Desarrollo de un sistema de monitorización y alertas en tiempo real para la detección de eventos críticos en la red Ethereum.

Trabajo Final de Grado

Autores:
- Denis Amilcar Giménez Álvarez.
- Matías Antonio Sosa Ramos.

Asesor: PhD. Marcos Villagra.

CONTEXTO Y ENFOQUE DEL TRABAJO

- **Línea:** sistemas distribuidos, blockchain y observabilidad.
- **Problema:** monitoreo operativo de Ethereum.
- **Enfoque**: arquitectura reproducible + herramientas open source.
- **Aplicación:** operadores de nodos y equipos de investigación.

ESTRUCTURA DE LA PRESENTACIÓN

1. Problema, motivación y objetivos.
2. Marco teórico.
3. Sistema de monitoreo propuesto.
4. Herramientas desarrolladas.
5. Resultados, limitaciones y trabajo futuro.
6. Conclusiones y preguntas.

ESTRUCTURA DE LA PRESENTACIÓN

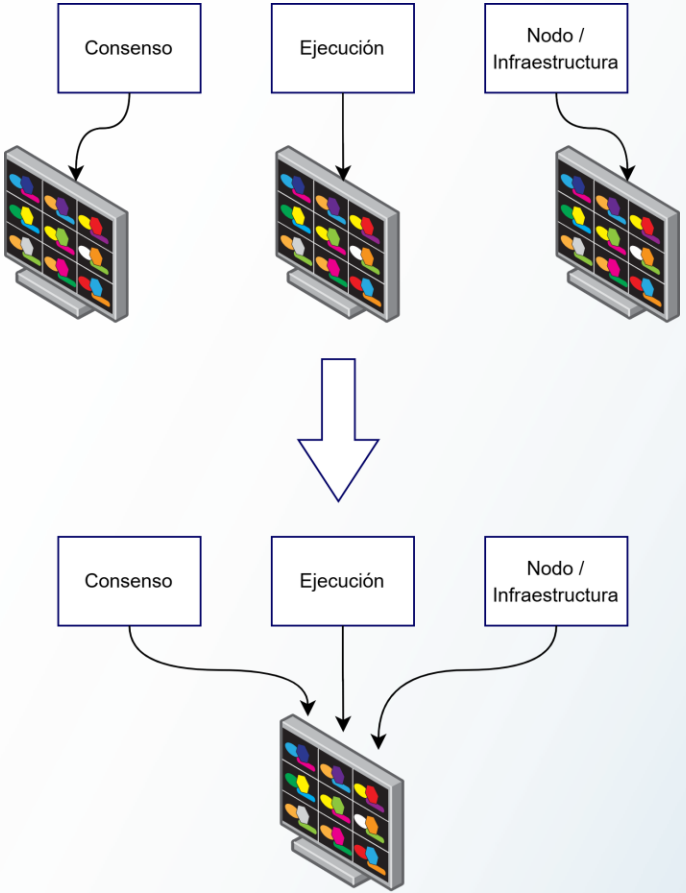
1. **Problema, motivación y objetivos.**
2. Marco teórico.
3. Sistema de monitoreo propuesto.
4. Herramientas desarrolladas.
5. Resultados, limitaciones y trabajo futuro.
6. Conclusiones y preguntas.

ETHEREUM POST-MERGE: POR QUÉ IMPORTA MONITOREAR

- Ethereum como infraestructura crítica (finanzas, logística, infraestructura digital).
- Transición a Proof-of-Stake y arquitectura en dos capas.
- Mayor complejidad operativa tras la transición al PoS.
- Incidentes inherentes al PoS.

PROBLEMA DE OBSERVABILIDAD

- Dificultad para correlacionar eventos de consenso, ejecución y salud del nodo.
- Escasez de conectores abiertos y mantenidos para la API de la capa de consenso (Beacon API).
- Integraciones a medida costosas y poco reproducibles.
- Paneles tradicionales centrados en métricas de infraestructura genéricas.



CONSECUENCIAS DEL PROBLEMA

- Mayor riesgo de no detectar eventos críticos a tiempo.
- Dificultad para entender incidentes y tomar decisiones.
- Riesgos económicos.
- Mayor esfuerzo manual para investigar comportamientos anómalos.

PROBLEMA Y ALCANCE DEL TRABAJO

Problema y Alcance

- Necesidad de un sistema reproducible de monitoreo para Ethereum.
- Capaz de integrar eventos de la red de consenso, métricas de ejecución y salud del nodo.
- Orientado a operadores y equipos de investigación.

Alcance / No-alcance

Alcance:

- Diseño de una arquitectura de referencia open source.
- Desarrollo de componentes específicos.
- Construcción de tableros y alertas.

No-alcance:

- No se diseña un protocolo de consenso nuevo.
- No se implementa un cliente de Ethereum desde cero.
- No se pretende cubrir todas las posibles métricas o todas las redes.

OBJETIVO GENERAL

- Diseñar y materializar un sistema de monitoreo para Ethereum que sea reproducible, abierto y sustentable, capaz de detectar y contextualizar eventos críticos a partir de los datos expuestos por la capa de consenso y la capa de ejecución.

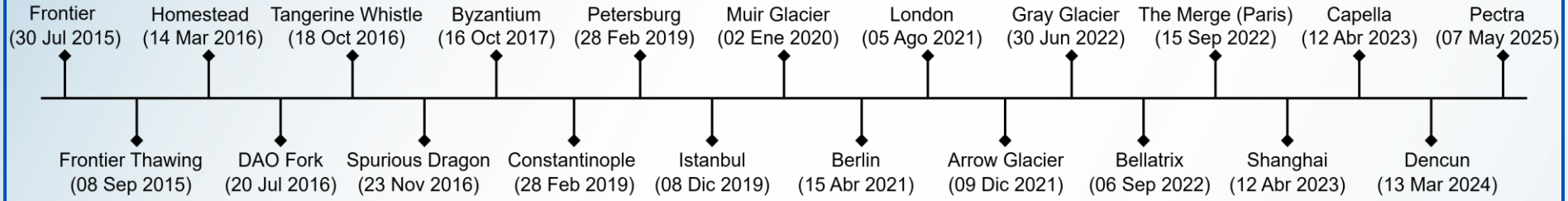
OBJETIVOS ESPECÍFICOS

- Elaborar un marco conceptual actualizado sobre la operación de Ethereum y las métricas/eventos relevantes para el monitoreo.
- Diseñar una arquitectura de referencia para la ingesta, transformación, almacenamiento y visualización de datos.
- Desarrollar y publicar una librería open source para consumir eventos SSE (Server Sent Events) de la capa de consenso de forma robusta y reutilizable.
- Implementar herramientas dentro del sistema de monitoreo que integre esos eventos en flujos ETL.
- Construir tableros y reglas de alerta que permitan seguir en tiempo casi real la evolución de la red y de los nodos.

ESTRUCTURA DE LA PRESENTACIÓN

1. Problema, motivación y objetivos.
2. **Marco teórico.**
3. Sistema de monitoreo propuesto.
4. Herramientas desarrolladas.
5. Resultados, limitaciones y trabajo futuro.
6. Conclusiones y preguntas.

