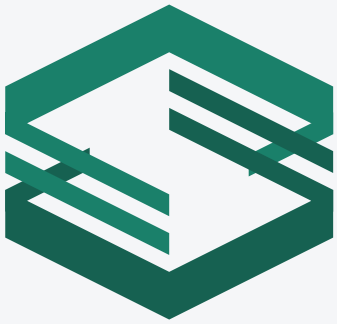
Manual 01 · install, check, first run

Getting Started & Installation



What to install on each machine, which system libraries and database drivers each piece needs, and a first end-to-end run — from a pipeline in the studio to a signed artifact executed by the runner — in about five minutes.

CONTENTS

01 The suite in one page

02 System requirements

03 Install the studio
- 3.1 Windows
- 3.2 Linux

04 Install the runner
- 4.1 Linux
- 4.2 Windows

05 Install the monitor (optional)

06 Databases and drivers
- 6.1 Oracle Instant Client
- 6.2 Informix Client SDK
- 6.3 Check a connection

07 Your first five minutes
- 7.1 Run it in the studio
- 7.2 Create your signing key (once per computer)
- 7.3 Compile and sign the artifact
- 7.4 Run it with the runner
- 7.5 Exit codes

08 Files and folders

09 When something does not start

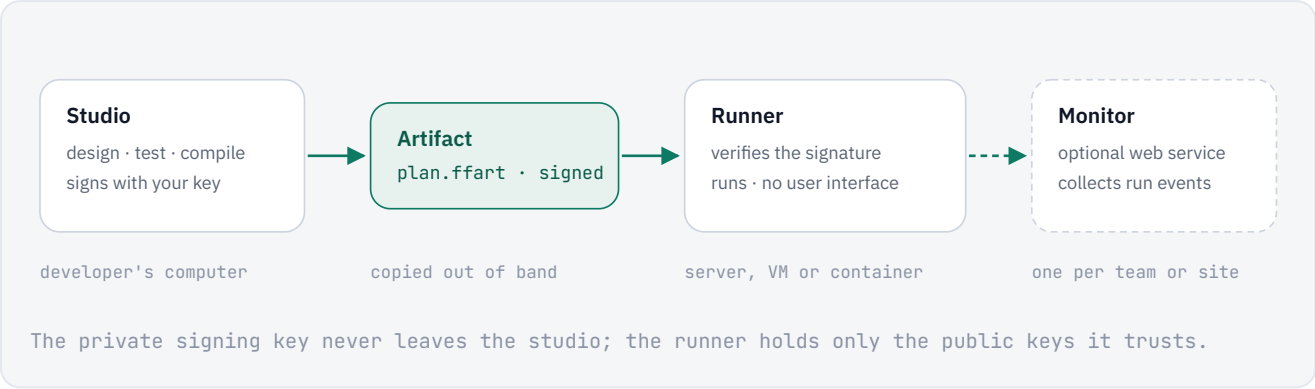
10 For beta testers
- 10.1 Which version you have
- 10.2 What to try
- 10.3 How to report a problem
- 10.4 Known limits of the beta

11 Appendix — building from source

Copyright © 2025–2026 Marco Dongu. All rights reserved.

01 The suite in one page

Stilton is made of four pieces. They can live on the same computer while you learn, and on different machines in production.



Piece	What it is	Where it runs	Needs
Studio	Desktop application: canvas, node panels, previews, local runs, compilation and signing of artifacts	The developer's computer (Windows or Linux)	A graphical session; WebView2 on Windows, WebKitGTK on Linux
Artifact	A `.ffart` file: the compiled plan plus a signed manifest. It contains the **names** of the secrets it needs, never their values	Copied to the run machine	—
Runner	`stilton-runner`, a command-line program that verifies an artifact and executes it	Any Windows or Linux machine, also headless	No graphical libraries
Monitor	`stilton-monitor`, a small web service that receives the events of every run and shows them	Usually one machine reachable by all runners	Nothing beyond the binary

Project files (`.ffplan`) are what you edit in the studio; artifacts (`.ffart`) are what you deploy. The studio can run a project directly while you design it; the runner only executes signed artifacts.

