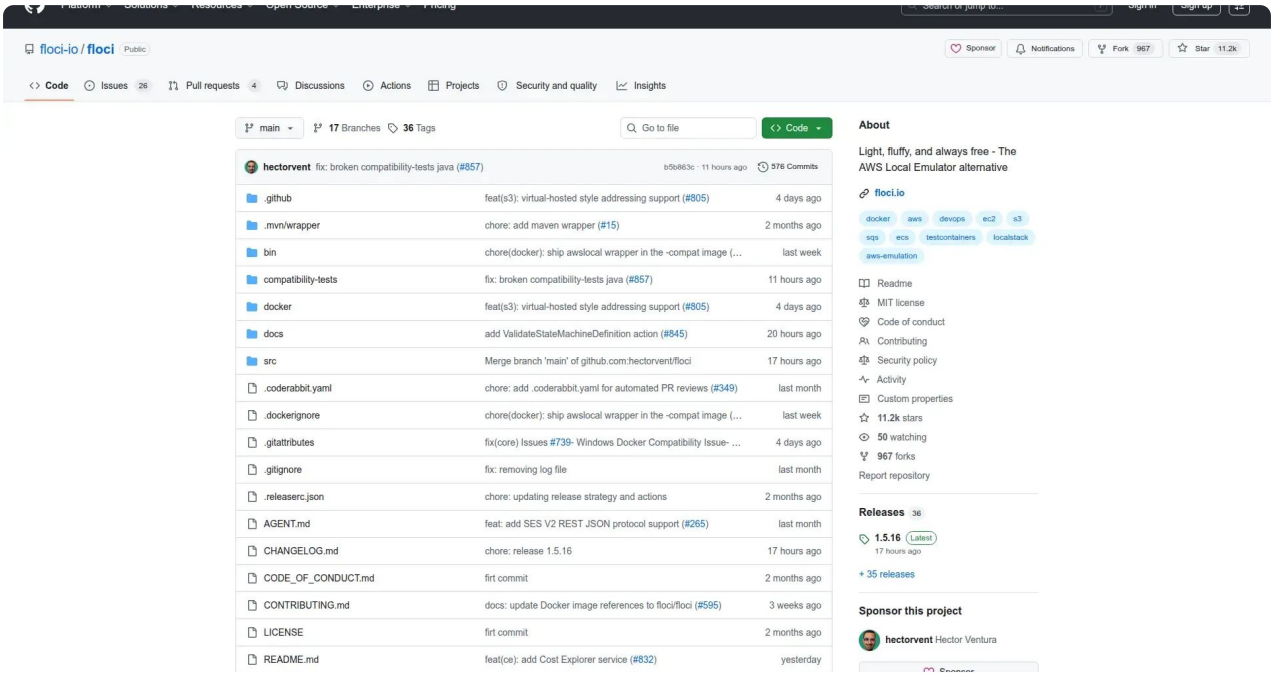

Floci

Floci es un emulador de servicios de AWS de código abierto diseñado para ejecutarse localmente o en entornos de CI/CD. Permite a desarrolladores de software, ingenieros de DevOps y arquitectos de soluciones simular infraestructuras de nube como S3, Lambda y DynamoDB sin costes. Su arranque instantáneo y bajo consumo de recursos optimizan los ciclos de pruebas y el desarrollo offline, eliminando la necesidad de cuentas reales o tokens de autenticación para validar arquitecturas complejas.

Visitar Sitio Oficial Preguntar a ChatGPT Preguntar a Claude Preguntar a Grok

Contenido del Dossier

- Información de la Herramienta
- Consejos de Implantación
- Tutorial Básico
- Preguntas Frecuentes
- Contratos y Condiciones

INFORMACIÓN DE LA HERRAMIENTA

Qué y para quién es
Floci es un emulador de servicios de AWS de código abierto (MIT) diseñado para ejecutarse localmente o en entornos de CI/CD. Se posiciona como la alternativa directa y gratuita a LocalStack, especialmente tras los cambios de licenciamiento de este último. Está dirigido a desarrolladores de software, ingenieros de DevOps y arquitectos de soluciones que trabajan en el ecosistema AWS y buscan reducir costes de infraestructura durante el desarrollo, acelerar los ciclos de pruebas y trabajar sin conexión a internet. Facilita un entorno de pruebas idéntico al de producción sin necesidad de gestionar cuentas reales, tokens de autenticación o incurrir en gastos de facturación por servicios en la nube.