EVOLUCIÓN DE ETHEREUM Y FUENTES CONSULTADAS



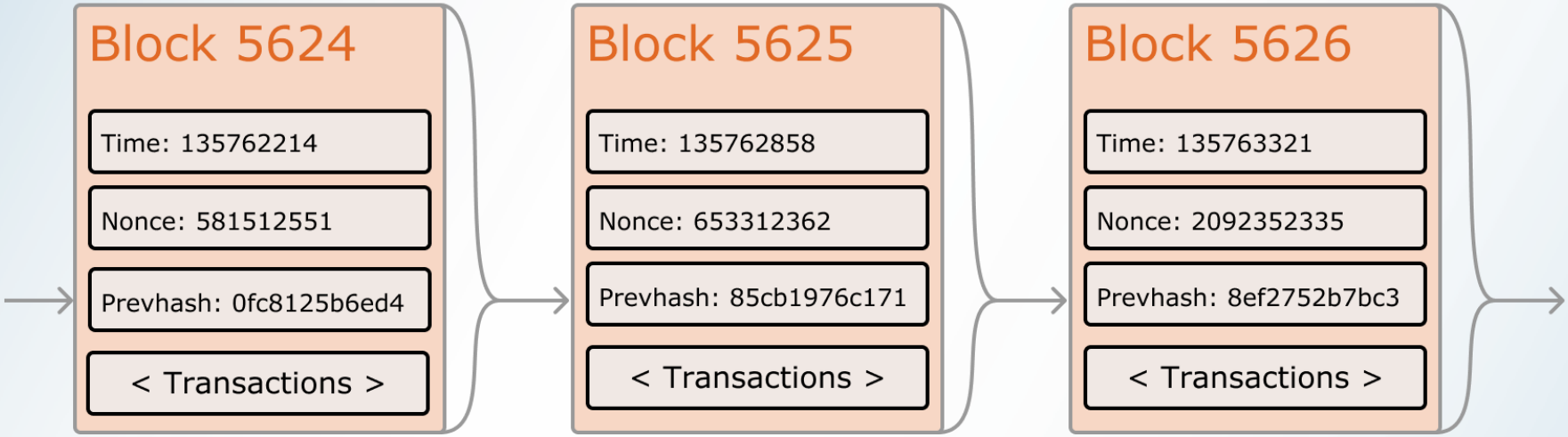
- Ethereum.org – "Proof-of-Stake (PoS)" y documentación oficial de Ethereum. Disponible en: https://ethereum.org/developers/docs/consensus-mechanisms/pos/
- Ethereum Improvement Proposals. Disponible en: https://eips.ethereum.org/
- Ethereum Consensus Specs & Beacon APIs – especificaciones de la Beacon Chain y APIs de nodos. Disponible en: https://github.com/ethereum/consensus-specs y https://github.com/ethereum/beacon-APIs

SOLUCIONES DE MONITOREO EXISTENTES (RESUMEN)

Enfoque	Limitación
Exporters Prometheus de clientes.	Se centran en métricas internas del cliente (CPU, peers, sincronización), no exponen el flujo SSE /eth/v1/events ni hacen ETL de la Beacon API.
Dashboards públicos (etherscan.io, beaconstate.info).	Muestran métricas agregadas de consenso y seguridad, pero el pipeline de ingesta y almacenamiento es **cerrado** y difícil de reproducir/auto-hospedar.
Propuesta de este trabajo.	Pipeline auto-hospedable basado en NiFi + InfluxDB + Grafana, diseñado para ser reproducible y extensible.

BLOCKCHAIN Y ETHEREUM

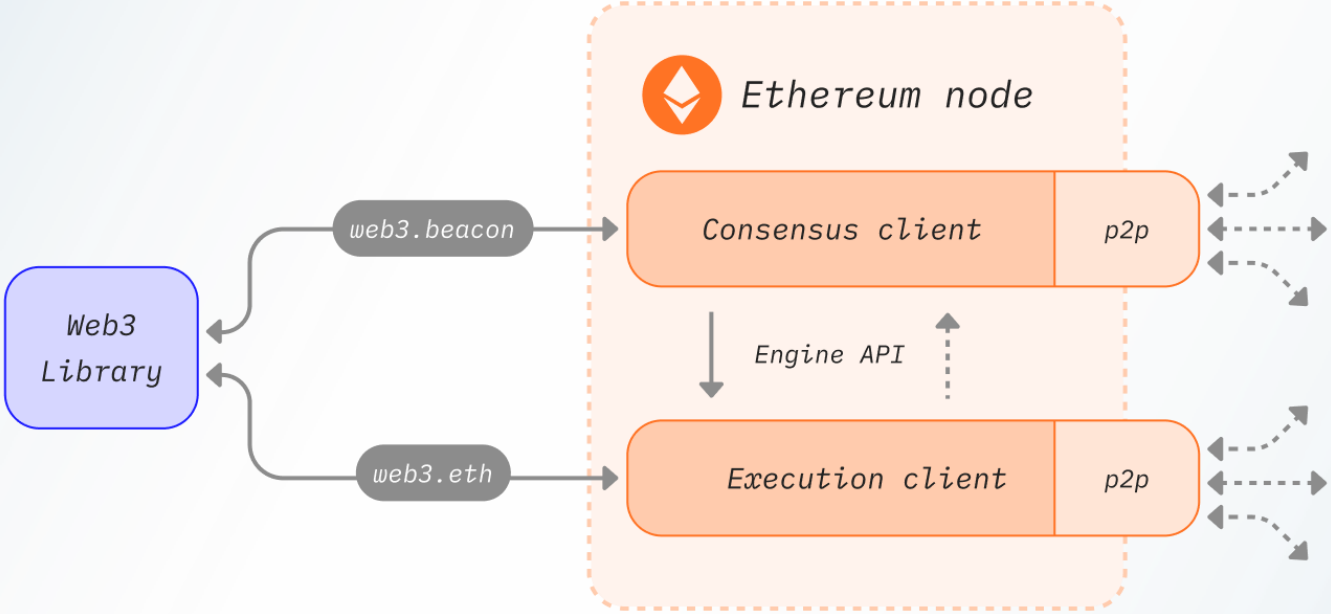
- Libro mayor distribuido y compartido entre muchos nodos.
- Bloques enlazados criptográficamente (hash del bloque previo).
- Cada bloque incluye transacciones y actualiza el estado.
- Ethereum: blockchain programable (smart contracts).



Fuente: Guía de Ethereum. ethereum.org. URL: https://ethereum.org/es/whitepaper/ (visitado 14-11-2025).

ARQUITECTURA POST-MERGE

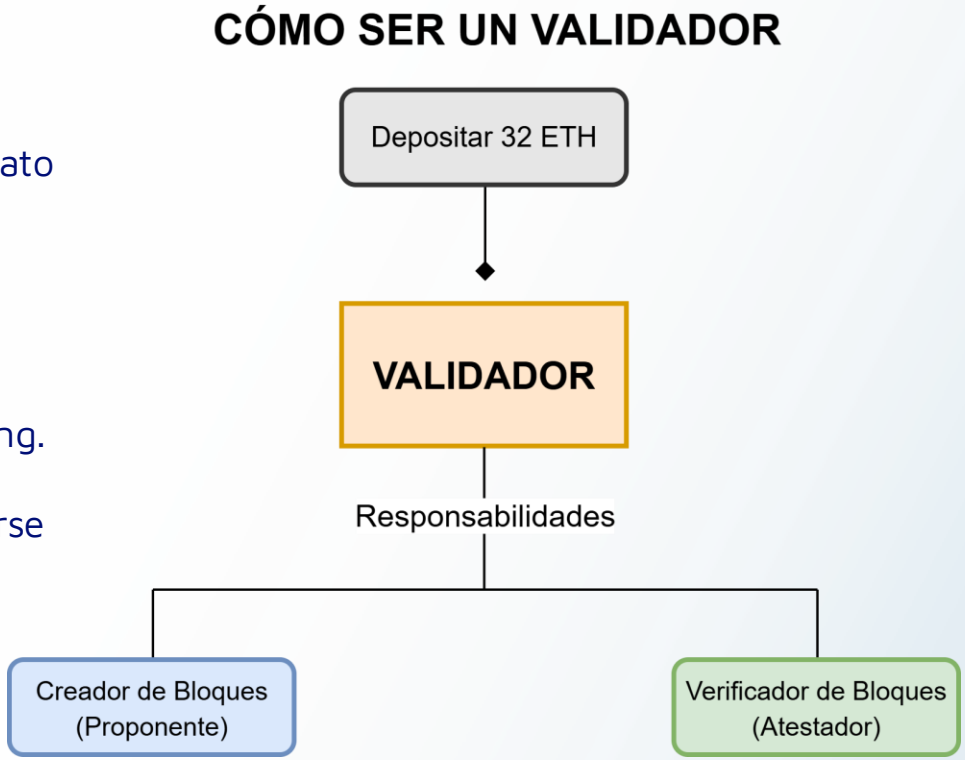
- Separación en capa de consenso (Beacon Chain) y capa de ejecución.
- Clientes de consenso ↔ clientes de ejecución (Engine API).
- La aplicación ve un "nodo Ethereum", pero internamente son dos capas.