02 System requirements

	Studio	Runner	Monitor
Windows	Windows 10 or 11, 64-bit	Windows 10/11 or Server 2019+, 64-bit	same as the runner
Linux	64-bit desktop distribution with WebKitGTK 4.1: Ubuntu 22.04 or later, Debian 12 or later	64-bit distribution with glibc 2.35 or later: Ubuntu 22.04 or later, Debian 12 or later	same as the runner
macOS	not part of the beta	not part of the beta	not part of the beta

	Studio	Runner	Monitor
Memory	4 GB minimum, 8 GB recommended	tens of MB for small flows; grows with the datasets held in memory (aggregate, sort, join)	grows with the stored runs
Disk	300 MB	40 MB plus run logs	the stored run events

Note. The beta is built for 64-bit x86 processors (`x86_64` / `x64`). ARM builds are not part of the beta.

03 Install the studio

Windows

01 Run the installer from the beta package: `Stilton-Studio_0.9.0_x64-setup.exe`, or the `.msi` version for managed installations.

02 If Windows shows *"Windows protected your PC"*, choose **More info → Run anyway**. The beta installers are not yet signed with a code-signing certificate; the warning disappears in a later release.

03 The installer adds **Microsoft Edge WebView2** if the computer does not already have it (Windows 11 always has it).

04 Start **Stilton Studio** from the Start menu.

Linux

Install the package for your distribution. The package manager pulls in the libraries the studio needs.

Debian / Ubuntu — bash

```
sudo apt install ./Stilton*_0.9.0_amd64.deb
```

Fedora / RHEL — bash

```
sudo dnf install ./Stilton*-0.9.0-1.x86_64.rpm
```

The AppImage needs no installation, but it uses the system libraries: WebKitGTK 4.1 and the ODBC driver manager must be present.

AppImage — bash

```
sudo apt install libwebkit2gtk-4.1-0 unixodbc     # Debian / Ubuntu
chmod +x Stilton*_0.9.0_amd64.AppImage
./Stilton*_0.9.0_amd64.AppImage
```

Warning. On Linux the studio and the runner are linked to the ODBC driver manager (`libodbc.so.2`). Without the `unixodbc` package they do not start at all — even if you never use Oracle or Informix. The `.deb` and `.rpm` packages declare it; with the AppImage or a bare binary, install it yourself.

04 Install the runner

The runner is a single executable. There is nothing to register and no service is installed for you: copy it where you keep your tools and, if you like, add that folder to the `PATH`.

Linux

Linux — bash

```
sudo install -m 755 stilton-runner-linux-x64 /usr/local/bin/stilton-runner
sudo apt install libssl3 libdbus-1-3 unixodbc        # Debian / Ubuntu
# sudo dnf install openssl-libs dbus-libs unixODBC   # Fedora / RHEL
stilton-runner --help
```

Library	Why the runner needs it
`libssl3` (OpenSSL 3)	TLS towards databases, HTTP services, mail, MQTT, FTP and the monitor
`libdbus-1-3`	access to the system keychain (Secret Service) for secrets
`unixodbc`	the ODBC driver manager — required to start, used by Oracle, Informix and generic ODBC sources

Windows

Windows (PowerShell) — powershell

```
New-Item -ItemType Directory -Force C:\Stilton | Out-Null
Copy-Item .\stilton-runner-windows-x64.exe C:\Stilton\stilton-runner.exe
C:\Stilton\stilton-runner.exe --help
```

Windows already provides TLS, the credential store and the ODBC driver manager. If the runner reports a missing `VCRUNTIME140.dll`, install the *Microsoft Visual C++ Redistributable (x64)*.

Tip. Run the runner as the account that will execute the jobs in production — a service account, not your own. The trust store, the run logs and the keychain secrets all belong to that account (see the *Runtime Operations* manual).

05 Install the monitor (optional)

The monitor is also a single executable. It refuses to start without two access tokens: one that runners use to send events, one that people use to read them.

Linux — bash

```bash
sudo install -m 755 stilton-monitor-linux-x64 /usr/local/bin/stilton-monitor
export MONITOR_TOKEN=$(openssl rand -hex 32)        # runners → monitor
export MONITOR_READ_TOKEN=$(openssl rand -hex 32)   # people → monitor
stilton-monitor 8787
```

Windows (PowerShell) — powershell

```powershell
$env:MONITOR_TOKEN = "<a long random value>"
$env:MONITOR_READ_TOKEN = "<another long random value>"
C:\Stilton\stilton-monitor.exe 8787
```

Open `http://localhost:8787`: the browser asks for a password — any user name, password = the read token. The monitor listens only on `127.0.0.1` by default and speaks plain HTTP; to reach it from other machines put it behind a reverse proxy with TLS. Tokens, network exposure and storage are covered in the *Monitor* and *Security & Certificates* manuals.