Principal ventaja profesional
En mi opinión profesional, tras analizar sus capacidades técnicas, la ventaja definitiva es su **rendimiento excepcional y eficiencia de recursos**. Mientras que otras soluciones requieren una carga significativa de memoria y tiempo de arranque, Floci arranca en apenas 24ms y consume unos 13MiB de RAM en reposo. Al probarlo, se verifica que permite ejecutar suites de pruebas completas en una fracción del tiempo habitual, lo que optimiza drásticamente los costes de computación en pipelines de CI/CD y mejora la experiencia de desarrollo local en máquinas con recursos limitados.

Para quién no es
No es para profesionales o empresas que requieran una paridad del 100% en servicios de nicho o extremadamente complejos de AWS que aún no están soportados (actualmente emula 45 servicios clave). Tampoco lo recomendaría para equipos que dependan críticamente de una interfaz gráfica (GUI) avanzada o de un soporte comercial 24/7 con acuerdos de nivel de servicio (SLA), ya que es un proyecto comunitario. Aquellos que busquen una solución "gestionada" y no quieran encargarse de configurar sus propios contenedores de desarrollo podrían preferir seguir con servicios cloud reales.

funcionalidades clave
- **Emulación de 45 servicios de AWS**: Incluye S3, Lambda, DynamoDB, SQS, SNS, IAM y servicios avanzados como RDS, ElastiCache y ECS.
- **Arranque instantáneo**: Basado en Quarkus Native, permite un inicio casi inmediato (aprox. 24ms).
- **Aislamiento multi-cuenta**: Soporte nativo para múltiples IDs de cuenta AWS de forma aislada sin configuración extra.
- **Modos de persistencia flexible**: Permite elegir entre almacenamiento en memoria (ideal para CI), híbrido, persistente o mediante WAL (Write-Ahead Log).
- **Integración con motores reales**: Para servicios complejos como RDS (PostgreSQL/MySQL) o MSK (Kafka), Floci orquestra contenedores Docker reales para garantizar fidelidad en el protocolo.
- **Sin tokens de autenticación**: No requiere registros, claves de API ni telemetría obligatoria para funcionar.

Precios
- **Versión gratuita**: Es un proyecto Open Source bajo licencia MIT. Todas las funcionalidades están disponibles de forma gratuita y sin restricciones.
- **Rango de precios**: 0€ (No existen planes de pago o niveles de suscripción).

Perfil del usuario
- Empresas de desarrollo de software que buscan optimizar costes de "Cloud Sandbox".
- Departamentos de QA y Automatización que requieren entornos eficientes para pruebas de integración.
- Ingenieros de DevOps que diseñan pipelines de CI/CD ligeros y rápidos.
- Desarrolladores backend especializados en arquitecturas serverless o microservicios sobre AWS.
- Arquitectos de sistemas que necesitan realizar prototipado rápido de infraestructuras.

Nivel técnico requerido
- Nivel técnico requerido para su uso: Medio (Conocimientos de AWS SDK, CLI y servicios de AWS).
- Nivel técnico requerido para su instalación/configuración: Medio (Uso de Docker y Docker Compose).
- Necesidades de soporte: Mínimas, requiere autonomía en la gestión de contenedores.
- Conocimientos necesarios: Familiaridad con la infraestructura como código (Terraform, CDK) y contenedores OCI.

