name* — имя группы процессов.
- *Process Group Parameter Context* — контекст параметров группы процессов, который используется для предоставления параметров компонентам потока.
- *Process Group Comments* — комментарии группы процессов.
- *Process Group FlowFile Concurrency* — используется для управления тем, как данные передаются в группу процессов (цветом помечены совместимые настройки):
  - *Unbounded* — входные порты в группе процессов будут принимать данные настолько быстро, насколько это возможно.
  - *Single FlowFile Per Node* — входные порты будут одновременно пропускать только один FlowFile.
  - *Single Batch Per Node* — входные порты будут вести себя так же, как и в режиме *Single FlowFile Per Node*, но при приеме входные порты будут продолжать принимать FlowFiles, пока очередь не станет пустой, после начнется обработка всех накопленных файлов в текущей группе процессоров.
- *Process Group Outbound Policy* — исходящая политика контролирует поток данных из группы процессов:
  - *Stream When Available* — данные, поступающие на выходной порт, немедленно передаются из группы процессов.
  - *Batch Output* — порты вывода не будут передавать данные из группы процессов до тех пор, пока все данные (FlowFiles), находящиеся в группе процессов, не будут поставлены в очередь к порту вывода.
- *Default Settings for Connections* — срок действия FlowFile по умолчанию.
- *Default Back Pressure Object Threshold* — количество файлов FlowFile, которые могут находиться в очереди.
- *Default Back Pressure Data Size Threshold* — пороговое значение размера данных, которые могут находиться в очереди.

### New Group Configuration

GENERAL

CONTROLLER SERVICES

Process Group Name

New Group

Process Group Parameter Context

No parameter context

Process Group Comments

Process Group FlowFile Concurrency

Unbounded

Process Group Outbound Policy

Stream When Available

Default FlowFile Expiration

0 sec

Default Back Pressure Object Threshold

10000

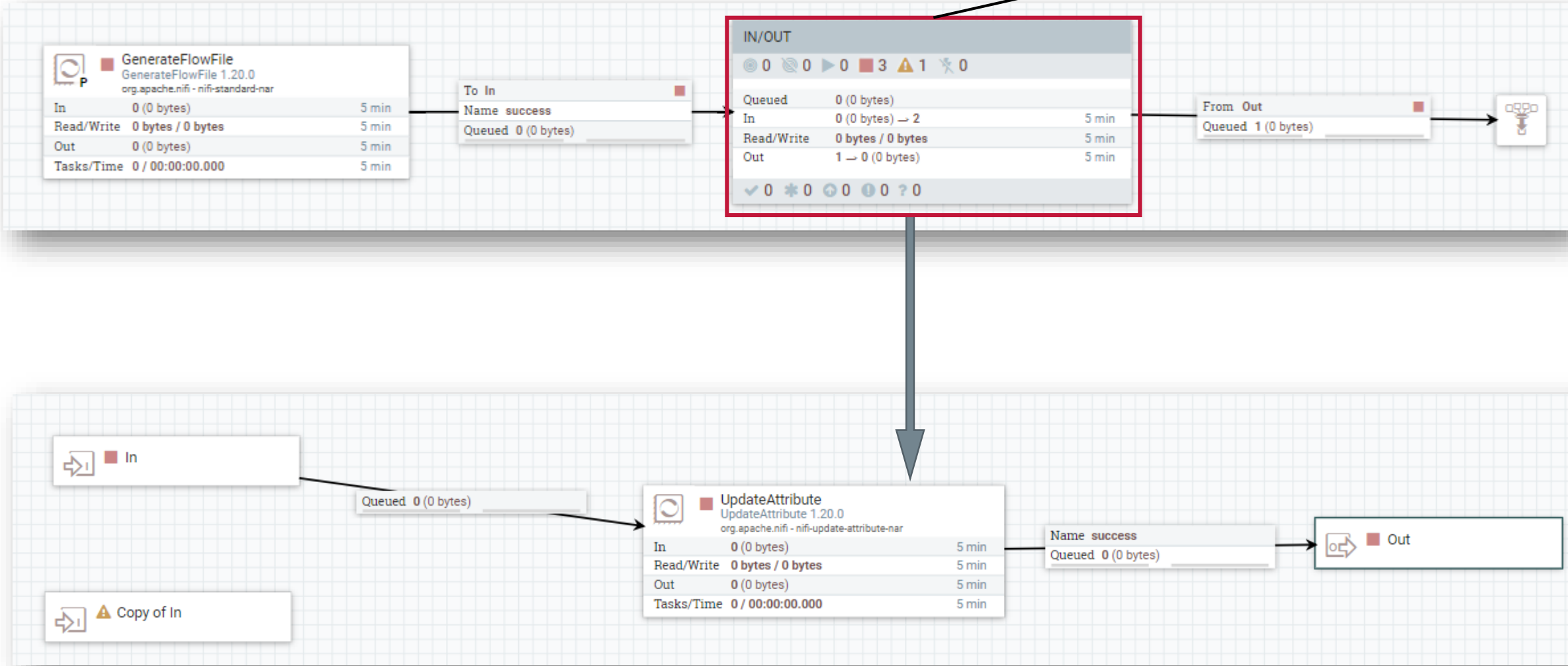
Default Back Pressure Data Size Threshold

1 GB

APPLY

# Демонстрация In/Out Ports

Processor Group



# Flow Files и атрибуты

# Flow Files. Атрибуты

- Все данные в Apache NiFi представлены абстракцией – FlowFile.
- FlowFile состоит из двух основных частей: содержимого и атрибутов.
- Информация о FlowFile доступна при нажатии на иконку info на странице «List queue» очереди потока
- Core атрибуты: (<https://nifi.apache.org/docs/nifi-docs/html/developer-guide.html#flowfile>)
  - Filename (filename): Имя файла FlowFile. Имя файла не должно содержать никакой структуры каталога.
  - UUID (uuid): Универсальный уникальный идентификатор (UUID), присвоенный этому FlowFile.
  - Path (path): относительный путь для каталога, в котором расположен FlowFile.
  - Absolute Path (absolute.path): абсолютный путь для каталога, в котором расположен FlowFile.
  - Priority (priority): Числовое значение, указывающее на приоритет FlowFile.
  - MIME Type (mime.type): Тип MIME этого FlowFile.
  - Discard Reason (discard.reason): Указывает причину, по которой FlowFile отбрасывается.
  - Alternate Identifier (alternate.identifier): альтернативный идентификатор, отличный от UUID.

The image displays two overlapping screenshots of the Apache NiFi FlowFile details interface. The left screenshot shows the 'Attribute Values' tab, and the right screenshot shows the 'Details' tab.

**FlowFile Details (Left Screenshot):**

- Attribute Values**
- filename: de82686a-117d-4e65-9bfd-48dd7840edc0
- path: /
- uuid: de82686a-117d-4e65-9bfd-48dd7840edc0

**FlowFile Details (Right Screenshot):**

- FlowFile Details**
- UUID: de82686a-117d-4e65-9bfd-48dd7840edc0
- Filename: de82686a-117d-4e65-9bfd-48dd7840edc0
- File Size: 10 bytes
- Queue Position: No value set
- Queued Duration: 01:29:02.701
- Lineage Duration: 01:29:06.321
- Penalized: No
- Node Address: ads-nifi-01-mnode2.ru-central1.internal:9090

**Content Claim (Right Screenshot):**

- Container: repo1
- Section: 1
- Identifier: 1706792593011-1
- Offset: 790
- Size: 10 bytes

Buttons: DOWNLOAD, VIEW, OK

# Подключение источников (File, Apache Kafka, СУБД (ADB))

# File flow

- Разбить файл на строки по определенному правилу.
- Сохранить строки в соответствии с тегом.

PutFile (put, local, copy, archive, files, filesystem) - NiFi processor that writes the contents...

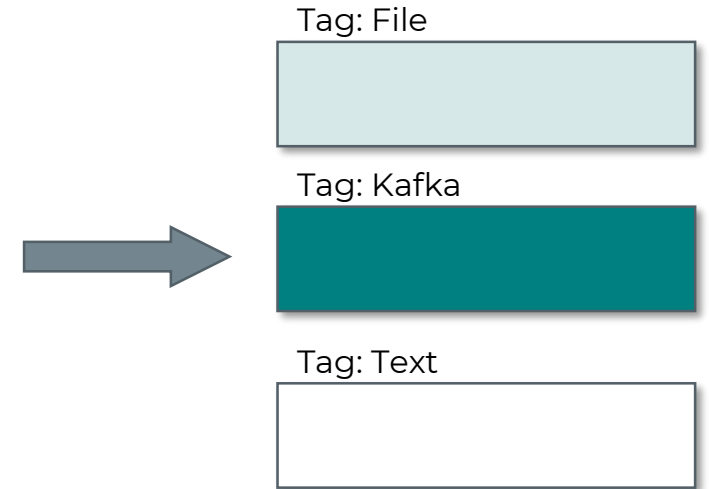
ConsumeKafka (kafka, get, ingest, ingress, topic, pubsub, consume) - NiFi processor that ...

GetFile (local, files, filesystem, ingest, ingress, get, source, input) - NiFi processor that creates ...

RouteText (attributes, routing, text, regexp, regex, filter, search, detect) - NiFi processor ...

