SECURITY PACK · LYRIKS

Lyriks security pack

For CISOs and infrastructure teams. How Lyriks is deployed, which network flows it opens, where your data lives and what goes to your LLM.



lyriks.io

CONTENTS

1. Summary for the CISO

2. What Lyriks is, in one paragraph

3. Deployment model

4. Network flows

5. Data handled and where it lives

6. The LLM question

7. Open source and auditability

8. The vendor

9. Pre-filled answers for your security questionnaire

10. Checklist for your own deployment review

This document is written for the people who have to approve, install or audit a tool before a development team can use it. It sticks to what Lyriks states publicly about its product and its installation. Where a question is not covered by those statements, the document says so and tells you whom to ask. It does not fill gaps with assumptions.

1. Summary for the CISO

> SIX FACTS TO START WITH
>
> - **It runs on your servers.** Lyriks installs on-premise, in containers, database included. It can run air-gapped.
> - **No outbound connection is required to run it.** The only outbound flow documented for installation is HTTPS to `registry.lyriks.io`, to pull container images. In isolated environments, these images can be delivered another way.
> - **You choose the LLM.** Hosted, local or sovereign, with your own tokens. Lyriks sends it nothing beyond what you already give that LLM to work on your code.
> - **The engine is open source.** It is published under the AGPL. The install scripts are readable before you run them.
> - **The project memory belongs to you.** It lives in a graph database on your side and exports in full, with its traceability.
> - **The vendor owns its technology.** Lyriks owns 100% of its intellectual property and built its engine in-house. Product telemetry is off by default.

2. What Lyriks is, in one paragraph

Lyriks is a deterministic software development life cycle tool. It reads an existing codebase and produces its specification: features, user journeys, actions, business rules, screens, objects and attributes, invariants, testable requirements. The same specification covers the technical side: architecture, modules, dependencies, data model, APIs, flows, integrations. Each rule points to the code that implements it. When a team plans a change, Lyriks computes what it touches before any code is written. When code is delivered, by a developer or by an AI coding assistant, Lyriks checks it against the specification, rule by rule.

The control loop is deterministic. The verdict comes from an engine based on graph rewriting, built in-house; a patent is pending. The same question always gets the same answer. For a security reviewer, this matters in one specific way: the checks Lyriks reports are reproducible and auditable, and they do not depend on a model's sampling.

3. Deployment model

Where it runs

Lyriks installs on infrastructure you control. The on-premise edition can operate air-gapped. Nothing in the product requires a vendor-hosted service to run. Your code, your LLM and your project memory stay on your servers.

How it is packaged

Everything runs in containers, including the database. On Linux and macOS, the installer sets up Docker Compose if needed. On Windows, it sets up WSL2 and Docker Desktop if needed. The supported architecture is x86-64; Apple Silicon machines run it through Rosetta.

How it is installed

Installation is one command.

SYSTEM	COMMAND	SCRIPT YOU CAN READ FIRST
Linux and macOS	`curl -fsSL https://get.lyriks.io | sh`	get.lyriks.io/install.sh
Windows (PowerShell)	`irm https://get.lyriks.io/windows | iex`	get.lyriks.io/windows

The installer works in six steps.

1. It checks the machine: x86-64, operating system, WSL2 on Windows, Docker and Compose.
2. It proposes what is missing. Required packages are installed if needed; Docker is offered if absent.
3. It installs Lyriks in containers, database included.
4. It runs a self-test and reports success only if the installation works.
5. It prints the address of the application, by default `http://localhost:3000`.
6. You claim the installation. Lyriks Community Edition is bound to the email address used to retrieve the licence key.

> **PIPING A SCRIPT INTO A SHELL**
>
> Many security teams do not allow `curl | sh` on production hosts. You do not have to. Download the script from the address above, read it, store a reviewed copy in your own repository and run that copy. The script you read is the script the command runs.

Starting small

Lyriks recommends starting with a pilot on one or two modules. For a security team, a pilot is also the simplest way to observe the actual behavior of the containers on an isolated host before wider approval.

---

4. Network flows

The table lists every flow that the public documentation describes. Anything not listed here should be treated as unknown until confirmed, and checked during your pilot with your own network monitoring.

PHASE	FLOW	DIRECTION	REQUIRED?	NOTES
Retrieving the installer	HTTPS to `get.lyriks.io`	Outbound	Only if you run the one-line command	Can be replaced by a reviewed local copy of the script.
Installation	HTTPS to `registry.lyriks.io`	Outbound	Yes, unless images are delivered another way	Pulls the container images. In isolated environments, images can be delivered without this flow.
Installation	Package sources for Docker, Compose, WSL2	Outbound	Only if these are missing	Not needed if your host already has them.
Operation	None to the vendor	n/a	No	No outbound connection is required to run Lyriks.
Operation	Users to the web application	Inbound, internal	Yes	Served at `http://localhost:3000` by default. Exposure beyond the host is your decision.
Operation	Lyriks to your sources (code, tickets, documentation)	Internal	Depends on the sources you connect	Destinations are your own systems.
Operation	Coding assistant to Lyriks, through MCP	Internal	Only if you use the production and verification loop	Claude Code, Cursor and Copilot are the documented clients.
Operation	To your LLM	Depends on your LLM	Your choice	Local or sovereign model: the flow stays where you host it. Hosted model: the flow goes where your LLM contract already sends it.
Operation	Product telemetry	Outbound	No. Off by default	If you turn it on: DO_NOT_TRACK is honored, no IP address is stored, the collector is hosted in the European Union.