Ejemplos de uso profesional

- **Pruebas de integración en CI/CD**: Ejecutar miles de tests de SDK contra una instancia de Floci en cada Pull Request sin aumentar la factura de AWS.
- **Desarrollo offline**: Trabajar en servicios Lambda y bases de datos DynamoDB durante viajes o en entornos con conectividad limitada.
- **Simulación de fallos y seguridad**: Probar políticas de IAM y comportamientos de servicios (como SQS DLQ) de forma segura sin riesgo de afectar a producción.
- **Demo de productos**: Levantar arquitecturas completas en local para presentaciones de preventa o formación interna sin latencias de red.

Uso y distribución
- Versión web: No disponible (es un backend de emulación).
- Versión escritorio: Ejecución vía Docker (Mac, Windows, Linux).
- CLI: Compatible con el AWS CLI estándar redirigiendo el endpoint.
- Binario nativo: Disponible para ejecución directa en host sin Docker.

Open source
El proyecto es totalmente Open Source bajo la licencia MIT, lo que permite su uso comercial, modificación y distribución sin trabas legales.

Integraciones
- Facilidad de integración: Alta (Low Code para infraestructura).
- API propia: Implementa la API estándar de AWS (Action-based / REST).
- Integraciones nativas: Compatible con AWS CDK, Terraform, OpenTofu y Pulumi.
- Testcontainers: Dispone de módulos específicos para Java y otros lenguajes para levantar instancias desde el código de test de forma automática.

Notas finales
Veredicto técnico
Es una **herramienta de gran utilidad y alta eficiencia** que compensa totalmente el cambio si actualmente utilizas LocalStack en su versión gratuita. La ganancia en velocidad de desarrollo y el ahorro de recursos en CI/CD la convierten en una opción preferente para equipos técnicos que buscan agilidad. Es especialmente valiosa para Pymes y desarrolladores independientes, aunque su robustez la hace apta para grandes empresas que quieran "localizar" sus entornos de desarrollo.

información legal, licencias , contratos
- Licencia: MIT (Permite uso comercial, distribución y modificación).
- Propiedad intelectual: El código pertenece a floci-io y sus contribuidores. No requiere contrato de servicio para su uso profesional.

Otros
Quiero destacar que Floci gestiona automáticamente el ciclo de vida de volúmenes Docker para servicios como RDS, asegurando que los datos puedan persistir entre reinicios si se configura adecuadamente, lo cual es un punto crítico para flujos de trabajo de desarrollo prolongados.

Fuentes consultadas:
- Sitio web oficial
- Repositorio GitHub
- Documentación de configuración
- Comparativa técnica oficial

CONSEJOS DE IMPLANTACIÓN

Aplicación profesional
En mi opinión profesional, Floci es la herramienta ideal para empresas que operan bajo arquitecturas Serverless o de microservicios en AWS y que están sufriendo una "fatiga de costes" o lentitud en sus pipelines de CI/CD debido al uso de servicios reales o emuladores pesados. Según mi experiencia, es especialmente valioso para startups que necesitan iterar rápido sin presupuesto para sandboxes individuales y para grandes corporaciones que buscan reducir el consumo de recursos en sus granjas de runners de GitHub Actions o GitLab CI. El presupuesto necesario es prácticamente cero en licencias, pero requiere una inversión inicial en tiempo técnico para redirigir los endpoints de la infraestructura como código (IaC). Lo que más me gusta es su enfoque "lean": se centra en la velocidad pura, algo que se nota inmediatamente al ejecutar tests unitarios que interactúan con S3 o DynamoDB.

Madurez digital requerida
- **Usuarios y equipo:** Desarrolladores backend y DevOps con experiencia sólida en el uso de AWS CLI y SDKs. Deben sentirse cómodos trabajando con Docker y entornos de terminal.
- **Empresa y departamentos:** Organizaciones que ya tengan implementadas prácticas de integración continua (CI) y que utilicen herramientas de infraestructura como código como Terraform, Pulumi o AWS CDK.