06 Databases and drivers

What a pipeline can reach depends on what is installed **on the machine that runs it** — the studio's computer for previews and local runs, the runner's machine in production. Nothing about drivers travels inside an artifact.

Database	How Stilton connects	What to install
PostgreSQL	built-in driver	nothing
MySQL / MariaDB	built-in driver	nothing
SQLite	built-in (embedded)	nothing
Oracle	ODBC	ODBC driver manager + Oracle Instant Client *Basic* and *ODBC*
Informix	ODBC	ODBC driver manager + Informix Client SDK
SQL Server, DB2, others	ODBC (generic)	the vendor's ODBC driver, registered on the machine

All three pieces of an ODBC connection — the driver manager, the vendor's client library and its ODBC driver — must be **64-bit**.

Oracle Instant Client

Linux — bash

```
sudo apt install unixodbc                       # driver manager (dnf: unixODBC)
# unpack "Basic" and "ODBC" of the same version into the same folder:
cd /opt/oracle/instantclient_23_26
sudo ./odbc_update_ini.sh / /opt/oracle/instantclient_23_26
echo /opt/oracle/instantclient_23_26 | sudo tee /etc/ld.so.conf.d/oracle-ic.conf
sudo ldconfig
odbcinst -q -d                                  # the Oracle driver must be listed
```

On Windows, unpack *Basic* and *ODBC* into the same folder, run `odbc_install.exe` from there and add the folder to the `PATH`. The driver then appears in the *ODBC Data Sources (64-bit)* tool (`odbcad32.exe`, tab *Drivers*).

In the studio, an Oracle resource needs host, port (1521), service name, user, password and the **ODBC driver** name exactly as registered (or the full path of `libsqora.so`). The connection uses EZConnect (`host:port/service`): no `tnsnames.ora` is required.

Informix Client SDK

Linux — bash

```
export INFORMIXDIR=/opt/informix
export LD_LIBRARY_PATH=$INFORMIXDIR/lib:$INFORMIXDIR/lib/cli:$INFORMIXDIR/lib/esql:$LD_LIBRARY_PATH
```

/etc/odbcinst.ini — ini

```
[IBM INFORMIX ODBC DRIVER]
Description = Informix ODBC (CSDK)
Driver      = /opt/informix/lib/cli/iclit09b.so
```

`IBM INFORMIX ODBC DRIVER` is the name Stilton uses when the resource leaves the driver field empty. For accented characters add `CLIENT_LOCALE=en_US.utf8;` (and `DB_LOCALE` matching the database) to the resource's *additional parameters*. On Windows the SDK registers the driver itself; check its exact name in `odbcad32.exe`.

> **Note.** When the runner runs as a `systemd` service, variables such as `INFORMIXDIR`, `LD_LIBRARY_PATH`, `TNS_ADMIN` or `NLS_LANG` must be set in the unit file (`Environment=`), not in a user's shell profile.

Check a connection

From the studio, use **Test connection** on the resource. From a terminal with unixODBC:

Linux — bash

```
odbcinst -q -d                # registered drivers
odbcinst -q -s                # registered data sources (DSN)
isql -v -k "Driver={Oracle 23ai ODBC driver};DBQ=dbhost:1521/ORCL;UID=app;PWD=…"
```

07 Your first five minutes

This walk-through uses the example **01-file-to-file** from the `examples` folder of the beta package: it reads a CSV of 20 customers, adds two computed fields and writes a new CSV. It needs no database and no network.

Run it in the studio

01 Start Stilton Studio and choose **Project → Open…**. Open `examples/01-file-to-file/piano.ffplan`.

02 Look at the canvas: a lane with three nodes in a row — *Sorgente clienti* (file input), *Arricchimento* (transform), *Uscita clienti* (file output). Click a node to see its settings in the side panel.

03 Press **Run**. The nodes turn green one after the other; the log panel at the bottom reports rows in and out for each node. The output file appears in `_out/` inside the example folder.