Fuente: Guía de Ethereum. ethereum.org. URL: https://ethereum.org/es/whitepaper/ (visitado 14-11-2025).

STAKING, VALIDADORES Y SLASHING

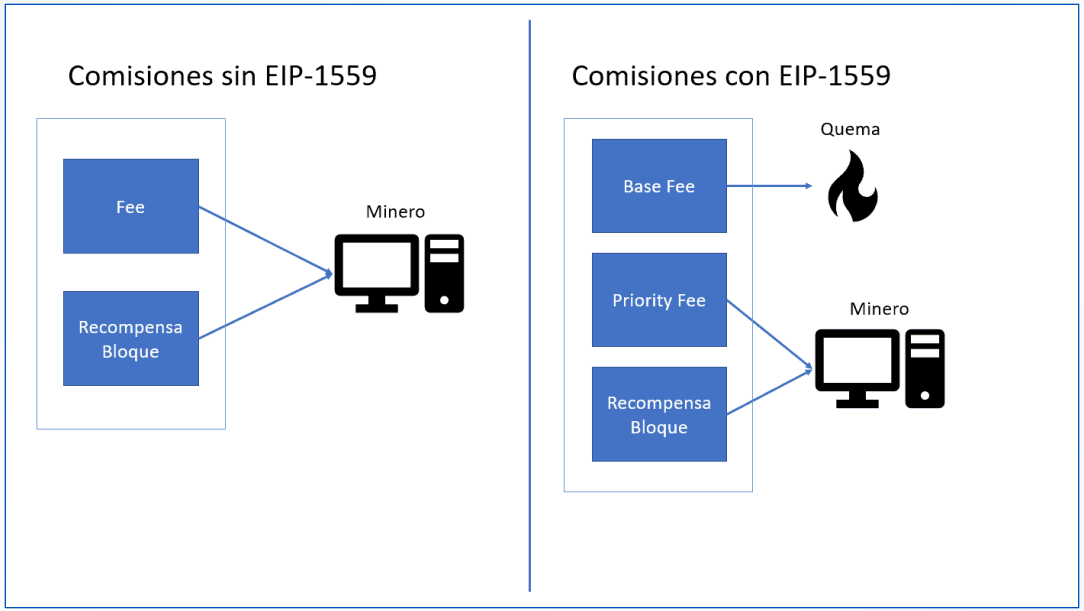
- Ser validador requiere 32 ETH (depósito en contrato de depósito).
- Validadores proponen bloques y atestan sobre la cadena.
- Penalizaciones económicas: inactividad vs slashing.
- Eventos de slashing son críticos y deben detectarse rápido.



GAS, EIP-1559

Gas y EIP-1559:

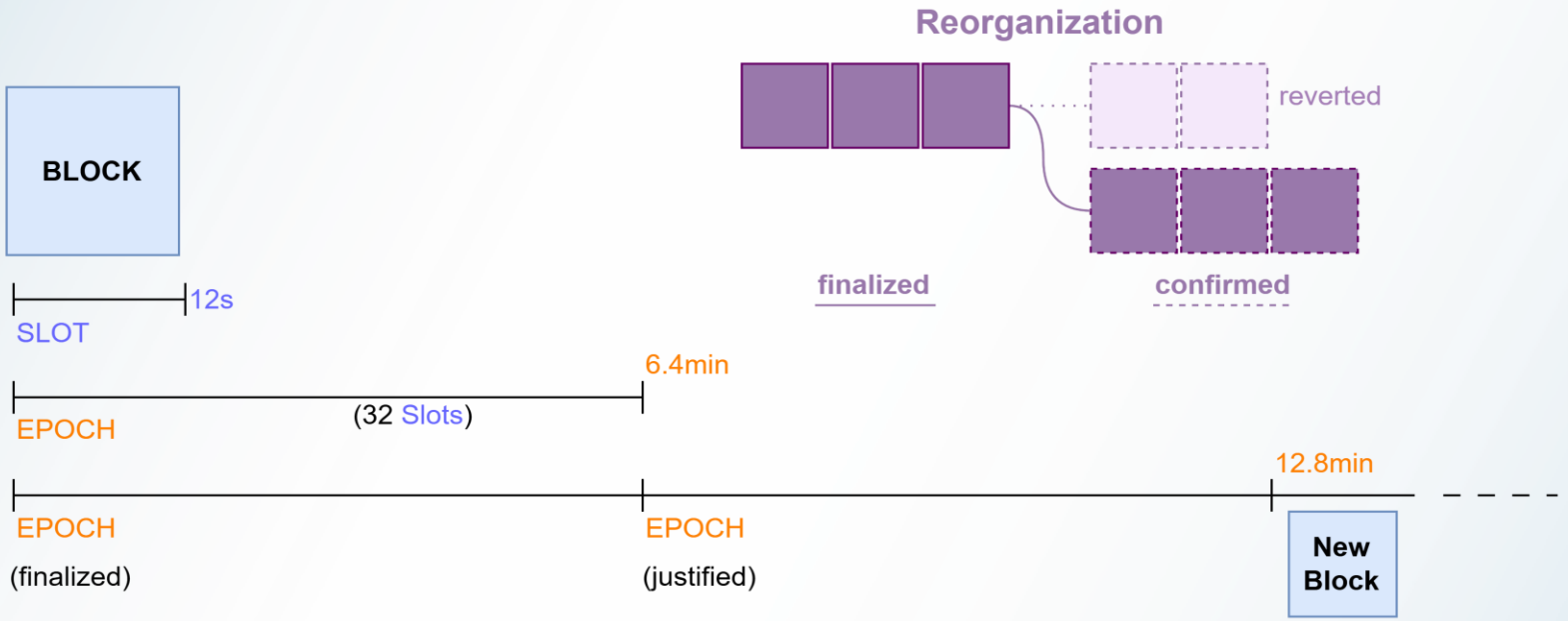
- El gas mide el trabajo que hace la EVM. Se mide en GWEI (1 GWEI = 1×10^{-9} ETH).
- EIP-1559 introduce base fee quemada + priority fee.
- Base fee sube cuando los bloques van llenos y baja cuando hay espacio.



Fuente: ¿Qué es EIP 1559? URL: https://academy.bit2me.com/que-es-eip-1559/ (visitado 18-11-2025).

TIEMPO EN ETHEREUM POS: SLOTS Y EPOCHS

- Slot ≈ 12 segundos (unidad básica de tiempo).
- Epoch = 32 slots (ventana lógica de consenso).
- Finality: cuándo un bloque se considera extremadamente difícil de revertir.
- Pueden ocurrir reorganizaciones ("reorgs") antes de la finality.



RESUMEN: QUÉ QUEREMOS OBSERVAR

Consenso

- Slots.
- Epochs.
- Finality.
- Reorganizaciones.
- Slashing.
- Participación.

Ejecución

- Gas.
- Uso de bloques.