PublishKafka (kafka, put, send, message, pubsub) - NiFi processor that sends the contents ...

ReplaceText (text, update, change, replace, modify, regex) - NiFi processor that updates the ...



# File flow. Parameter Context

- Создадим Контекст с параметрами для Kafka
- Для Группы процессоров указываем контекст

### Add Parameter Context

SETTINGS PARAMETERS INHERITANCE

Name  
File Example

Description

### Update Parameter Context

SETTINGS PARAMETERS INHERITANCE

Name	Value	
dir_in	/tmp/stage	
dir_out1	tag-files	
dir_out2	/tmp/tag-kafka	
dir_out3	/tmp/tag-text	

+ Parameter dir\_out3

Referencing Components  
Pending Apply

### File Example Configuration

GENERAL CONTROLLER SERVICES

Process Group Name  
File Example

Process Group Parameter Context  
File Example

Process Group Comments

Process Group FlowFile Concurrency  
Unbounded

Process Group Outbound Policy  
Stream When Available

Default FlowFile Expiration

Last updated: 03:57:41 UTC

# File flow. GetFile

- **PROPERTIES:**
  - изменить параметр *Input Directory* (папка, из которой должен быть считан файл):  
`#{dir_in}`
  - Для контекстной подсказки, используйте **Ctrl + Space**
- На каждом узле NiFi выполните действия
  - Создать файл `/tmp/file.txt`
  - `sudo mkdir /tmp/stage`
  - `sudo chmod -R 777 /tmp/stage`
  - `sudo cp /tmp/file.txt /tmp/stage`

Configure Processor | GetFile 1.20.0

Invalid

SETTINGS SCHEDULING PROPERTIES RELATIONSHIPS COMMENTS

Required field

Property	Value	
Input Directory	#{dir_in}	→
File Filter	[^\.]*	↑
Path Filter	No value set	↑
Batch Size	10	↑
Keep Source File	false	↑
Recurse Subdirectories	true	↑
Polling Interval	0 sec	↑
Ignore Hidden Files	true	↑
Minimum File Age	0 sec	↑
Maximum File Age	No value set	↑
Minimum File Size	0 B	↑
Maximum File Size	No value set	↑



# File flow. RouteFile

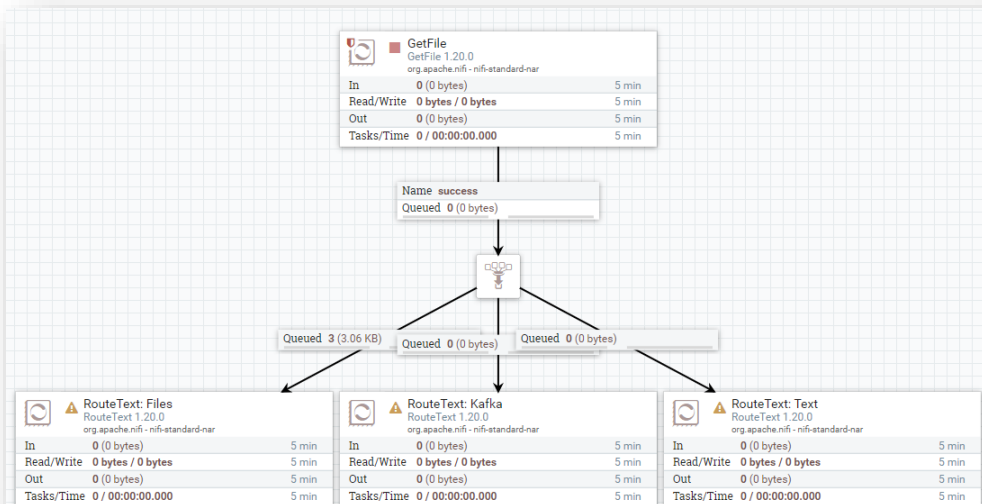
- **PROPERTIES:**

- изменить параметр *Routing Strategy* (направляет в поток **matched** все строки, соответствующие заданному условию) :  
Route to "matched" if lines matches any condition
- изменить параметр *Matching Strategy* (проверяет строки на содержание текста в указанном регулярном выражении) :  
Contains Regular Expression
- Создать пользовательское свойство *tags*:  
(.\*files.\*)

- **RELATIONSHIPS:**

- original: terminate
- unmatched: terminate

- Выполнить данные настройки для каждого процессора RouteFile в соответствии с тегами: files, kafka, text



Configure Processor | RouteText 1.20.0

Invalid

SETTINGS SCHEDULING PROPERTIES RELATIONSHIPS COMMENTS

Required field

Property	Value
Routing Strategy	Route to 'matched' if lines matches any condition
Matching Strategy	Contains Regular Expression
Character Set	UTF-8
Ignore Leading/Trailing Whitespace	true
Ignore Case	false
Grouping Regular Expression	No value set
tags	(.*kafka.*)

Configure Processor | RouteText 1.20.0

Invalid

SETTINGS SCHEDULING PROPERTIES RELATIONSHIPS COMMENTS

Automatically Terminate / Retry Relationships

matched

☐ terminate ☐ retry

Data that satisfies the required user-defined rules will be routed to this Relationship

original

☒ terminate ☐ retry

The original input file will be routed to this destination when the lines have been successfully routed to 1 or more relationships

unmatched

☒ terminate ☐ retry

Data that does not satisfy the required user-defined rules will be routed to this Relationship

# File flow. PutFile

## PROPERTIES:

- изменить параметр *Directory* (папка, для записи файла): `#{dir_out1}`
- изменить параметр *Conflict Resolution Strategy* (действие, если файл с таким именем уже существует в выходном каталоге): `replace`
- изменить параметр *Create Missing Directories* (создание целевого каталога): `true`

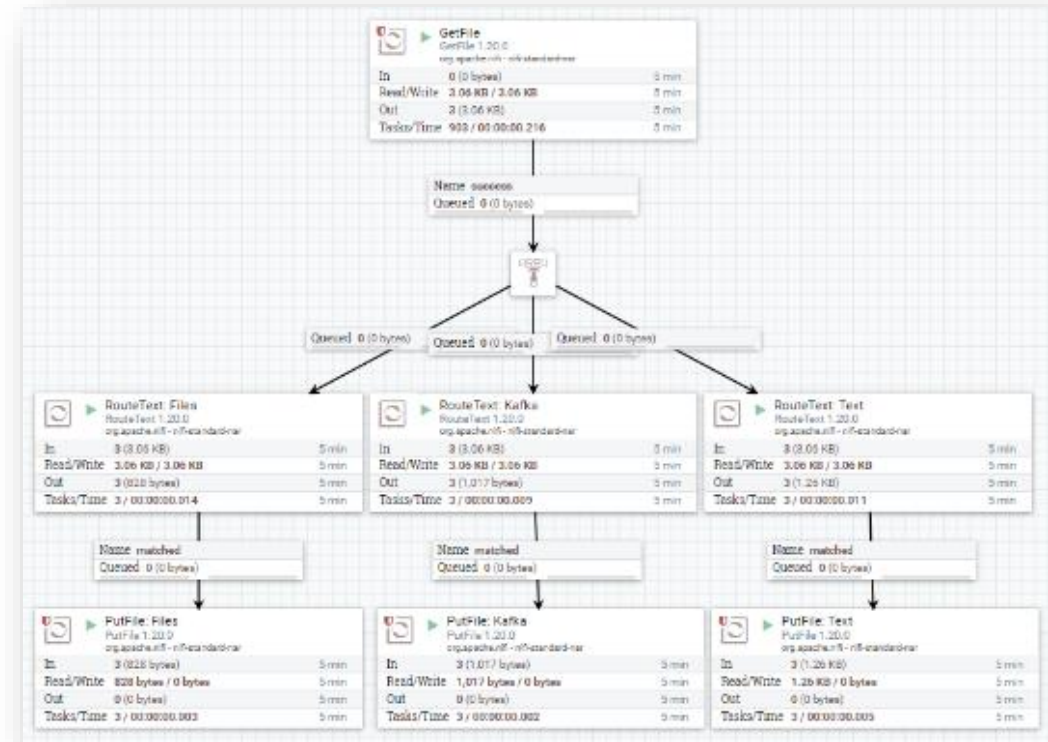
## RELATIONSHIPS:

- success: `terminate`
- failure: `terminate`

При создании соединения отметить RouteFile → PutFile отметить поток **matched**

На каждом узле NiFi выполнить действия:

- `sudo rm -rf /tmp/tag*`
- `cat /tmp/tag-*/file.txt`



## Configure Processor | PutFile 1.20.0

Invalid

SETTINGS SCHEDULING PROPERTIES RELATIONSHIPS COMMENTS

Required field

Property	Value
Directory	#{dir_out1}
Conflict Resolution Strategy	replace
Create Missing Directories	true
Maximum File Count	No value set
Last Modified Time	No value set
Permissions	No value set
Owner	No value set
Group	No value set

## Configure Processor | PutFile 1.20.0

Invalid

SETTINGS SCHEDULING PROPERTIES RELATIONSHIPS COMMENTS

Automatically Terminate / Retry Relationships

failure

☒ terminate ☐ retry

Files that could not be written to the output directory for some reason are transferred to this relationship

success

☒ terminate ☐ retry

Files that have been successfully written to the output directory are transferred to this relationship