> IN PRACTICE
>
> Once the images are on the host, an air-gapped deployment needs no route to the Internet. The flow to your LLM is the only one whose destination depends on a choice you make, and it is the same flow your teams already open when they use that LLM.

---

5. Data handled and where it lives

What Lyriks reads

Lyriks builds a single knowledge base from the sources you connect. The documented sources are:

- your source code, including repositories on GitHub;
- your tickets, in Jira;
- your documentation, in Confluence and Notion;
- your design files, in Figma.

Lyriks reads these sources to link them and to flag inconsistencies between them, for example a refund delay that says 14 days in a ticket, 30 days in the documentation and 21 days in the code. Which sources you connect, and with which access rights, is up to you. A read-only account per source is a reasonable default to evaluate during a pilot.

What Lyriks produces

- The retro-specification: features, journeys, actions, business rules, screens, objects and attributes, invariants, testable requirements.
- The retro-documentation: architecture, modules, dependencies, data model, APIs, flows, integrations, technologies.
- Impact reports for planned changes: what is touched, what conflicts, what is not touched, what must be tested, the decision to make.
- Compliance reports at each release: compliant, added without being requested, planned but not implemented, regression.

Where it is stored

The knowledge base is a graph database, not a retrieval index over text chunks. Every element is linked to its source. The database runs in a container on your host, alongside the application.

What happens when you stop

The project memory is company property. It exports in full, with its objects, attributes and traceability. It is independent of the LLM you use: you can change models without losing it.

DATA	WHERE IT LIVES	WHO CONTROLS IT
Source code	Your repositories; read by Lyriks on your host	You
Tickets, documentation, design files	Your tools; read by Lyriks on your host	You
Specification, documentation, reports	Graph database, in a container on your host	You
Prompts and context sent to an LLM	Your LLM, wherever you host it	You and your LLM provider, under your existing terms
Licence key request (email, company)	Submitted to Lyriks to obtain the key	Lyriks, under its privacy policy

6. The LLM question

This is usually the first question a security team asks about any tool that touches code and AI. Lyriks answers it with three statements.

It is your LLM

Lyriks uses the LLM you choose, hosted, local or sovereign, with your tokens. Lyriks does not supply a model and does not route your traffic through a model of its own.

Lyriks sends nothing beyond what you already give your LLM

When a validated specification goes to a coding assistant, it travels with its rules and their sources, through MCP, to the assistant you already use (Claude Code, Cursor or Copilot). The assistant writes the code. Lyriks then checks the result rule by rule and sends each gap back to the assistant, until every rule is met. The material involved is your code and the specification derived from it, which is what your teams already give that assistant when they work on the same codebase.

The LLM is not in the control loop

The verification is done by the deterministic engine, not by the LLM. A model can write code; it does not decide whether that code complies. This separation is what allows Lyriks to say that the same question always gets the same answer.

> FOR YOUR REVIEW
>
> Your existing LLM policy applies unchanged. If a model is already approved for your code, using it with Lyriks does not widen what it sees. If no model is approved yet, a local model keeps the whole chain inside your perimeter.

7. Open source and auditability

The licence

The Lyriks engine is published under the GNU Affero General Public License (AGPL). You can read the code, inspect it and verify what it does. The AGPL also carries obligations if you modify the software and make it available to others over a network; your legal team will know how your organization handles AGPL components.

The installer

The scripts behind the one-line commands are published at get.lyriks.io/install.sh and get.lyriks.io/windows. They can be read, versioned and run from your own copy.

The results

Every rule, gap and decision Lyriks reports is linked to its source. A reviewer can trace a finding back to the line of code, the ticket or the page it comes from, then challenge it or defend it. Because the engine is deterministic, running the same check twice on the same inputs gives the same result, which makes the results usable as evidence.

8. The vendor

Lyriks owns 100% of its intellectual property. The engine is built in-house, and a patent is pending. Your code and your project memory stay on your infrastructure, whatever the vendor does.

Security and compliance questions go to contact@lyriks.io. This is also the address for the items marked "To confirm" in the next section.

9. Pre-filled answers for your security questionnaire

The answers below are limited to what Lyriks publishes about its product and installation. Where the published material does not cover a question, the answer says so. Please do not treat a "To confirm"

line as a yes or a no.