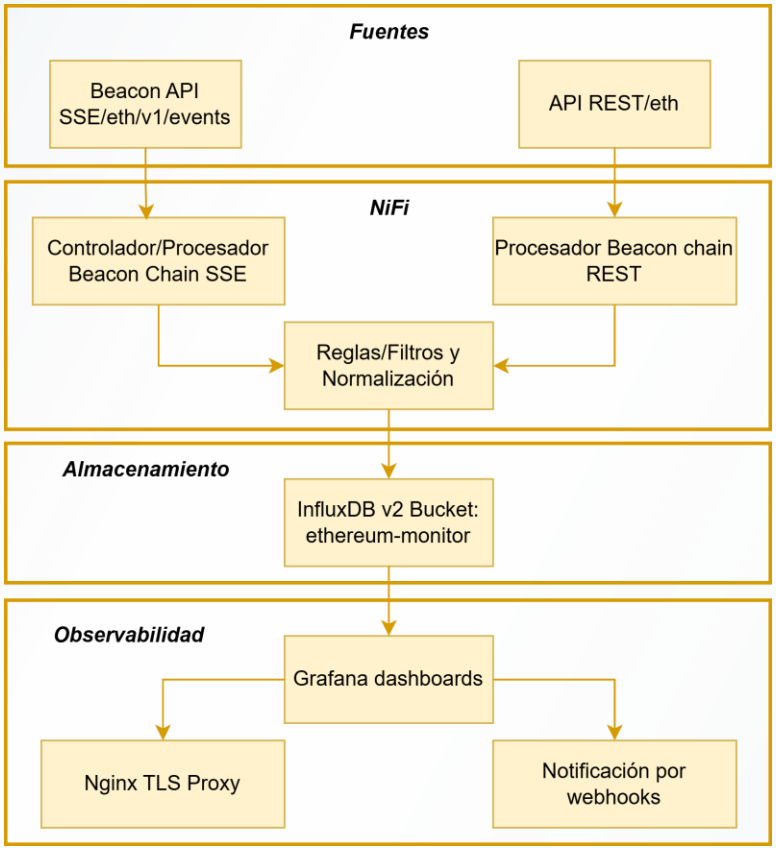
Salud de los nodos

- Health status.
- Estado de sincronizacion.
- Distancia de sincronizacion.
- Conexion entre nodos.

ESTRUCTURA DE LA PRESENTACIÓN

1. Problema, motivación y objetivos.
2. Marco teórico.
3. **Sistema de monitoreo propuesto.**
4. Herramientas desarrolladas.
5. Resultados, limitaciones y trabajo futuro.
6. Conclusiones y preguntas.

VISIÓN GENERAL DEL SISTEMA DE MONITOREO



De eventos de consenso y métricas → a paneles y alertas operables.

FUENTES DE DATOS: BEACON NODE API Y MÉTRICAS

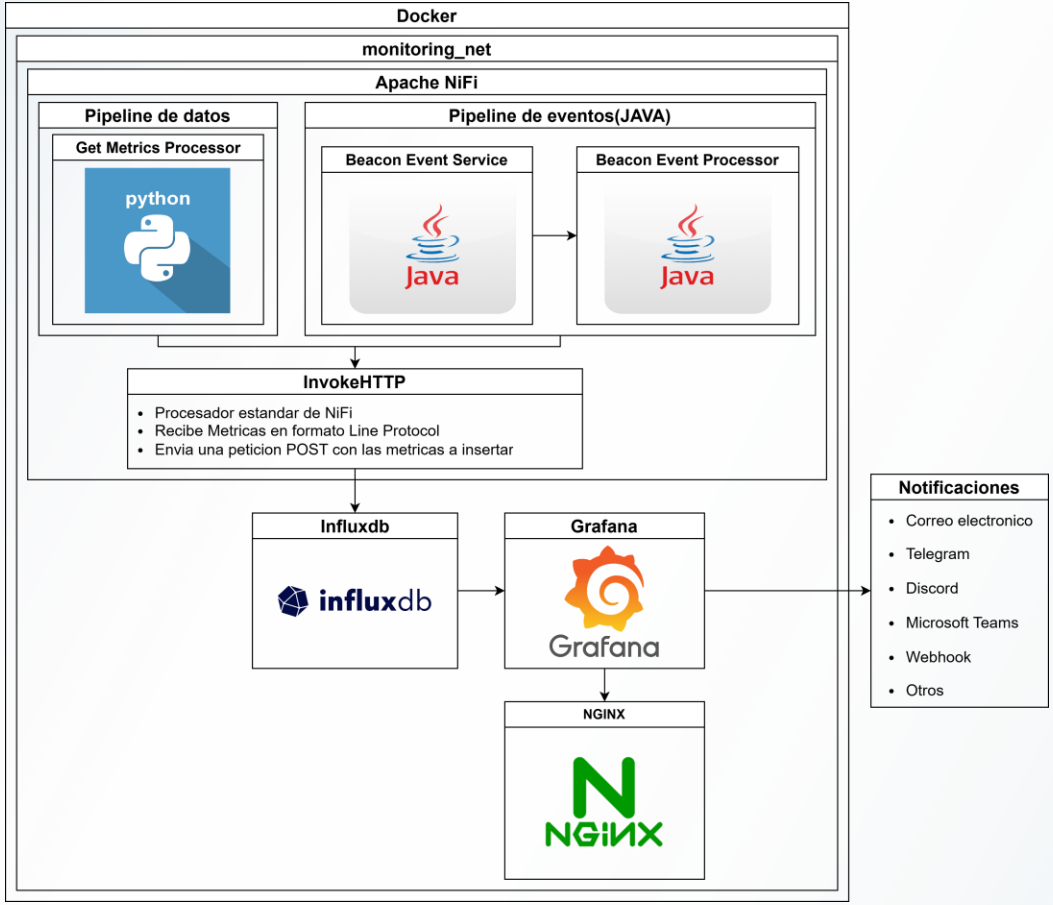
Beacon Node API / SSE:

- Eventos de progreso de cadena (slots, epochs, bloques).
- Eventos de validadores (participación, cambios de estado).
- Eventos de seguridad (alertas, slashing, degradación de finality).
- Otros eventos relevantes según la implementación del cliente.

Métricas complementarias:

- Métricas de ejecución (gas, uso de bloques).
- Métricas de salud del nodo(sincronizacion, health check).

DESPLIEGUE CON DOCKER COMPOSE



ENTORNO EXPERIMENTAL Y CONFIGURACIÓN

Plataforma:
- VPS (gentileza del Nidtec), Ubuntu 24.04 LTS.
- 4 vCPU, 16 GB RAM, SSD 256 GB.

Stack de monitoreo:
- NiFi, InfluxDB, Grafana y Nginx.
- Todo orquestado en Docker.

Fuentes de datos y red:
- Red: Ethereum mainnet.
- Beacon API: endpoints de proveedores externos.
- Suscripción a tópicos SSE relevantes: head, block, finalized_checkpoint, chain_reorg, etc.

Ventana de pruebas:
- Duración: campaña continua de 24 horas.
- Aproximadamente 7.200 slots y 225 epochs.
- Recolección simultánea de métricas internas del stack (NiFi, InfluxDB, Grafana).

ESTRUCTURA DE LA PRESENTACIÓN

1. Problema, motivación y objetivos.
2. Marco teórico.
3. Sistema de monitoreo propuesto.
4. **Herramientas desarrolladas.**
5. Resultados, limitaciones y trabajo futuro.
6. Conclusiones y preguntas.