## Create Connection

DETAILS SETTINGS

From Processor

File processors search  
RouteText

Within Group  
NIFI Flow

For Relationships

☒ matched

☐ original

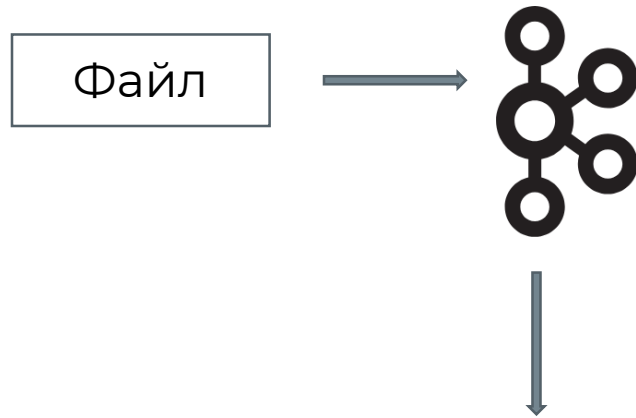
☐ unmatched

To Processor

File processors  
PutFile

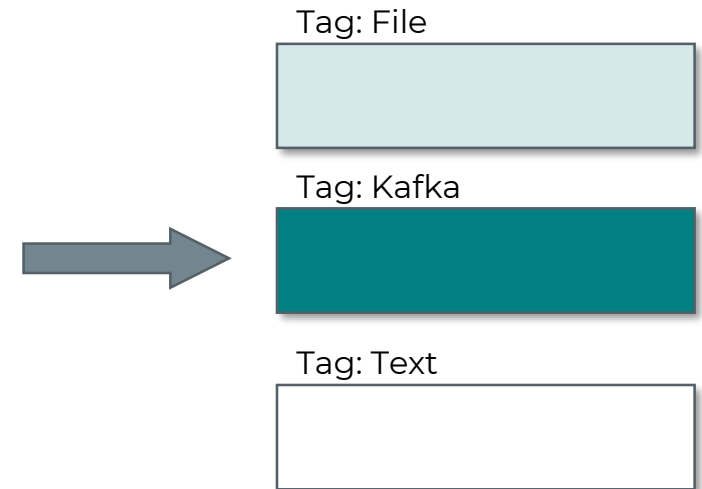
Within Group  
NIFI Flow

# Kafka PublishConsume flow



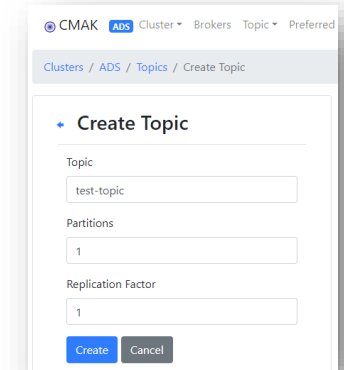
- Опубликовать файл в топик Kafka.
- Считать строки из топика Kafka по определенному правилу.
- Сохранить строки в соответствии с тегом.

PutFile (put, local, copy, archive, files, filesystem) - NiFi processor that writes the contents...  
ConsumeKafka (kafka, get, ingest, ingress, topic, pubsub, consume) - NiFi processor that ...  
GetFile (local, files, filesystem, ingest, ingress, get, source, input) - NiFi processor that creates ...  
RouteText (attributes, routing, text, regexp, regex, filter, search, detect) - NiFi processor ...  
PublishKafka (kafka, put, send, message, pubsub) - NiFi processor that sends the contents ...  
ReplaceText (text, update, change, replace, modify, regex) - NiFi processor that updates the ...



# Kafka PublishConsume flow. Parameter Context

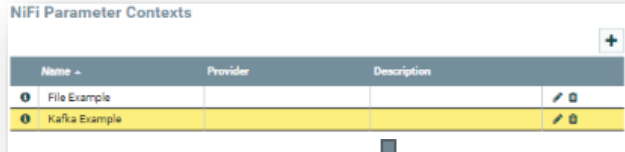
- Создадим топик (CMAK): test-topic



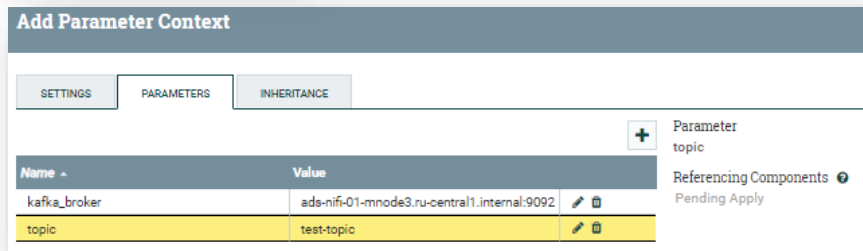
- Создать *Group ID*—идентификатор группы для определения потребителей:

```
/usr/lib/kafka/bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic test-topic --group nifi-user
```

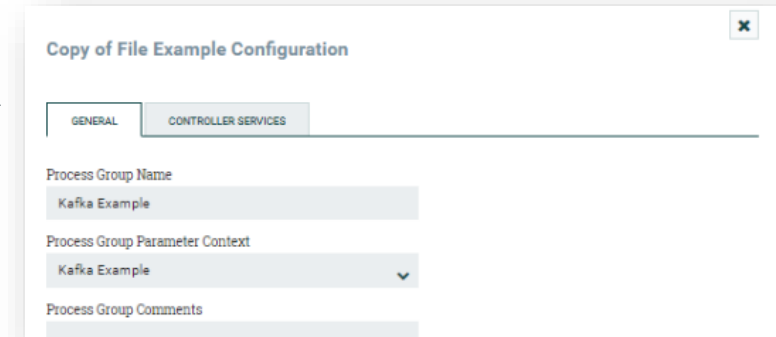
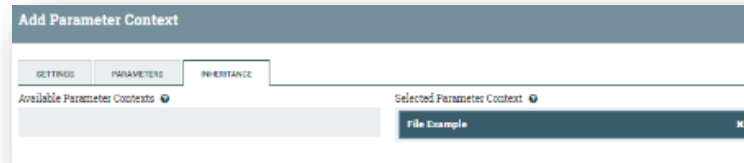
- Создадим новый контекст для параметров Kafka-процессоров, используя наследование. Процессорной группе назначим данный контекст



Name	Provider	Description
File Example		
Kafka Example		



Name	Value
kafka_broker	ads-nifi-01-mnode3.ru-central1.internal:9092
topic	test-topic



# Kafka PublishConsume flow. PublishKafka. ConsumeKafka

- **PROPERTIES:**
  - изменить параметр *Kafka Brokers*: `#{kafka_broker}`
  - изменить параметр *Topic Name*: `#{topic}`
- **RELATIONSHIPS:**
  - success: `terminate`
  - failure: `terminate`

Configure Processor | PublishKafka\_2\_6120.0

Stopped

SETTINGS

SCHEDULING

PROPERTIES

RELATIONSHIPS

COMMENTS

Required field

Property	Value	
Kafka Brokers	#{kafka_broker}	→
Topic Name	#{topic}	→
Use Transactions	true	⌵
Transactional Id Prefix	No value set	⌵
Message Demarcator	No value set	⌵
Failure Strategy	Route to Failure	⌵
Delivery Guarantee	Guarantee Replicated Delivery	⌵
Attributes to Send as Headers (Regex)	No value set	⌵
Message Header Encoding	UTF-8	⌵
Security Protocol	PLAINTEXT	⌵
SASL Mechanism	GSSAPI	⌵
Kerberos Credentials Service	No value set	⌵

- **PROPERTIES:**
  - изменить параметр *Kafka Brokers*: `#{kafka_broker}`
  - изменить параметр *Topic Name*: `#{topic}`
  - изменить параметр *Group ID*: `nifi-user`