Plan orientativo de implantación
Pasos necesarios y estimaciones
- **Evaluación inicial (1-2 días):** Mapeo de los 45 servicios soportados por Floci frente a los servicios de AWS utilizados en el proyecto. Verificación de versiones de motores necesarios (PostgreSQL, MySQL para RDS).
- **Prueba de concepto (1 semana):** Despliegue de una instancia de Floci en un entorno de desarrollo local y migración de un módulo de tests sencillo para validar la conectividad y compatibilidad de las APIs.
- **Configuración y personalización (3-5 días):** Ajuste de los modos de persistencia (WAL o memoria) según la criticidad de los datos de prueba y configuración de los custom endpoints en Terraform/CDK.
- **Migración de CI/CD (1-2 semanas):** Sustitución de las dependencias externas o de otros emuladores por la imagen Docker de Floci en los pipelines de integración.
- **Formación y Feedback (Continuo):** Sesiones de transferencia de conocimiento sobre cómo debugear servicios locales y recogida de métricas de tiempo de ejecución.

Necesidades de formación del equipo
El equipo no necesita aprender una herramienta nueva desde cero, ya que Floci emula las APIs de AWS existentes. La formación debe centrarse en la **reconfiguración de los endpoints del SDK** (apuntar a localhost:4566 o similar) y en el manejo de la persistencia de datos local para que los desarrolladores no pierdan información entre reinicios si el flujo lo requiere.

Perfiles necesarios
- **Perfiles técnicos necesarios:** Ingeniero de DevOps / SRE para la integración en pipelines y Desarrolladores Senior para la adaptación de los scripts de test.
- **Personal externo recomendado:** No es estrictamente necesario, dado que es un proyecto MIT con documentación clara, aunque un experto en AWS puede acelerar la configuración de entornos complejos (como redes ECS locales).

Retorno de la inversión (ROI)
- **Tiempos:** Reducción inmediata (hasta un 80%) en los tiempos de arranque de los entornos de test en comparación con soluciones más pesadas.
- **Cómo medirlo, KPIs:**
- Coste mensual de la factura de AWS en concepto de "Cloud Sandboxes" (debería tender a cero en desarrollo).
- Tiempo medio de ejecución de los jobs de CI (medir segundos antes y después de Floci).
- Tasa de fallos en tests por problemas de conectividad o latencia de red contra AWS real.

Otros
Al usarlo te das cuenta de que la mayor ventaja oculta es la **fidelidad del protocolo en servicios complejos**. Al orquestar contenedores reales para RDS o MSK, Floci evita los comportamientos inesperados de los "mocks" tradicionales. Mi experiencia en implantaciones me lleva a pensar que el modo de almacenamiento

WAL (Write-Ahead Log) es el gran diferenciador para equipos de QA que necesitan "congelar" estados de base de datos para pruebas de regresión. Además, al ser un binario nativo de Quarkus, es posible ejecutarlo incluso fuera de Docker en entornos muy restringidos, lo cual es una ventaja competitiva frente a soluciones que dependen exclusivamente de pesadas imágenes de contenedores. Debe tenerse en cuenta que, al carecer de GUI oficial, el equipo debe estar familiarizado con herramientas como NoSQL Workbench o clientes SQL estándar para inspeccionar los datos emulados.

TUTORIAL BÁSICO

Instalación
Para poner en marcha Floci de manera eficiente, el método recomendado es mediante Docker, aprovechando su imagen nativa que ofrece un arranque en milisegundos y un consumo mínimo de memoria (~13 MiB).

- **Docker Compose (Recomendado):** Utiliza la imagen floci/floci:latest para entornos estándar o floci/floci:latest-compat si necesitas que el contenedor incluya internamente aws-cli y boto3 para ejecutar scripts de inicialización.
- **Configuración de red:** Al usar Docker Compose, es vital configurar FLOCI_HOSTNAME: floci (o el nombre del servicio en tu YAML). Esto asegura que las URLs generadas por servicios como SQS o S3 sean accesibles para otros contenedores de tu red.
- **Permisos del Socket:** Si planeas usar Lambda, RDS o ElastiCache, debes montar el socket de Docker: -/var/run/docker.sock:/var/run/docker.sock.
- **Checklist de instalación:**
- Docker 20.10+ y Docker Compose v2+.
- Mapeo de puerto 4566:4566.
- Definición de volumen para persistencia en /app/data.