QUESTION	ANSWER
Is the solution SaaS or self-hosted?	Self-hosted. Lyriks installs on your servers, on-premise.
Can it run in an air-gapped environment?	Yes. The on-premise edition can operate air-gapped. Container images can be delivered without access to `registry.lyriks.io`.
Does it require outbound Internet access in operation?	No outbound connection is required to run it.
Which outbound flows are needed at installation?	HTTPS to `registry.lyriks.io` to pull images, unless they are delivered another way. Package sources for Docker, Compose or WSL2 only if missing.
How is it packaged?	Containers, database included. Docker Compose on Linux and macOS; WSL2 and Docker Desktop on Windows.
Which architectures are supported?	x86-64. Apple Silicon runs through Rosetta.
Can we review the install script before running it?	Yes, at get.lyriks.io/install.sh and get.lyriks.io/windows.
Is the source code available?	Yes. The engine is open source under the AGPL.
Which LLM does the product use?	The one you choose, hosted, local or sovereign, with your tokens.
What data is sent to the LLM?	Nothing beyond what you already give your LLM to work on your code: the specification, its rules and their sources, sent to your coding assistant through MCP.
Does the vendor train models on our data?	Lyriks does not supply a model; the LLM is yours. For any further commitment, to confirm with Lyriks: contact security questions to contact@lyriks.io.
Is AI involved in the compliance verdict?	No. Verification is performed by a deterministic engine based on graph rewriting.
Where is customer data stored?	On your host, in a graph database running in a container.
Can we export all our data?	Yes. The project memory exports in full, with its objects, attributes and traceability.
Is there telemetry?	Off by default. If enabled: DO_NOT_TRACK is honored, no IP address is stored, the collector is hosted in the European Union.
Who owns the technology?	Lyriks, 100%. The engine is built in-house and a patent is pending.

QUESTION	ANSWER
How is the Community Edition licensed to a user?	It is bound to the email address used to retrieve the licence key.
Does the licence key require online validation?	To confirm with Lyriks: contact security questions to contact@lyriks.io.
SOC 2, ISO 27001 or other certifications?	To confirm with Lyriks: contact security questions to contact@lyriks.io.
Penetration test reports?	To confirm with Lyriks: contact security questions to contact@lyriks.io.
Encryption at rest?	To confirm with Lyriks: contact security questions to contact@lyriks.io.
Encryption in transit between users and the application?	The default address is `http://localhost:3000`. For TLS options, to confirm with Lyriks: contact security questions to contact@lyriks.io.
SSO (SAML, OIDC), LDAP or directory integration?	To confirm with Lyriks: contact security questions to contact@lyriks.io.
Role-based access control and user audit logs?	To confirm with Lyriks: contact security questions to contact@lyriks.io.
Data retention and deletion policy?	Data stays on your host, so retention follows your own operations. For product-level retention or purge features, to confirm with Lyriks: contact security questions to contact@lyriks.io.
Backup and restore procedure?	To confirm with Lyriks: contact security questions to contact@lyriks.io.
How are updates and security patches delivered?	To confirm with Lyriks: contact security questions to contact@lyriks.io.
Are container images signed? Is an SBOM available?	To confirm with Lyriks: contact security questions to contact@lyriks.io.
Vulnerability disclosure process?	To confirm with Lyriks: contact security questions to contact@lyriks.io.
Hardware sizing requirements?	x86-64 with Docker is documented. For CPU, memory and storage sizing, to confirm with Lyriks: contact security questions to contact@lyriks.io.

QUESTION	ANSWER
Data processing agreement and subprocessors?	To confirm with Lyriks: contact security questions to contact@lyriks.io.
Support levels and response times?	To confirm with Lyriks: contact security questions to contact@lyriks.io.

10. Checklist for your own deployment review

These checks are for your team to run on your own pilot. They do not depend on anything the vendor tells you.

- ☐ Read `install.sh` or the Windows script, store a reviewed copy and run that copy rather than the piped command.
- ☐ Decide how container images reach the host: direct pull from `registry.lyriks.io`, or delivery into an isolated environment.
- ☐ Install on a dedicated host or VM for the pilot, scoped to one or two modules.
- ☐ After installation, confirm the self-test passed and the application answers on the expected address.
- ☐ Monitor outbound traffic from the containers during normal use and compare it with the flows in section 4.
- ☐ Confirm telemetry is off, or turn it on deliberately and record that decision.
- ☐ Decide whether the web application stays bound to the host or is exposed to the internal network, and put it behind your usual reverse proxy and TLS if exposed.
- ☐ Connect each source (code, tickets, documentation, design files) with a dedicated account and the least privilege that works.
- ☐ Point the coding assistant at an LLM already approved by your policy, or at a local model.
- ☐ Run a full export of the project memory and check that you can read it without Lyriks.
- ☐ Include the container volumes, database included, in your backup scope.
- ☐ Have your legal team review the AGPL against your internal rules for open source components.
- ☐ Send the open "To confirm" questions from section 9 to contact@lyriks.io and file the answers with this pack.

Next step

The security page summarizes these commitments and is kept up to date at www.lyriks.io/security. To see Lyriks on a codebase and go through your questions with the team, book a demo at www.lyriks.io/demo.