Configure Processor | ConsumeKafka\_2\_6120.0

Invalid

SETTINGS

SCHEDULING

PROPERTIES

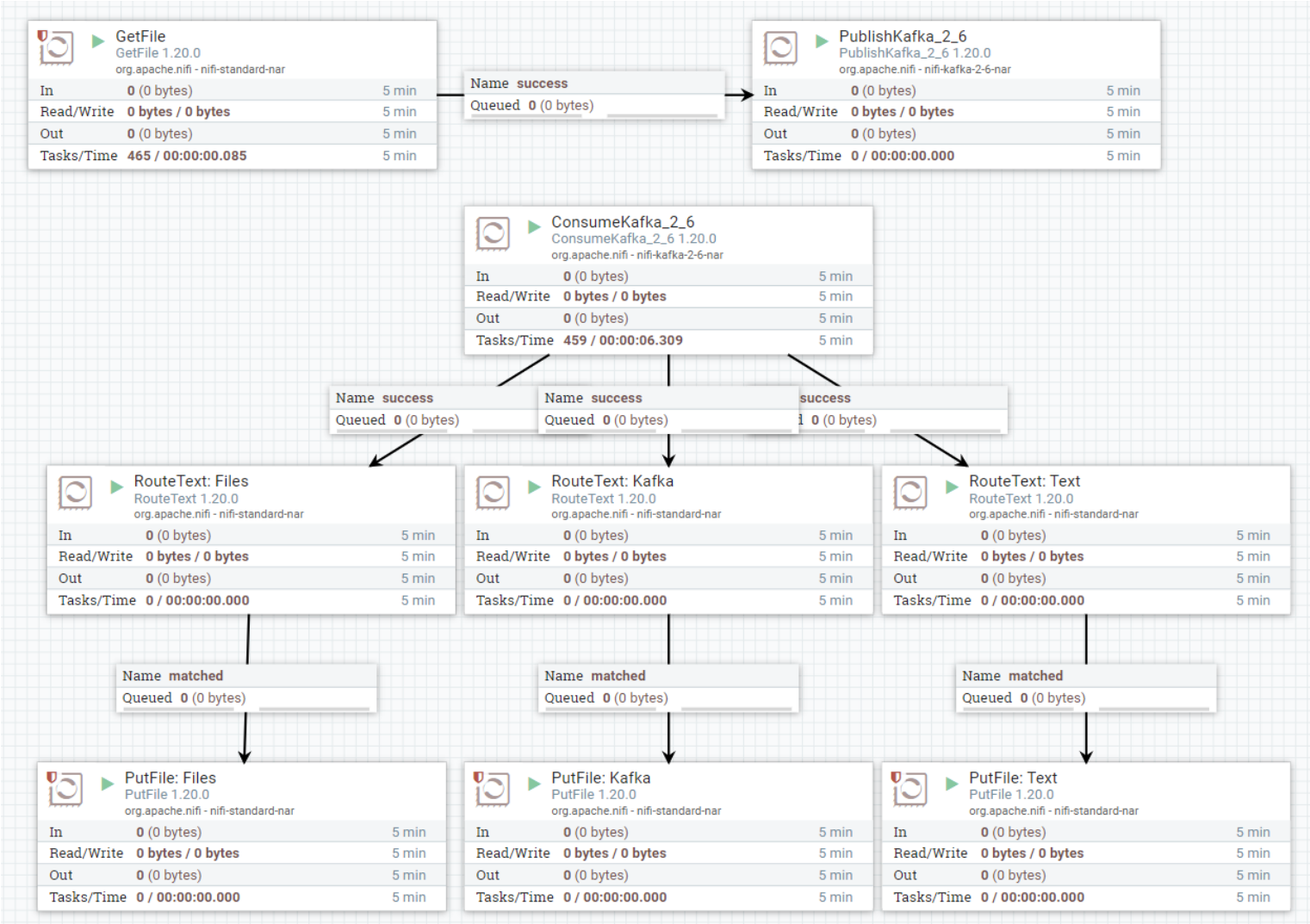
RELATIONSHIPS

COMMENTS

Required field

Property	Value	
Kafka Brokers	#{kafka_broker}	→
Topic Name(s)	#{topic}	→
Topic Name Format	names	⌵
Group ID	nifi-user	⌵
Commit Offsets	true	⌵
Max Uncommitted Time	1 secs	⌵
Honor Transactions	true	⌵
Message Demarcator	No value set	⌵
Separate By Key	false	⌵
Security Protocol	PLAINTEXT	⌵
SASL Mechanism	GSSAPI	⌵
Verbose Producer Logs	No value set	⌵

# Kafka PublishConsume flow



# Лабораторная работа

- Создать и запустить File flow
- Создать и запустить Kafka flow

# ADB GetPut flow

Файл



- Загрузить файл в таблицу ADB.
- Считать строки из таблицы ADB по определенному правилу.
- Сохранить строки в соответствии с тегом.



PutFile (put, local, copy, archive, files, filesystem) - NiFi processor that writes the contents...

ConsumeKafka (kafka, get, ingest, ingress, topic, pubsub, consume) - NiFi processor that ...

GetFile (local, files, filesystem, ingest, ingress, get, source, input) - NiFi processor that creates ...

RouteText (attributes, routing, text, regexp, regex, filter, search, detect) - NiFi processor ...

PublishKafka (kafka, put, send, message, pubsub) - NiFi processor that sends the contents ...

ReplaceText (text, update, change, replace, modify, regex) - NiFi processor that updates the ...



Tag: File



Tag: Kafka



Tag: Text





# ADB GetPut flow. Context. Variables. DBCPConnectionPool

- Создадим Контекст с параметрами для ADB, унаследовав от File Example.
- Для Группы процессоров указываем контекст
- Создаем список переменных для реквизитов подключения к ADB
- Используем созданные переменные в Контексте

NiFi Parameter Contexts

Name ^
ADB Example
File Example
Kafka Example

Variables

Process Group  
ADB Example

Scope	Name ^	Value
ADB Example	dev_driver_class	org.postgresql.Driver
ADB Example	dev_driver_location	/tmp/postgresql-42.7.1.jar
ADB Example	dev_jdbc_url	jdbc:postgresql://10.130.0.9:5432/adb
ADB Example	dev_user_name	gpadmin
ADB Example	dev_user_password	Empty string set

- Подключение DBCPConnectionPool 1.20.0 для Группы процессоров

ADB Example Configuration

GENERAL CONTROLLER SERVICES

Name ^	Type	Bundle	State	Scope
ADB Connectio...	DBCPConnecti...	org.apache.nifi ...	Invalid	ADB Example

Update Parameter Context

SETTINGS PARAMETERS INHERITANCE

Name ^	Value
driver_class	\${dev_driver_class}
driver_location	\${dev_driver_location}
jdbc_url	\${dev_jdbc_url}
user_name	\${dev_user_name}
user_password	\${dev_user_password}
dir_in	/tmp/stage
dir_out1	/tmp/tag-files
dir_out2	/tmp/tag-kafka
dir_out3	/tmp/tag-text

Parameter driver\_class

Referencing Components

None

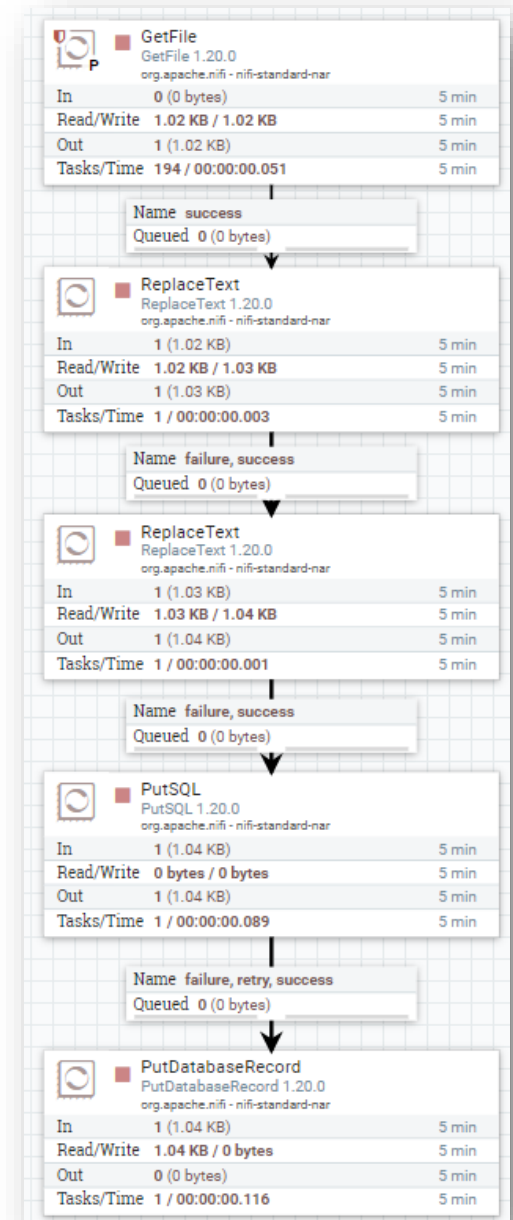
# ADB GetPut flow. File to ADB

- GetFile – считывает данные из файла.
- ReplaceText(1) – добавляет “” к строке
- ReplaceText(1) – добавляет заголовок
- PutSQL – создает таблицу в ADB и передает flowfile дальше:  
CREATE TABLE IF NOT EXISTS test\_table(descr text) WITH (appendoptimized=true);
- PutDatabaseRecord – записывает содержимое flowfile в таблицу ADB.

```
select * from test_table;
```

descr

-----  
-----  
-----  
GetFile (local, files, filesystem, ingest, ingress, get, source, input) - NiFi processor that creates FlowFiles from files in a directory.  
ReplaceText (text, update, change, replace, modify, regex) - NiFi processor that updates the content of a FlowFile by evaluating a Regular Expression (regex) against it and replacing the section of the content that matches the Regular Expression with some alternate value.  
PutFile (put, local, copy, archive, files, filesystem) - NiFi processor that writes the contents of a FlowFile to the local file system.  
ConsumeKafka (kafka, get, ingest, ingress, topic, pubsub, consume) - NiFi processor that consumes messages from Apache Kafka specifically built against the Kafka Consumer API.  
RouteText (attributes, routing, text, regexp, regex, filter, search, detect) - NiFi processor that routes textual data based on a set of user-defined rules.  
PublishKafka (kafka, put, send, message, pubsub) - NiFi processor that sends the contents of a FlowFile as a message to Apache Kafka using the Kafka Producer API.  
(6 rows)