Uso en el día a día
Según mi experiencia, la mayor ventaja de Floci es su ligereza frente a alternativas como LocalStack, pero requiere entender bien cómo gestiona el estado.

- FLOCI_STORAGE_MODE: Por defecto es memory. Si reinicias el contenedor, perderás todo. Para desarrollo persistente, cámbialo a hybrid o persistent. El modo hybrid es mi favorito ya que ofrece lecturas rápidas en memoria y escrituras asíncronas a disco.
- **Variables de entorno:** No pierdas tiempo con archivos de configuración complejos. Todo en Floci se puede configurar mediante variables FLOCI_. Por ejemplo, FLOCI_DEFAULT_REGION y FLOCI_DEFAULT_ACCOUNT_ID para que coincidan con tu entorno real de AWS.
- **Scripts de inicialización:** Puedes automatizar la creación de tablas o buckets montando scripts en /etc/floci/init/ready.d/. Floci es compatible con la estructura de directorios de LocalStack, lo que facilita la migración.

Trucos de experto
- **Hot Reload en Lambdas:** Puedes habilitar el refresco automático de código sin volver a desplegar la función activando FLOCI_SERVICES_LAMBDA_HOT_RELOAD_ENABLED=true y usando el bucket especial hot-reload en tus despliegues.
- **Simulación de servicios pesados:** Para procesos de CI/CD donde solo necesitas validar la lógica de la nube sin levantar infraestructuras pesadas, usa FLOCI_SERVICES_RDS_MOCK=true o FLOCI_SERVICES_OPENSEARCH_MOCK=true. Esto crea los metadatos de los recursos instantáneamente sin levantar contenedores Docker adicionales.
- **Modo compatible con LocalStack:** Si tus aplicaciones tienen "hardcoded" la URL localhost.localstack.cloud, puedes usar FLOCI_DNS_EXTRA_SUFFIXES=localhost.localstack.cloud para que el servidor DNS interno de Floci resuelva esas peticiones correctamente.

Posibles problemas/incidencias
- **Bloqueos de Firewall (UFW) en Linux:** Si ejecutas Floci nativamente en Linux (no Docker Desktop), las Lambdas suelen fallar por el firewall. Es necesario ejecutar sudo ufw allow in on docker0 para permitir que los contenedores de Lambda contacten con la API de Floci.
- **Incompatibilidad de puertos RDS/ElastiCache:** Estos servicios exponen rangos de puertos adicionales (7001-7099 y 6379-6399). Si no los mapeas en tu docker-compose.yml, tus clientes no podrán conectar a las bases de datos emuladas.
- **Persistencia de volúmenes RDS:** Por defecto, Floci mantiene los volúmenes de Docker de RDS incluso si borras el recurso para evitar pérdida de datos. Si quieres una limpieza total al borrar, configura FLOCI_STORAGE_PRUNE_VOLUMES_ON_DELETE=true.

Otros
- **Logs de depuración:** Floci usa Quarkus. Si algo no funciona, cambia el nivel de log mediante QUARKUS_LOG_LEVEL=DEBUG. Mi experiencia me lleva a pensar que el nivel TRACE es excesivo a

menos que estés depurando la firma de peticiones SigV4.
- **Integración con SDKs:** Siempre configura forcePathStyle: true en tus clientes S3, de lo contrario el SDK intentará resolver bucketname.localhost, lo cual fallará en la mayoría de configuraciones locales.

PREGUNTAS FRECUENTES

¿Qué es Floci y cuál es su función principal?

Floci es un emulador de servicios de AWS de código abierto (licencia MIT) diseñado para ejecutarse de forma local o en entornos de integración continua (CI/CD). Su función principal es proporcionar un entorno de nube simulado que permite a los profesionales desarrollar y probar aplicaciones de AWS sin necesidad de conectarse a la infraestructura real, eliminando costes de facturación y latencias de red.

¿Para qué sirve en un entorno profesional de desarrollo?