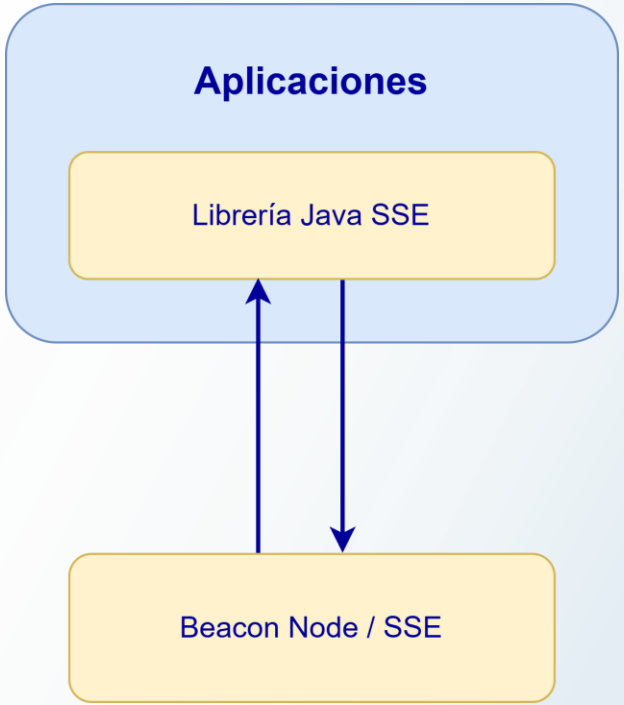
LIBRERÍA JAVA PARA EVENTOS SSE

Librería Web3j:

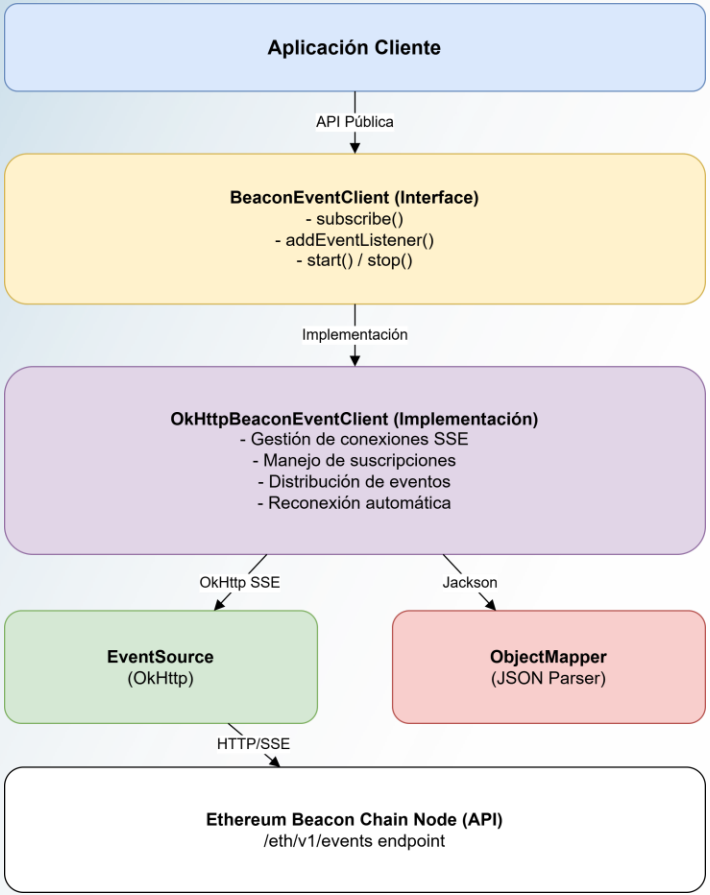
- Suscripción limitada a tópicos SSE de la Beacon Chain: head, block, attestation, voluntary exit, finalized checkpoint, chain reorg.
- Centrado en la capa 2 de ethereum (L2).

Librería eth-events-listener:

- Suscripción robusta a todos los tópicos SSE de la Beacon Chain hasta la fecha.
- Manejo de reconexiones, timeouts y errores de red.
- Modelo de eventos reutilizable en otros proyectos.
- Publicación como artefacto open source.



ARQUITECTURA DE LA LIBRERÍA Y API PÚBLICA



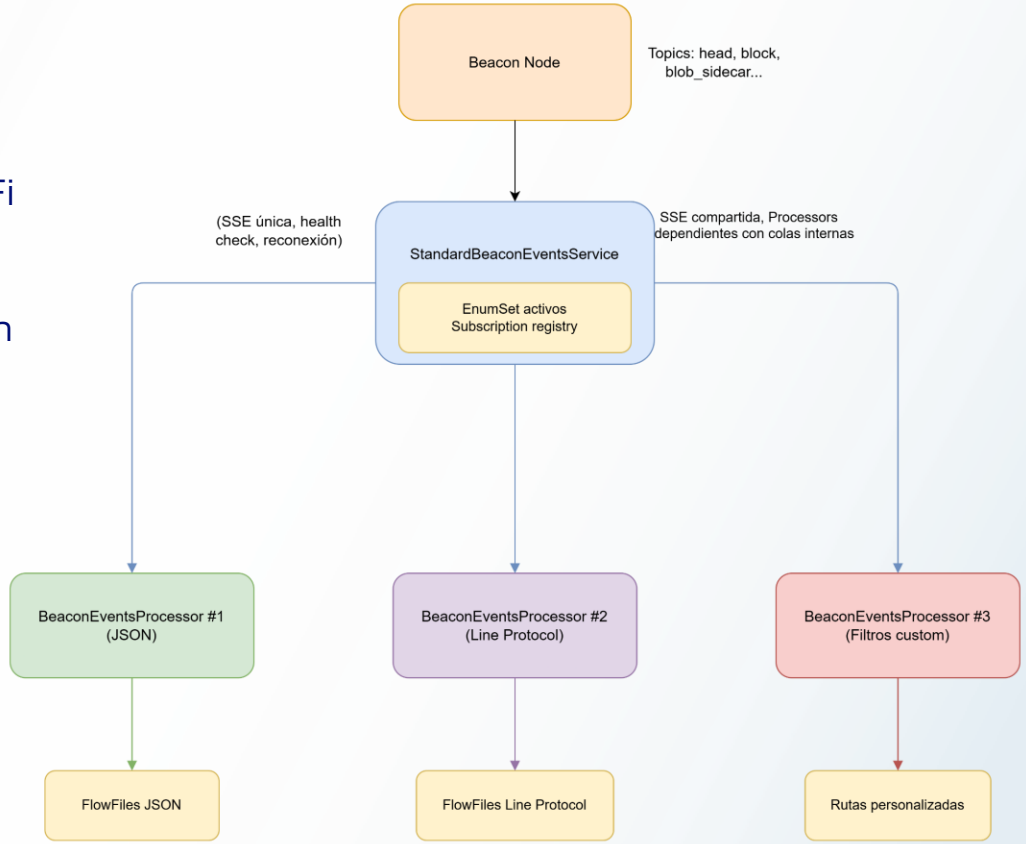

```
BeaconSseClient client = BeaconSseClient.builder()
    .endpoint("https://.../eth/v1/events")
    .topics(List.of("head", "finality", "block"))
    .listener(event -> { /* procesar evento */ })
    .build();

client.start();
```

INTEGRACIÓN CON EL STACK Y ROBUSTEZ

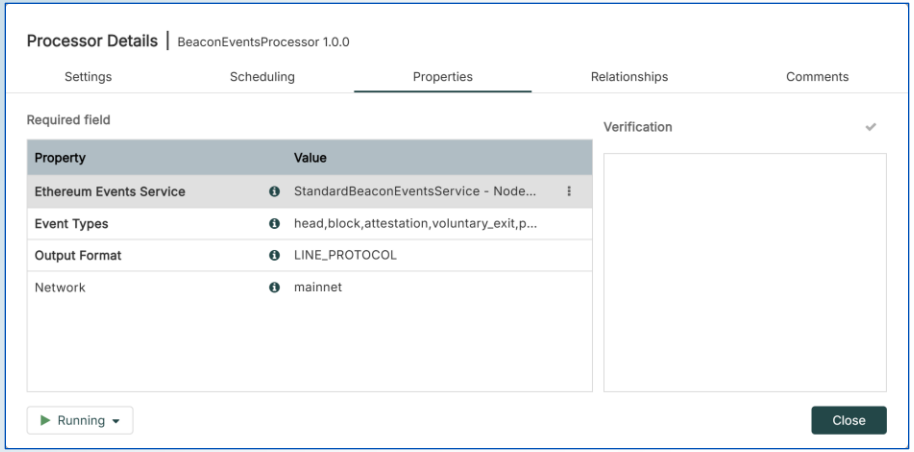
- Integración directa con NiFi a través del bundle desarrollado.
- Capacidad de ser usada por otras aplicaciones Java.
- Manejo de reconexión y errores transitorios.
- Consideraciones de rendimiento y seguridad básica.