# ADB GetPut flow. ADB to tag-file

- ExecuteSQLRecord – считывает данные из таблицы ADB.
- Необходимо добавить CSVRecordSetWriter для записи в текстовый файл

Configure Processor | ExecuteSQLRecord 1.20.0

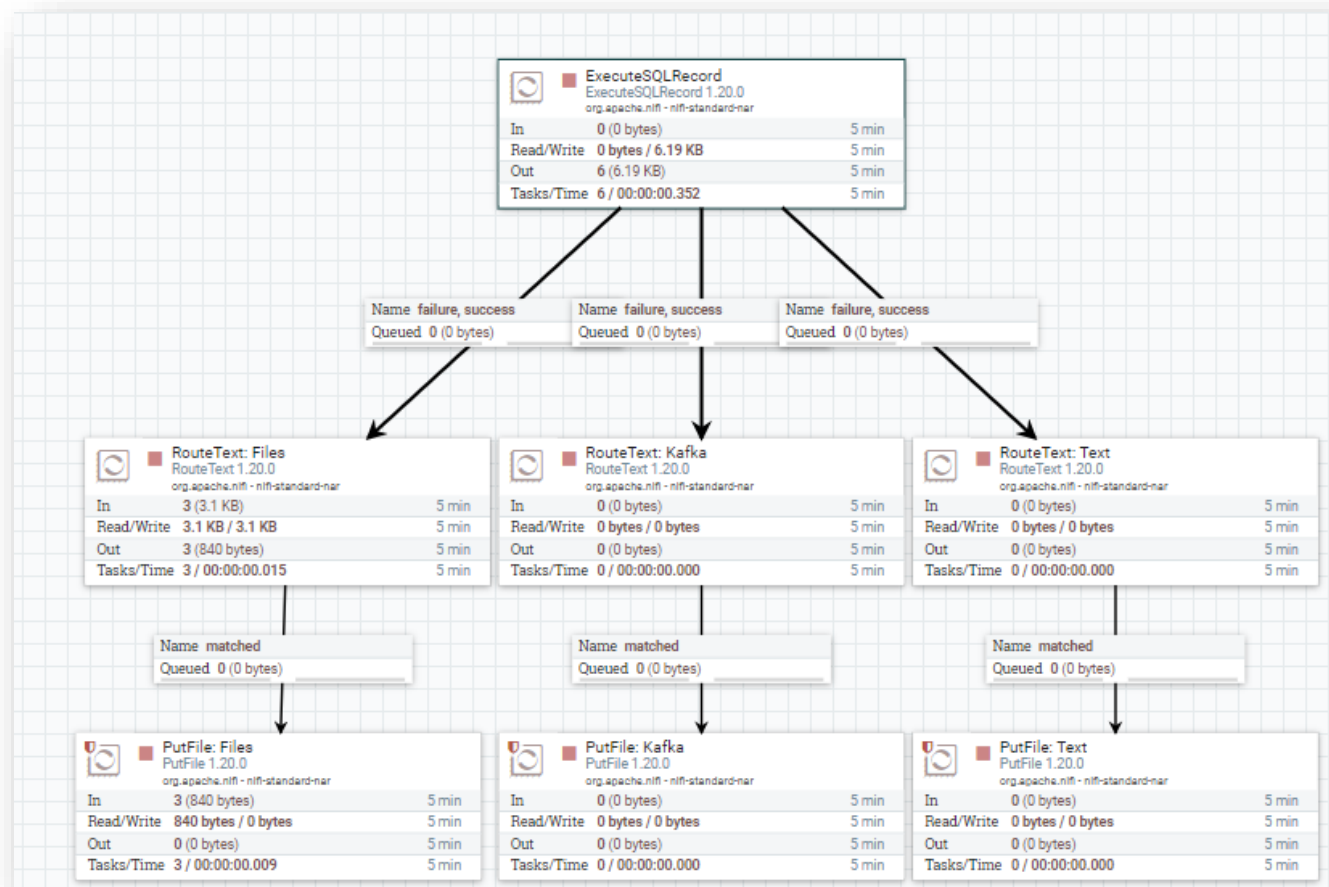
Stopped

SETTINGS SCHEDULING PROPERTIES RELATIONSHIPS COMMENTS

Required field

Property	Value
Database Connection Pooling Service	ADB ConnectionPool
SQL Pre-Query	No value set
SQL select query	SELECT * FROM test_table;
SQL Post-Query	No value set
Max Wait Time	0 seconds
Record Writer	CSVRecordSetWriter
Normalize Table/Column Names	false
Use Avro Logical Types	false
Default Decimal Precision	10
Default Decimal Scale	0
Max Rows Per Flow File	0
Output Batch Size	0

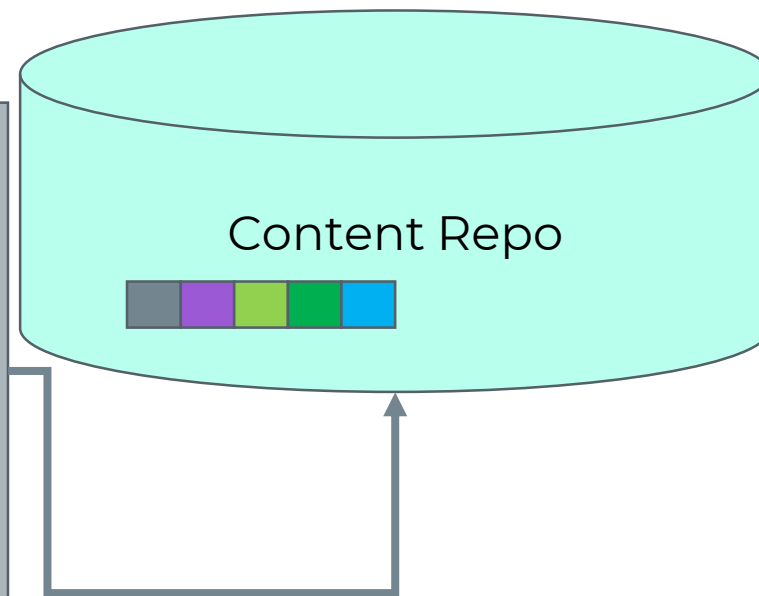
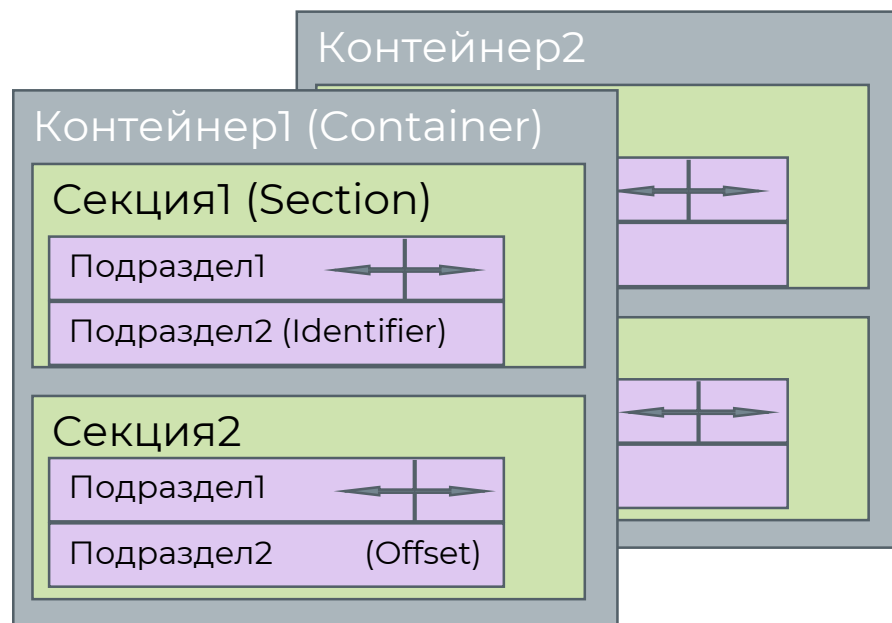
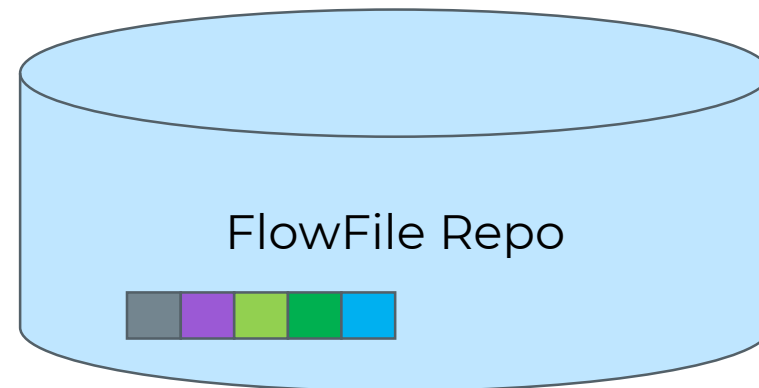
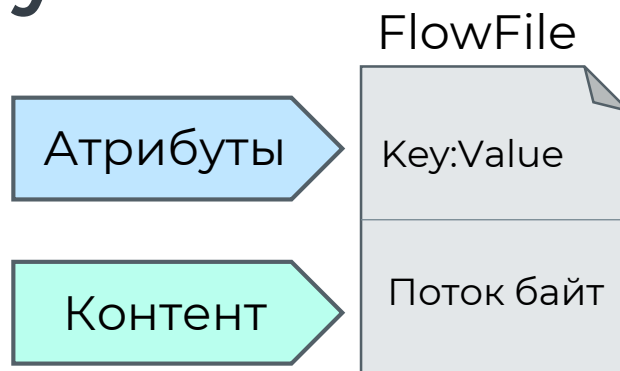
CANCEL APPLY