Create your signing key (once per computer)

04 Press **Signing**. The panel says *No signing key on this computer*. Under **Create the signing key**, type a passphrase and press **Generate from system**. Keep the passphrase safe: if you forget it the key is unrecoverable, and you will have to create a new one and authorise it again on every runner.

05 In the same panel, under *Export the public key*, press **Copy trust store entry** and save the text into a file named `entry.json`. This is the public half of your key — safe to share.

Compile and sign the artifact

06 Press **Compile**. Leave the profile and the monitor empty for now, choose the target platform and the log detail (*Normal*), then press **Generate →**.

07 Save the artifact as `artifact.ffart` **inside the example folder** — the example uses paths relative to its own folder. The file comes out signed.

Run it with the runner

08 Authorise your own key on this machine — once, as the user who will run the runner:

Linux bash

```
stilton-runner trust add entry.json --owner your-name
#   key authorised
#     keyId (fingerprint): ed25519:9f3c…
```

Windows (PowerShell) powershell

```
C:\Stilton\stilton-runner.exe trust add entry.json --owner your-name
```

09 Run the artifact from the example folder and keep the log:

Linux bash

```
cd examples/01-file-to-file
stilton-runner artifact.ffart > run.ndjson
echo "exit code: $?"
```

Windows (PowerShell) powershell

```
cd examples\01-file-to-file
C:\Stilton\stilton-runner.exe artifact.ffart > run.ndjson
echo "exit code: $LASTEXITCODE"
```

10 The runner first prints `signature OK: artifact signed by your-name (ed25519:…)` on the error stream, then the run events, one JSON object per line. Exit code **0** means completed, and `_out/clienti_arricchiti.csv` has 20 rows.

Note. The runner executes **only** artifacts signed by a key listed in its trust store, and the check cannot be switched off. With an empty trust store nothing runs — that is the expected behaviour, not an error. The *Build & Publish* and *Security & Certificates* manuals explain keys, rotation and revocation.

Exit codes

Code	Meaning
`0`	the run completed
`1`	the run started and failed (a node failed and the error handler did not recover)
`2`	wrong usage, or the artifact file cannot be read
`3`	the artifact was **refused** before running: signature, trust store, engine version, runtime profile or configuration. Nothing was executed

08 Files and folders

Every piece keeps its state in a `.stilton` folder in the home directory of the account that runs it. Nothing is written next to the program.

Path (Linux)	Path (Windows)	Content	Written by
`~/.stilton/studio.json`	`%USERPROFILE%\.stilton\studio.json`	the studio's identity (label shown in the monitor)	studio
`~/.stilton/signing-key.json`	`%USERPROFILE%\.stilton\signing-key.json`	your **private** signing key, encrypted with your passphrase	studio
`~/.stilton/trust-store.json`	`%USERPROFILE%\.stilton\trust-store.json`	the public keys this machine trusts	runner (`trust add`)
`~/.stilton/runtime.json`	`%USERPROFILE%\.stilton\runtime.json`	optional runtime profile: which node types this machine accepts	administrator
`~/.stilton/runs/`	`%USERPROFILE%\.stilton\runs\`	the complete local log of every run	studio and runner
`~/.stilton/monitor-data/`	`%USERPROFILE%\.stilton\monitor-data\`	the runs received by the monitor	monitor

Secrets live in the environment of the run machine or in the system keychain (service name `stilton`), never in these files.

09 When something does not start

Symptom	Cause and remedy
`libodbc.so.2: cannot open shared object file`	The ODBC driver manager is missing: `sudo apt install unixodbc` (Fedora/RHEL: `unixODBC`).
`libssl.so.3` or `libdbus-1.so.3` not found	Install `libssl3` and `libdbus-1-3` (Fedora/RHEL: `openssl-libs`, `dbus-libs`).
The studio window stays blank on Linux	WebKitGTK 4.1 is missing or too old: install `libwebkit2gtk-4.1-0`, or use the `.deb` / `.rpm` package.
Windows protected your PC	Expected for the beta installers, which are not yet code-signed: **More info → Run anyway**.
`VCRUNTIME140.dll was not found`	Install the Microsoft Visual C++ Redistributable (x64).
`SIGNATURE REFUSED: trust store missing`	No key is authorised for this user on this machine: `stilton-runner trust add entry.json --owner NAME`.
`SIGNATURE REFUSED: trust store writable by ANYONE`	Restrict the file: `chmod 600 ~/.stilton/trust-store.json` (and the folder must not be writable by others).