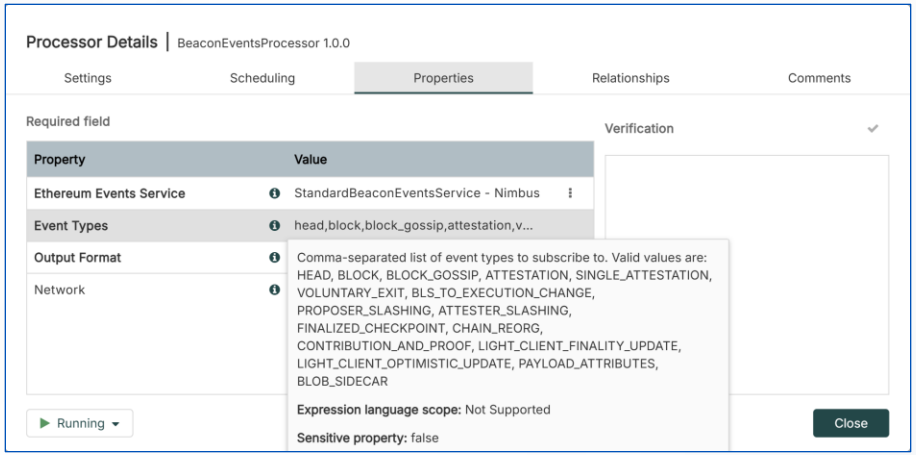
BUNDLE NIFI PARA EVENTOS DE ETHEREUM

- Extensiones personalizadas para Apache NiFi (NAR).
- Controller Service para gestionar la conexión SSE al beacon node.
- Processor BeaconEventsProcessor: suscribe eventos y crea FlowFiles en JSON o Line Protocol.
- Configuración vía interfaz gráfica de NiFi.

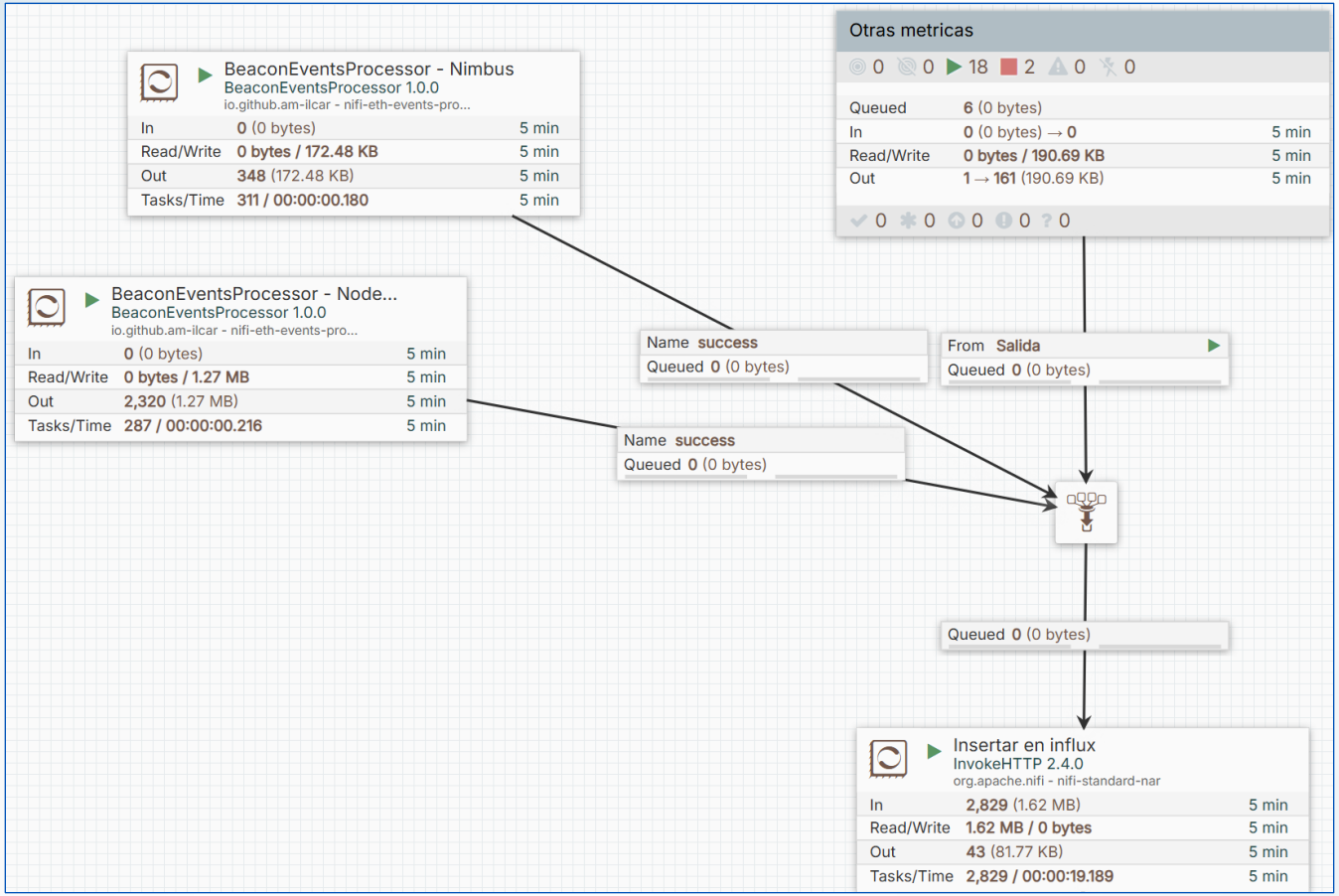


DISEÑO DEL BUNDLE Y CONFIGURACIÓN EN NIFI





DISEÑO DEL BUNDLE Y CONFIGURACIÓN EN NIFI



PRUEBAS Y VALIDACIÓN DE LAS HERRAMIENTAS

Listener de eventos SSE

Pruebas unitarias de parseo:
- JSON de la Beacon API a modelos internos tipados.

Pruebas de callbacks:
- Eventos entregados al listener correcto, sin duplicados.

Ciclo de vida y resiliencia:
- Start/stop/close con cortes de red y timeouts, verificando reconexión sin pérdida de eventos.

Compatibilidad de ejecución:
- Validada en Java 21 sobre Linux.