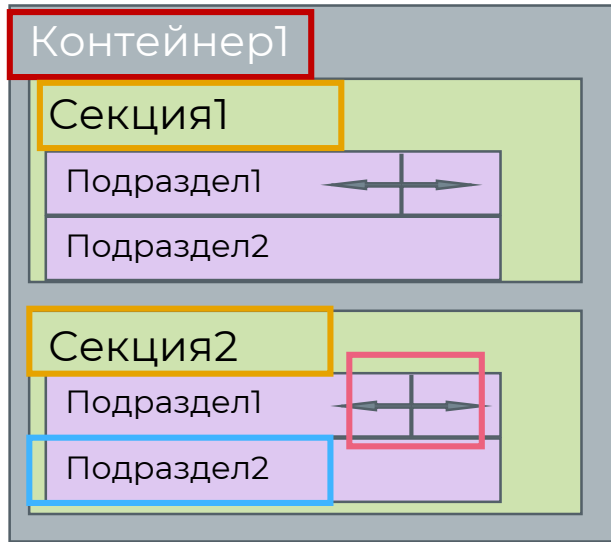
# FlowFile Repository, Content Repository, Provenance Repository: детальное изучение

# FlowFile. Repository

- Атрибуты загружаются в память
- Атрибут FlowFile хранится: WAL и hashmap в рабочей памяти JVM.
- Если FlowFile изменяется, то delta записывается в журнал WAL и изменяются объекты в памяти JVM
- Блоки репозитория – Immutable
- При изменениях атрибутов происходит их сохранение и загрузка в память.
- При изменении контента, указатель на место в репозитории контента обновляется
- В один и тот же подраздел можно записать несколько контентов flowfiles.
- Если нет ссылок из FlowFile Repo, то контент удаляется (архивируется)

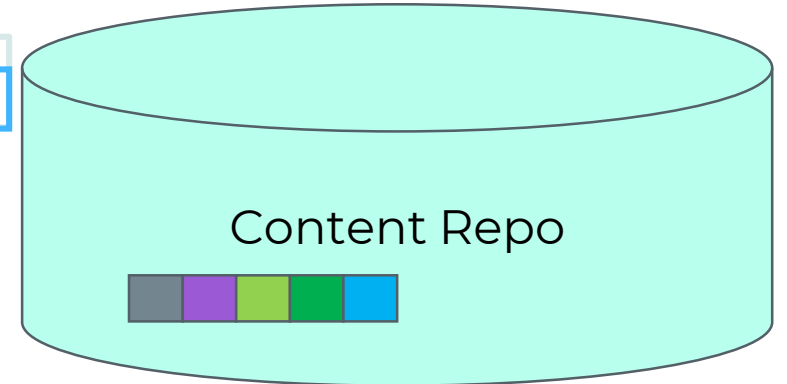
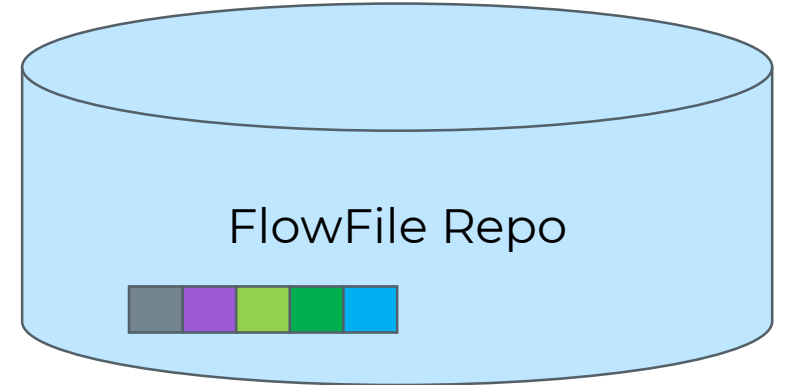


# FlowFile. Repository



```
/usr/lib/nifi-server/content_repository0/
drwxr-xr-x. 2 nifi nifi 6 Jan 31 10:49 0
drwxr-xr-x. 2 nifi nifi 6 Jan 31 10:49 1
drwxr-xr-x. 2 nifi nifi 6 Jan 31 10:49 10
drwxr-xr-x. 2 nifi nifi 6 Jan 31 10:49 100
drwxr-xr-x. 2 nifi nifi 6 Jan 31 10:49 1000
drwxr-xr-x. 2 nifi nifi 6 Jan 31 10:49 1001
drwxr-xr-x. 2 nifi nifi 6 Jan 31 10:49 1002
...
```

```
ls /usr/lib/nifi-server/content_repository1/1
-rw-r--r--. 1 nifi nifi 13806 Feb 2 19:39 1706792593011-1
```



```
cat /usr/lib/nifi-server/content_repository1/1/1706792593011-1
```

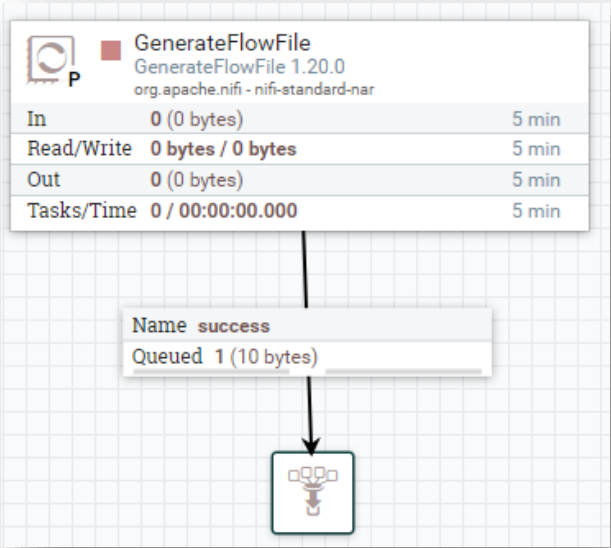
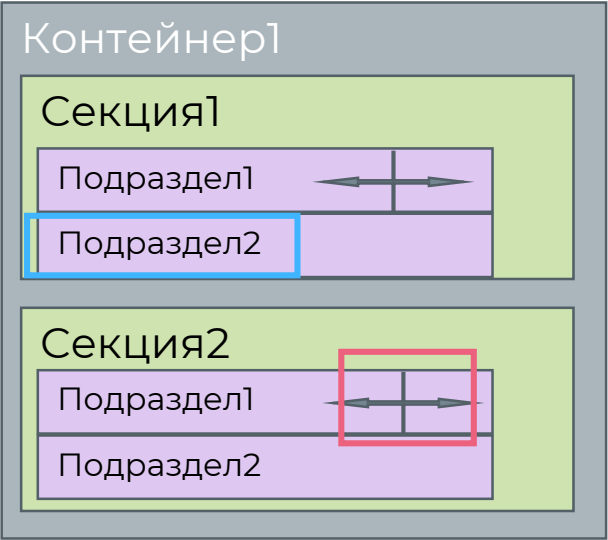
...

"PublishKafka (kafka, put, send, message, pubsub) - NiFi processor that sends the contents of a FlowFile as a message to Apache Kafka using the Kafka Producer API."

"ReplaceText (text, update, change, replace, modify, regex) - NiFi processor that updates the content of a FlowFile by evaluating a Regular Expression (regex) against it and replacing the section of the content that matches the Regular Expression with some alternate value."