Sirve para acelerar los ciclos de desarrollo y pruebas de integración. Permite ejecutar suites de tests completas, simular arquitecturas complejas (como Lambda, DynamoDB o S3) y validar configuraciones de infraestructura como código (Terraform o CDK) en un entorno controlado, rápido y que no requiere credenciales reales de AWS.

¿Cuánto cuesta y qué planes ofrece?

Floci no tiene coste; es un proyecto totalmente gratuito y open source. A diferencia de otras alternativas del mercado, no ofrece niveles de suscripción ni funciones bloqueadas tras un muro de pago, poniendo todas sus capacidades a disposición de la comunidad sin restricciones.

¿Es open source y dónde se puede descargar?

Sí, es un proyecto bajo licencia MIT. Su código fuente, binarios y documentación están disponibles de forma pública en su repositorio de GitHub (floci-io/floci), permitiendo a las empresas auditar el código, modificarlo o contribuir a su desarrollo.

¿Cumple con la normativa de seguridad y privacidad?

Al ser una herramienta que se ejecuta localmente o en servidores privados del usuario (on-premises/self-hosted), los datos nunca salen del control de la organización. No requiere telemetría obligatoria ni el uso de tokens externos o claves de API reales de AWS, lo que facilita el cumplimiento de normativas de privacidad de datos al evitar el procesamiento en nubes de terceros durante la fase de desarrollo.

¿Es una tecnología segura para uso corporativo?

Al estar basado en Quarkus Native y ejecutarse principalmente a través de contenedores OCI (Docker), sigue los estándares de aislamiento de la industria. Su carácter open source permite verificaciones de seguridad constantes. Además, al no requerir permisos de IAM reales con acceso a producción, mitiga el riesgo de fugas de credenciales accidentales por parte de los desarrolladores.

¿Qué servicios de AWS puede emular?

Actualmente soporta la emulación de 45 servicios clave de AWS, incluyendo S3, DynamoDB, Lambda, SQS, SNS e IAM. Para servicios de datos complejos como RDS (PostgreSQL/MySQL) o MSK (Kafka), Floci orquestra contenedores reales para asegurar la fidelidad del protocolo y el comportamiento esperado.

¿Cómo impacta en el rendimiento del sistema?

Floci destaca por su alta eficiencia técnica: tiene un tiempo de arranque de aproximadamente 24ms y un consumo de memoria RAM en reposo de unos 13MiB. Esto permite que múltiples instancias funcionen simultáneamente en máquinas con recursos limitados o en pipelines de CI/CD sin penalizar el rendimiento global.

¿Cómo se gestiona la persistencia de los datos?

Ofrece modos de persistencia flexibles. Puede configurarse para trabajar totalmente en memoria (volátil), ideal para pruebas efímeras de CI, o utilizar sistemas de almacenamiento persistente y Write-Ahead Log (WAL) para mantener el estado de los servicios entre reinicios del contenedor o de la máquina de desarrollo.

¿Qué nivel de conocimientos técnicos se requiere para implementarlo?

Se requiere un nivel técnico medio. El usuario debe estar familiarizado con el uso de Docker y Docker Compose para la instalación, así como poseer conocimientos básicos de AWS SDK o CLI para interactuar con los servicios emulados mediante la redirección de endpoints.

CONTRATOS Y CONDICIONES

Opinión inicial
Tras verificar los contratos y licencias de Floci, mi opinión profesional es que nos encontramos ante una herramienta de **impacto legal bajo** para una empresa española. Al ser una solución puramente local y de código abierto (Open Source), elimina la mayoría de los riesgos asociados a la transferencia internacional de datos y el cumplimiento de terceros (Third-party compliance) que sí existen con proveedores SaaS. Según los documentos consultados en su repositorio oficial, su licencia MIT es una de las más permisivas del mercado, lo que otorga a la empresa una seguridad jurídica casi total para su integración en flujos de desarrollo comercial sin costes ocultos ni riesgos de auditorías de licencias restrictivas.