Bundle NiFi

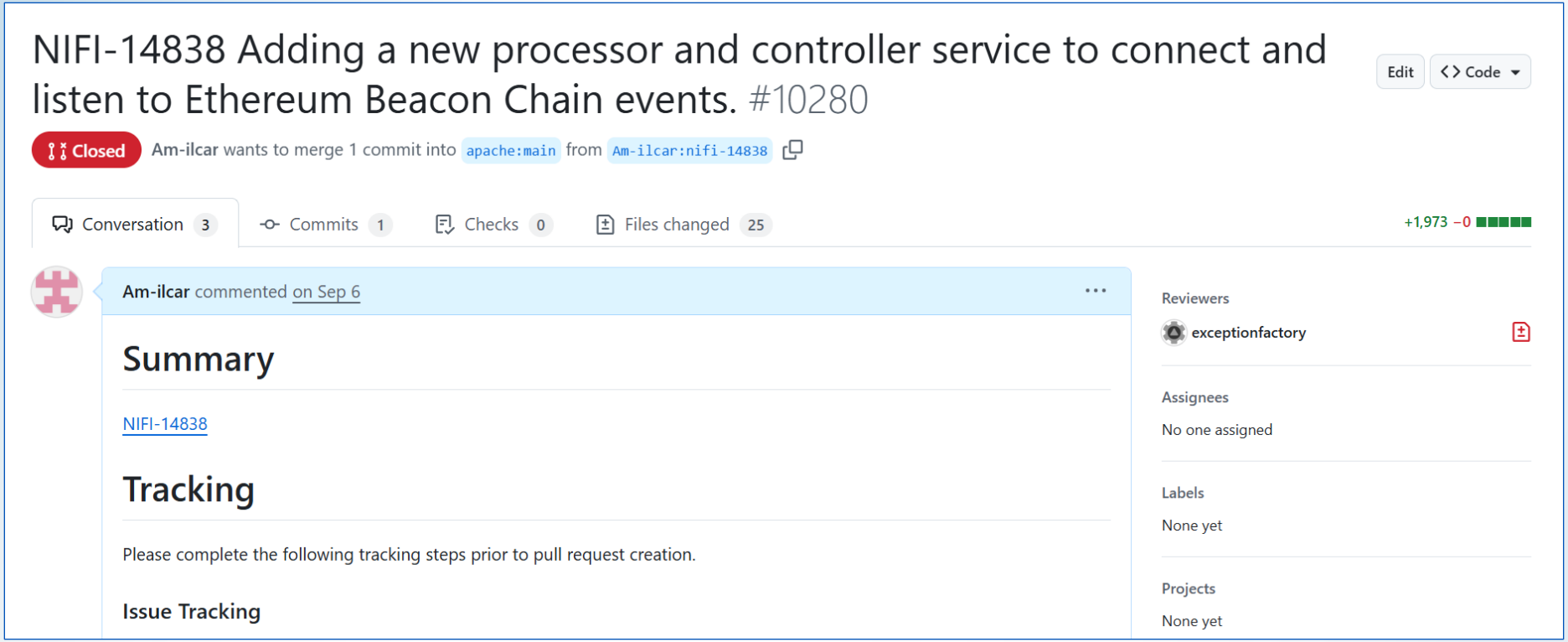
Pruebas automatizadas con nifi-mock + Junit:
- Validación de propiedades obligatorias y rechazo de configuraciones inválidas.

Escenarios de suscripción múltiple:
- Varios processors suscritos al mismo servicio, gestión correcta de cada suscriptor.

Verificación de salida JSON:
- Estructura y atributos del FlowFile según tipo de evento; variante Line Protocol usada solo en el entorno de la tesis.

Ciclo de vida y errores de conexión:
- OnEnabled/onDisabled/onStopped y reintentos ante fallos transitorios sin romper el flujo de NiFi.

APORTE A LA COMUNIDAD OPEN SOURCE



NIFI-14838 Adding a new processor and controller service to connect and listen to Ethereum Beacon Chain events. #10280

Edit Code

Closed Am-ilcar wants to merge 1 commit into apache:main from Am-ilcar:nifi-14838

Conversation 3 Commits 1 Checks 0 Files changed 25 +1,973 −0

Am-ilcar commented on Sep 6

Summary

NIFI-14838

Tracking

Please complete the following tracking steps prior to pull request creation.