0123456789

# FlowFile. Repository



FlowFile

DETAILS

ATTRIBUTES

FlowFile Details

UUID

fb0b456b-0de7-44e6-9737-1f024425ad58

Filename

fb0b456b-0de7-44e6-9737-1f024425ad58

File Size

10 bytes

Queue Position

No value set

Queued Duration

00:10:34.084

Lineage Duration

00:10:34.085

Penalized

No

Node Address

ads-nifi-01-mnode2.ru-central1.internal:9090

Content Claim

Container

repo1

Section

1

Identifier

1706792593011-1

Offset

13796

Size

10 bytes

DOWNLOAD

VIEW

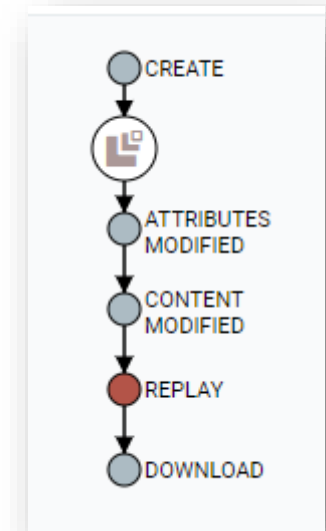
OK

nifi.content.claim.max.appendable.size:

Максимальный размер батча данных для  
сохранения на диск (content claim) (1MB)

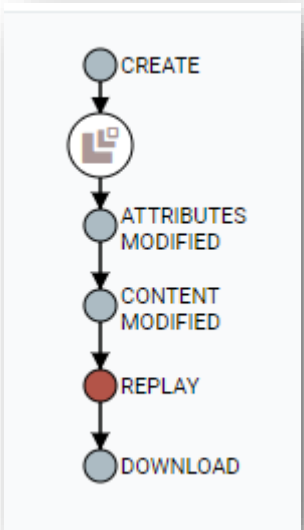
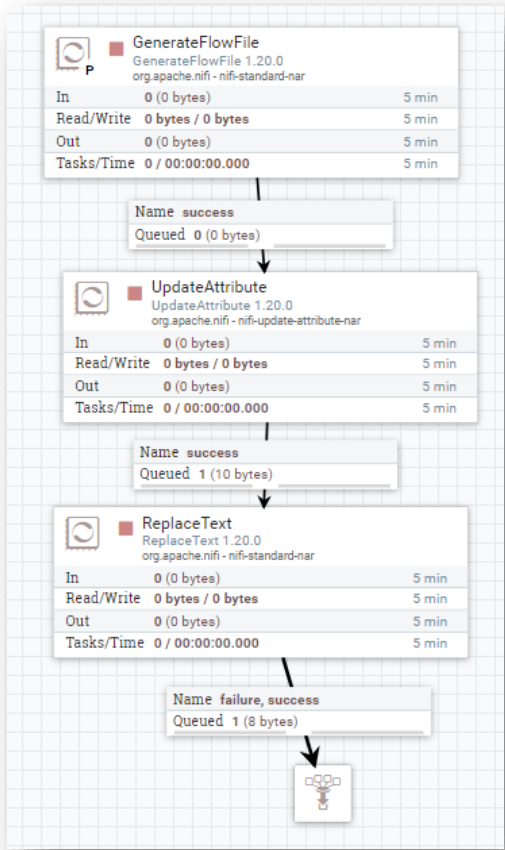
# Provenance Repository

- Репозиторий Provenance – это место, где хранится история каждого FlowFile
- Каждое событие для FlowFile (FlowFile создается, разветвляется, клонируется, изменяется и т. д.), то создается новое исходное событие в Provenance Repository.
- Исходное событие в Provenance Repository является моментальным снимком FlowFile.
- При создании исходного события он копирует все атрибуты и указатели FlowFile на содержимое FlowFile
- Provenance не копирует контент в репозиторий контента, он просто копирует указатель FlowFile на контент





# Provenance Repository



### Provenance Event

DETAILS ATTRIBUTES CONTENT

**Input Claim**  
Container  
repo0  
Section  
4  
Identifier  
1714989185469-4  
Offset  
838074  
Size  
10 bytes  
[DOWNLOAD](#) [VIEW](#)

**Output Claim**  
Container  
repo0  
Section  
1  
Identifier  
1714983900638-1  
Offset  
891461  
Size  
24 bytes  
[DOWNLOAD](#) [VIEW](#)

**Replay**  
Connection Id  
7f93a63f-c245-3a09-aaa0-9f2a957e14c4  
[REPLAY](#)

OK

### Provenance Event

DETAILS ATTRIBUTES CONTENT

**Attribute Values** ☒ Show modified attributes only  
filename  
newfile  
755695d9-3e0a-4166-a602-8a926d4fd46e (previous)

# Производительность и оптимизация потоков данных

# Производительность и оптимизация потоков данных

- `nifi.bored.yield.duration` – количество времени, которое компонент будет ждать перед проверкой, есть ли у него новые данные для работы (10 мс)

Меньшие значения соответствуют меньшей задержке, но более высокой загрузке ЦП. Таким образом, в зависимости от того, насколько важна задержка для вашего общего потока данных, увеличение этого значения еще больше снизит общую загрузку ЦП.

- `nifi.queue.swap.threshold` – порог очереди, при котором NiFi начинает swap информации FlowFile на диск (20000).

Если порог очереди, то нужно увеличить Heap size

- `nifi.content.repository.directory` – путь хранилищ контента  
Желательно назначать уникальные имена для каждой ноды (ADCM нет возможности)
- `nifi.provenance.repository.query.threads` – количество потоков, используемых для запросов к репозиторию Provenance (2)

Может потребоваться увеличить количество потоков, если пользователей будет много.

- `nifi.provenance.repository.index.threads` – количество потоков, используемых для индексации событий Provenance, чтобы их можно было найти(2)

Индексации событий Provenance влияет на скорость основного потока данных

- `nifi.provenance.repository.index.shard.size` – большие значения размера сегмента приведут к большему использованию java heap при поиске в репозитории Provenance, но должны обеспечить более высокую производительность (500 МБ)

`nifi.provenance.repository.index.shard.size = 0,5 * nifi.provenance.repository.max.storage.size` (максимальный объем информации о происхождении данных, который можно хранить одновременно).

- JVM

Про G1 GC!!! Стал под запретом, но нет явных причин для производительных узлов отказываться от него!

`java.arg.codecachesize=-XX:ReservedCodeCacheSize=256m` – память, отдельная от кучи, содержащая весь байт-код JVM для метода, скомпилированного в машинный код, при переполнении кэша нужно перезапустить JVM.

# Управление репликацией и балансировкой нагрузки

## Репликация данных:

- **Site-to-Site (S2S):** NiFi поддерживает репликацию данных между разными экземплярами NiFi с использованием протокола Site-to-Site (позволяет передавать потоки данных между различными кластерами NiFi).
- **Remote Process Groups (RPG):** С помощью RPG можно настроить отправку и получение данных от удаленных NiFi кластеров.

## Балансировка нагрузки:

- **Connection Load Balancing:** NiFi позволяет балансировать нагрузку на уровне соединений, распределяя очереди данных между узлами кластера.
- **Data Prioritization:** Вы можете настроить приоритеты для различных типов данных, чтобы управлять порядком их обработки:
  - **FirstInFirstOutPrioritizer:** Обработка данных в порядке их поступления.
  - **NewestFlowFileFirstPrioritizer:** Приоритет новым данным.
  - **OldestFlowFileFirstPrioritizer:** Приоритет старым данным.
  - **PriorityAttributePrioritizer:** Приоритет на основе атрибута priority.

# Производительность и оптимизация потоков данных

- Run Duration – определяющий длительность повторных запусков Процессора.

При запуске процессора, открывается транзакция чтения из очереди. Run Duration позволяет экономить на количестве запусков процессора, если в очередь медленно поступают данные.

- Load Balancing:
  - **Do not balance** — ничего не делаем, обрабатываем данные на тех нодах кластера, куда они изначально попали.
  - **Partition by attribute** — распределяем данные по значению выбранного flow file attribute, файлы с одинаковым значением атрибута гарантированно распределяются на одну ноду.
  - **Round Robin** — распределение flow-файлов *равномерно* по всем нодам.
  - **Single node** — все файлы едут на одну ноду, но на какую именно, неизвестно.

Ставить балансировку на каждую очередь плохой кейс!

Вначале “раскидываем” flowfiles, а потом уже не используем балансировку.

Для процессоров Merge со стратегией Defragment, желательно использовать

**Partition by attribute** для атрибута корреляции.

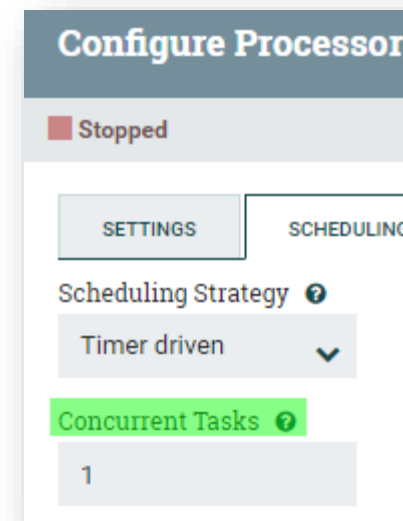
The screenshot shows the 'SCHEDULING' tab with the following settings: 'Scheduling Strategy' is set to 'Timer driven'; 'Concurrent Tasks' is set to '1'; 'Run Schedule' is set to '1 min'; 'Run Duration' is set to '0ms' with a slider ranging from '0ms' to '2s' and labels 'Lower latency' and 'Higher throughput'; 'Execution' is set to 'Primary node'.

The screenshot shows the 'SETTINGS' tab with the following settings: 'Name' is empty; 'Id' is '3bf440e4-fd86-1fe6-0000-00004b8f441e'; 'FlowFile Expiration' is '0 sec'; 'Back Pressure Object Threshold' is '10000'; 'Size Threshold' is '1 GB'; 'Load Balance Strategy' is set to 'Do not load balance' from a dropdown menu; 'Available Prioritizers' includes 'FirstInFirstOutPrioritizer', 'NewestFlowFileFirstPrioritizer', 'OldestFlowFileFirstPrioritizer', and 'PriorityAttributePrioritizer'; 'Selected Prioritizers' is empty. 'CANCEL' and 'APPLY' buttons are at the bottom right.