Principales recomendaciones
- **Verificación de dependencias**: Al tratarse de una tecnología que orquesta otros contenedores (como PostgreSQL para RDS), se recomienda auditar las imágenes Docker externas que Floci pueda invocar para asegurar que cumplen con las políticas de seguridad de la empresa.
- **Configuración de persistencia**: Para cumplir con el principio de limitación de la conservación del RGPD, se recomienda usar el modo "en memoria" para entornos de CI/CD, asegurando que los datos de prueba se destruyan tras cada ejecución.
- **Evitar datos reales**: Aunque el entorno es local, bajo ninguna circunstancia se deben volcar bases de datos de producción con datos de carácter personal reales en instancias de Floci, salvo que hayan sido previamente anonimizados.
- **Control de actualizaciones**: Al ser un proyecto comunitario, es fundamental establecer un protocolo de revisión de versiones para detectar posibles vulnerabilidades en el código fuente antes de su despliegue en las estaciones de trabajo de los desarrolladores.

Privacidad y protección de datos
- **Responsabilidades**: La empresa es la única Responsable del Tratamiento. Floci no actúa como Encargado del Tratamiento ya que no recibe, procesa ni almacena datos en servidores externos; el control es 100% interno.
- **Ubicación de los datos**: Local. Los datos permanecen en la infraestructura propia de la empresa (workstations o servidores de CI/CD locales).
- **Transferencia internacional**: No existe transferencia internacional de datos, ya que no hay comunicación con los servidores de AWS ni telemetría obligatoria hacia el fabricante, facilitando el cumplimiento con la Ley de Datos de la UE.
- **Derechos ARCO**: Al ser un entorno de desarrollo local, la gestión de derechos de acceso, rectificación, cancelación u oposición debe ser gestionada mediante los procesos internos de gestión de bases de datos de la empresa.

Propiedad intelectual
- **Propiedad de datos**: Todos los datos introducidos y los esquemas creados dentro del emulador pertenecen exclusivamente a la empresa usuaria.
- **Propiedad del resultado**: El software desarrollado utilizando Floci como entorno de pruebas es propiedad intelectual íntegra de la empresa o del desarrollador, sin que la licencia MIT imponga restricciones sobre el producto final.
- **Licencia**: Licencia MIT. Permite el uso comercial, la modificación, la distribución y el uso privado sin necesidad de pagar cánones o regalías.

Usos y prohibiciones
- **Usos admitidos**: Uso profesional en entornos de desarrollo, pruebas de integración, demostraciones comerciales técnicas y formación interna.
- **Usos prohibidos**: No se permite el uso de la marca "AWS" o nombres de servicios registrados de Amazon de forma que induzca a error sobre la oficialidad del software. No debe usarse como sustituto de producción para servicios que requieran alta disponibilidad o garantías de seguridad de grado cloud.

Seguridad y certificaciones
- **Seguridad**: Al ejecutarse en local, la seguridad depende totalmente de los perímetros de seguridad de la red corporativa. No expone puertos al exterior de forma predeterminada a menos que se configure el contenedor para ello.
- **Certificaciones**: Al ser un proyecto de código abierto comunitario, carece de certificaciones tipo ISO 27001

o SOC2 individuales, por lo que la empresa debe incluirlo en su propio análisis de riesgos.

Otros
Es importante recalcar que, a diferencia de otras herramientas similares que se han movido a licencias propietarias o de "fuente disponible" (como el reciente cambio de LocalStack), Floci garantiza la soberanía tecnológica de la empresa al no depender de una conexión activa a internet ni de la validación de una licencia en un servidor centralizado. Esto es crítico para empresas que manejan secretos industriales o trabajen en entornos altamente regulados.

Fuentes consultadas:
- Licencia MIT de Floci
- Condiciones de almacenamiento y persistencia
- Arquitectura y diseño de servicios
- Repositorio de código oficial

Para más información y herramientas:

Explora look4.tools para descubrir las mejores soluciones tecnológicas del mercado.

Inicio Todas las herramientas Categorías