Issue Tracking

Reviewers
exceptionfactory

Assignees
No one assigned

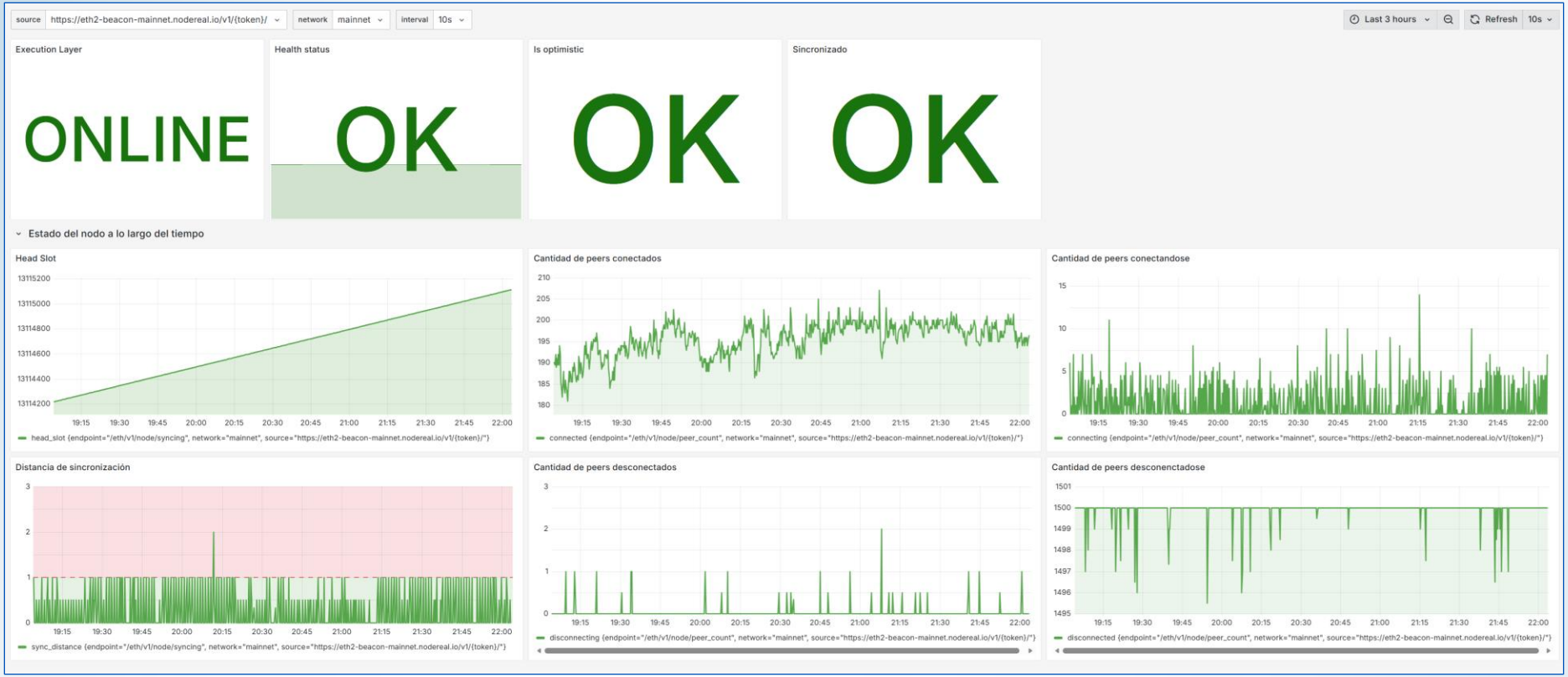
Labels
None yet

Projects
None yet

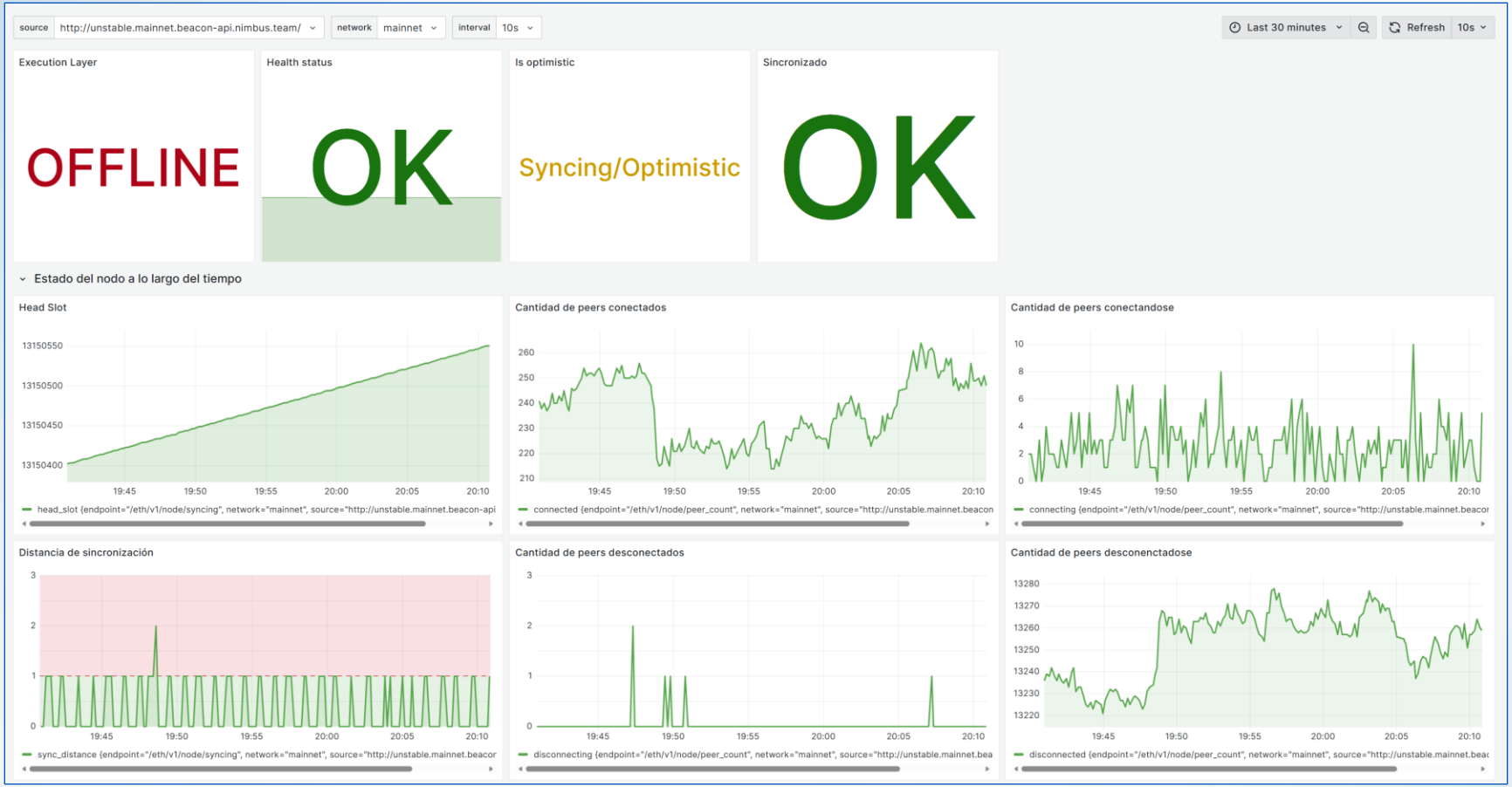
ESTRUCTURA DE LA PRESENTACIÓN

1. Problema, motivación y objetivos.
2. Marco teórico.
3. Sistema de monitoreo propuesto.
4. Herramientas desarrolladas.
5. **Resultados, limitaciones y trabajo futuro.**
6. Conclusiones y preguntas.

PANELES DE GRAFANA Y CRITERIOS DE EVALUACIÓN



PANELES DE GRAFANA Y CRITERIOS DE EVALUACIÓN



PANELES DE GRAFANA Y CRITERIOS DE EVALUACIÓN

PANELES DE GRAFANA Y CRITERIOS DE EVALUACIÓN

PANELES DE GRAFANA Y CRITERIOS DE EVALUACIÓN

22:14 APP Grafana 🔥 **[FIRING] Evento - chain_reorg**

Resumen: Evento chain_reorg detectado

Descripción: Esto ocurre cuando se detectan temporalmente dos versiones de la blockchain (una bifurcación) y el nodo elige unirse a la que se considera la versión canónica (la más fuerte), haciendo que los bloques de la cadena descartada sean inválidos.

Network: `mainnet`

Endpoint: `/eth/v1/events`

Event type: `chain_reorg`

Instancias afectadas:

- `http://unstable.mainnet.beacon-api.nimbus.team/` → *firing*
- `https://eth2-beacon-mainnet.nodereal.io/v1/` → *firing*
- `https://ethereum-mainnet.core.chainstack.com/` → *firing*

[FIRING:3] Evento - chain_reorg Evento detectado (/eth/v1/events chain_reorg mainnet)

Grafana v12.0.0

22:29 APP Grafana 🔥 **[FIRING] Estado de la ingesta de eventos**

Resumen: Estado de la ingesta de eventos

Descripción: Indica si existen inconvenientes o no con respecto a la ingesta de eventos

Instancias afectadas:

- `http://unstable.mainnet.beacon-api.nimbus.team` → *firing*

[FIRING:1] Estado de la ingesta de eventos Estado de la ingesta de eventos (http://unstable.mainnet.beacon-api.nimbus.team/)

Grafana v12.0.0

APP Grafana ✅ **[RECOVERY] Estado de la ingesta de eventos**

Resumen: Estado de la ingesta de eventos

Descripción: Indica si existen inconvenientes o no con respecto a la ingesta de eventos

Instancias afectadas:

- `http://unstable.mainnet.beacon-api.nimbus.team` → *resolved*

[RESOLVED] Estado de la ingesta de eventos Estado de la ingesta de eventos (http://unstable.mainnet.beacon-api.nimbus.team/)

Grafana v12.0.0

RESULTADOS EXPERIMENTALES Y ESCENARIOS DE USO

Escenario	Qué se midió	Resultado
Operación continua 24 h.	Slots/epochs esperados vs registrados; huecos en series head/finalized.	Sin pérdida de eventos; series completas para head/finalized.
Latencia evento → dashboard.	Diferencia entre timestamp del evento de consenso y su aparición en Grafana.	Mediana ~350 ms; p95 < 750 ms (monitoreo < 1 s).
Consumo de recursos del stack.	CPU y RAM combinados (NiFi, InfluxDB, Grafana, Nginx).	CPU < 40 %; ~3,5 GB RAM; sin swap ni saturación de disco.
Cortes de Beacon API / reinicios internos.	Comportamiento ante interrupciones breves y reinicios de InfluxDB/Grafana.	Colas de NiFi absorben retrasos; no se observan huecos ni duplicados tras la recuperación.

LIMITACIONES DEL SISTEMA

- Dependencia de la evolución de la Beacon API (nuevos esquemas y tipos de evento).
- Cobertura acotada de formatos de salida y volumen de eventos.
- Variante de Line Protocol usada sólo en el contexto de la tesis.
- Foco en un conjunto representativo de métricas, no en todas las posibles.

TRABAJOS FUTUROS

- Extender la cobertura de tipos de eventos a medida que evoluciona la Beacon API.
- Extender la observabilidad a nivel de paneles de manera a poder aumentar el detalle.
- Explorar integración con otras bases de series temporales.
- Implementar alertas inteligentes de eventos a través de Machine Learning.

ESTRUCTURA DE LA PRESENTACIÓN

1. Problema, motivación y objetivos.
2. Marco teórico.
3. Sistema de monitoreo propuesto.
4. Herramientas desarrolladas.
5. Resultados, limitaciones y trabajo futuro.
6. **Conclusiones y preguntas.**

CONCLUSIONES GENERALES

- Se abordó el problema de observabilidad en Ethereum post-Merge.
- Se diseñó y materializó un sistema de monitoreo reproducible, abierto y sustentable.
- Se integró un marco conceptual con una arquitectura de datos y herramientas específicas.
- El sistema sostiene la cadencia de la Beacon Chain con latencia < 1 s y consumo moderado de recursos.
- Se cumplieron el objetivo general y los objetivos específicos (marco teórico, arquitectura, librería, bundle y dashboards).

LECCIONES APRENDIDAS

- Un marco teórico bien alineado con la Beacon API facilita el diseño de dashboards útiles.
- Separar claramente consenso, ejecución e infraestructura ayuda a estructurar el monitoreo.
- La modularidad del stack (NiFi, InfluxDB, Grafana) simplifica la replicabilidad y la evolución.
- La observabilidad de Ethereum requiere combinar teoría de protocolo con prácticas de ingeniería de datos.

Muchas Gracias.