# Производительность и оптимизация потоков данных

- Maximum Timer Driven Thread Count – максимальное количество потоков для процессоров доступных в системе.
- Maximum Timer Driven Thread Count =  $[2-4] * \text{vCPU}$  (лучше 2)
- **Concurrent Tasks** — параметр, определяющий количество используемых потоков для процессора. То есть если у вас  $n$  нод, а параметр процессора concurrent task установлен в значение  $m$ , максимально возможное число потоков для этого процессора =  $n*m$ .

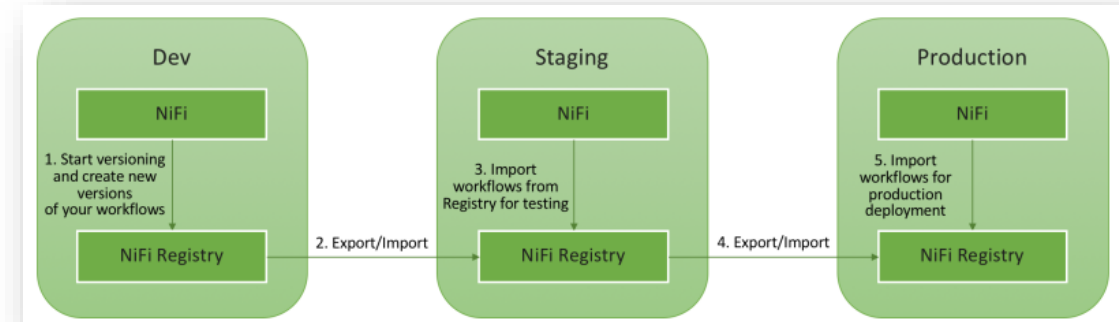
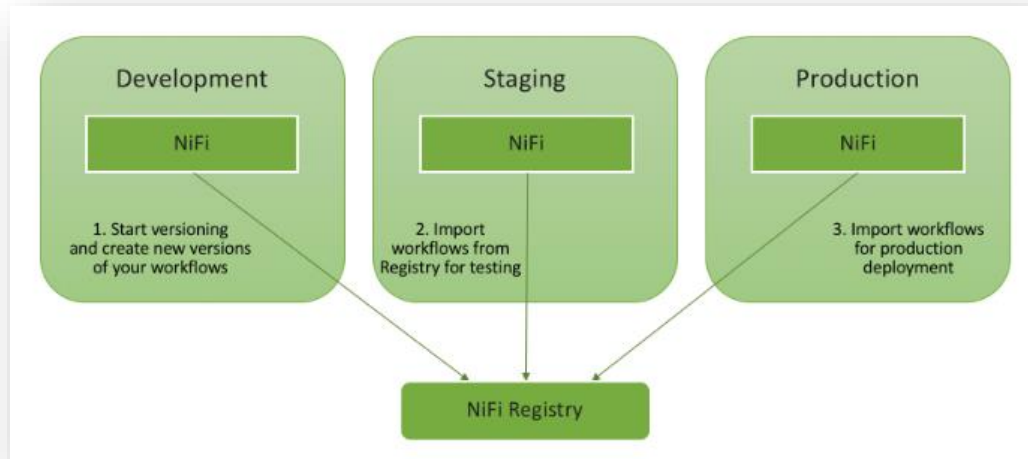
Контролирует, сколько FlowFiles должно быть обработано этим процессором одновременно.



# NiFi Registry

# NiFi Registry

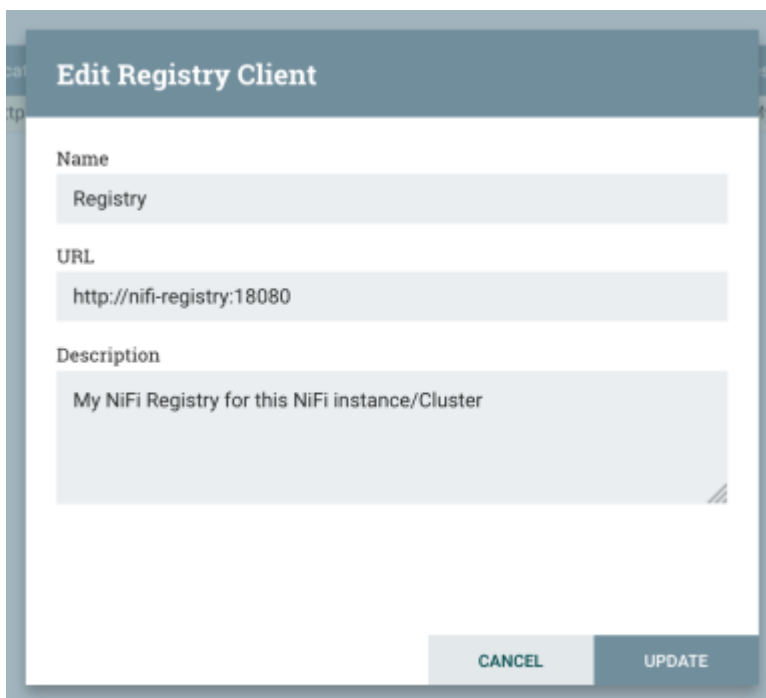
- *Apache NiFi Registry — подпроект Apache NiFi, представляющий инструмент для хранения flow и управления версиями.*
- *Flow для хранения объединяется в process group и в таком виде хранится в registry*



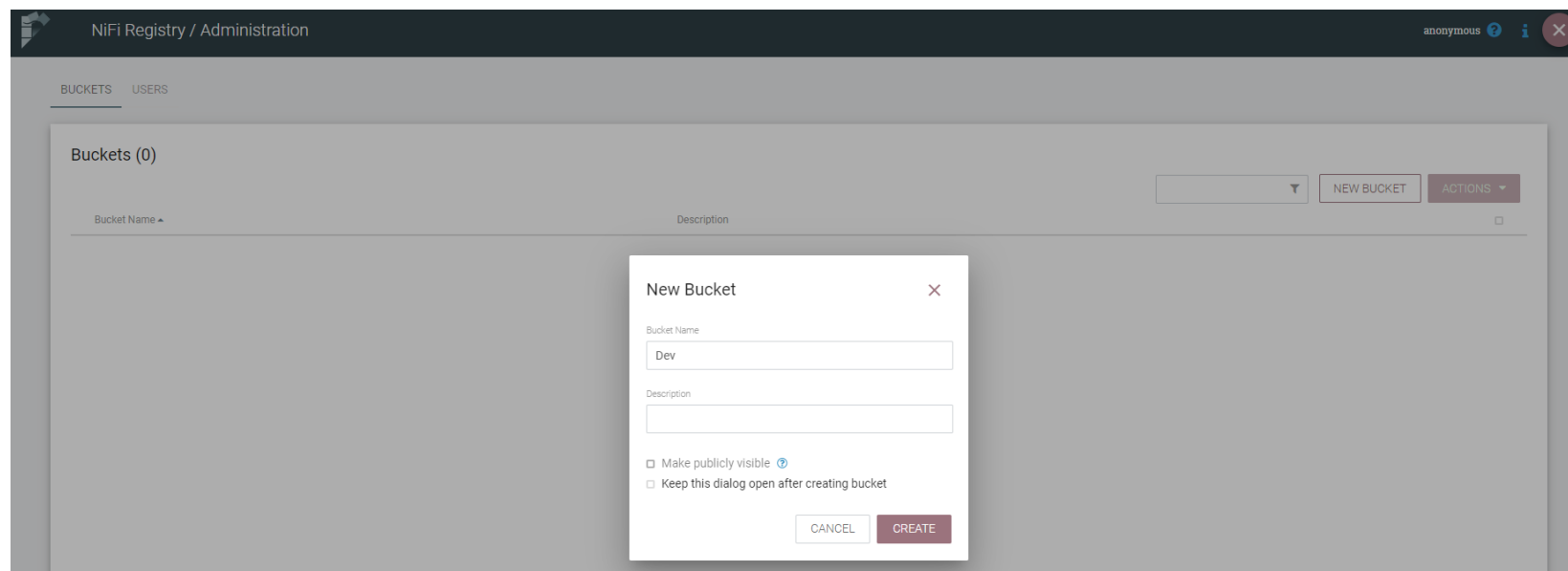


# NiFi Registry

- Необходимо настроить клиента (Controller Settings → Registry Client).
- Создать Bucket для публикации версий Процессорных Групп



The screenshot shows the 'Edit Registry Client' dialog box. It has a title bar 'Edit Registry Client'. Below it, there are three input fields: 'Name' with the value 'Registry', 'URL' with the value 'http://nifi-registry:18080', and 'Description' with the value 'My NiFi Registry for this NiFi instance/Cluster'. At the bottom right, there are two buttons: 'CANCEL' and 'UPDATE'.



The screenshot shows the 'NiFi Registry / Administration' interface. The 'BUCKETS' tab is selected. A 'New Bucket' dialog box is open in the foreground. The dialog has a title bar 'New Bucket' and a close button. It contains two input fields: 'Bucket Name' with the value 'Dev' and 'Description'. Below these fields, there are two checkboxes: 'Make publicly visible' (checked) and 'Keep this dialog open after creating bucket' (unchecked). At the bottom right, there are two buttons: 'CANCEL' and 'CREATE'.

# NiFi Registry

- Включаем версионирование для Процессорной группы, с указанием Bucket .
- При изменений внутри группы, появляется возможность сделать Commit