Symptom	Cause and remedy
`Can't open lib '…'` (ODBC)	The vendor's driver is not installed, or `odbcinst.ini` points to a library without a full path. Put the full path in the resource's *ODBC driver* field, or fix `odbcinst.ini`.
`Data source name not found`	Wrong driver or DSN name. `odbcinst -q -d` lists drivers, `odbcinst -q -s` lists DSNs.
A `${NAME}` appears unchanged in the log	A secret is not set on the machine that runs: set it as an environment variable or in the keychain.

10 For beta testers

Thank you for trying Stilton before everyone else. This section says what is most useful to try and what to send when something goes wrong.

Which version you have

Open **Help**: the line at the bottom shows the studio version and the system, for example `Stilton Studio 0.9.0 · Windows NT 10.0 · WebView2 131.0…`. **Copy details** puts it in the clipboard. Start every report with it. The runner prints its own version with `stilton-runner --version`.

What to try

01 Install, then run **Your first five minutes** above, studio and runner, on your own system.

02 Follow the two tutorials in **Help → Guides**: *Your first flow* and *All or nothing — transactions* (this one needs a PostgreSQL database; the guide shows how to start one with Docker). Tell us where a step was unclear or did not tick.

03 Open the other examples (`02` to `08`): each one has a `NOTE.md` that says what it shows and what result to expect. Run them in the studio.

04 Build one flow of your own with your real data — a file, a database you use, an API. This is where the surprises are.

05 Try the things you do every day: drag nodes from the palette, connect, undo and redo, save and reopen, import lanes from another project, look at the **Monitor** tab while a flow runs.

06 If you have projects from an earlier build, open them: you should see a compatibility report, never an error (see *Projects from other versions of Stilton* in the Studio Guide).

How to report a problem

A good report lets us reproduce the problem on our side. Send:

What	Where to find it
Version and system	Help → **Copy details**
What you did, what you expected, what happened	a few lines, in order — *"opened example 05, pressed Run, the Join node stayed grey"*
The log panel	select the lines in the log panel at the bottom of the studio and copy them
The run log	`~/.stilton/runs/` (Windows: `%USERPROFILE%\.stilton\runs\`), one `.ndjson` file per run, named after the run id — send the most recent
For the runner	the command line, the exit code and the `run.ndjson` you redirected the output to
The project, if you can	the `.ffplan` file. It contains no secret values — only their names — but it does contain paths, queries and field names: send it only if that is acceptable for you
A screenshot	for anything visual: a node drawn wrong, a panel that does not open

Do not send real data or credentials. If the problem needs data to show up, a few made-up rows with the same columns are enough.

Known limits of the beta

- The installers are **not code-signed** yet: Windows shows *Windows protected your PC* (**More info → Run anyway**).
- There are **no automatic updates**: install the new version over the old one; projects, keys and settings stay in place.
- Windows and Linux only; macOS comes later.
- Projects saved with the beta keep opening in later versions. Artifacts compiled by Studio 0.9.x run on runner 0.9.0 or later.

11 Appendix — building from source

Testers receive ready-made packages. To build the suite yourself you need Git, Node.js 20 or later, Rust through `rustup` (the version is pinned by `src-tauri/rust-toolchain.toml` and downloaded automatically) and, on Windows, the Visual Studio Build Tools with the *Desktop development with C++* workload.

Linux — system packages — bash

```
sudo apt install build-essential curl wget file pkg-config \
  libwebkit2gtk-4.1-dev librsvg2-dev libssl-dev libdbus-1-dev \
  libayatana-appindicator3-dev libxdo-dev unixodbc-dev
```

Build bash

```
npm install
npm run tauri build                          # studio installers → src-tauri/target/release/bundle/
cd src-tauri
cargo build --bin stilton-runner  --no-default-features --profile release-lean
cargo build --bin stilton-monitor --no-default-features --features monitor --release
```

The runner and the monitor are always built with `--no-default-features`: without it Cargo tries to include the desktop libraries they do not need.