FOSDEM 2026

Building a microkernel-based operating system from the ground up

Presented by **Clem**

skift OS

Slides and links

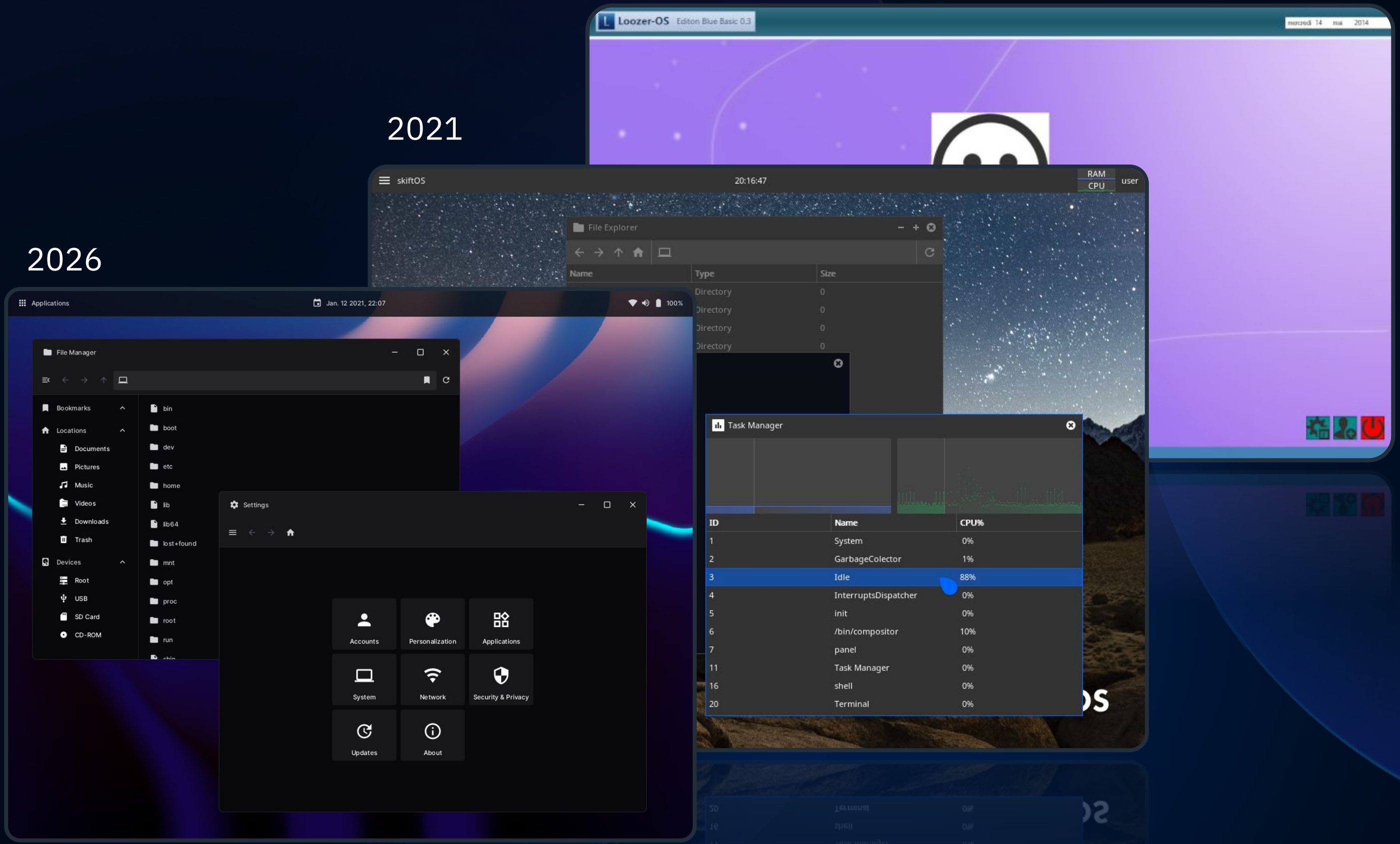
QUESTION

WHAT HAPPENS WHEN YOU DESIGN THE ENTIRE STACK?

2014

2021

2026



Name	Type	Size
	Directory	0
	Directory	0
	Directory	0
	Directory	0

ID	Name	CPU%
1	System	0%
2	GarbageColector	1%
3	Idle	88%
4	InterruptsDispatcher	0%
5	init	0%
6	/bin/compositor	10%
7	panel	0%
11	Task Manager	0%
16	shell	0%
20	Terminal	0%

ABOUT ME

Builds Browser by day and OS by night

`https://smnx.sh`

Clem

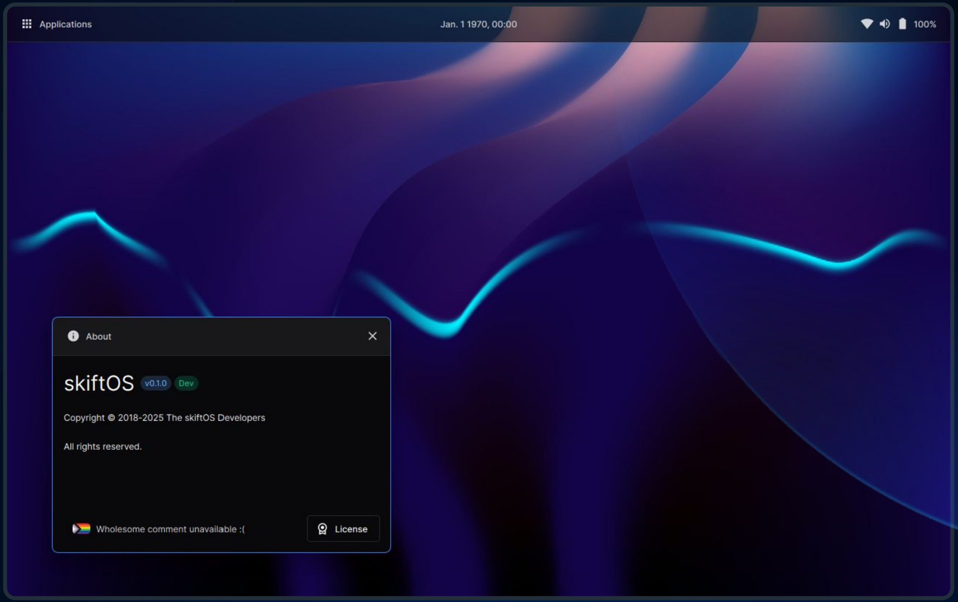
WHAT IS SKIFT OS?

General purpose

Microkernel

Rich C++ Framework

POSIX 🤮

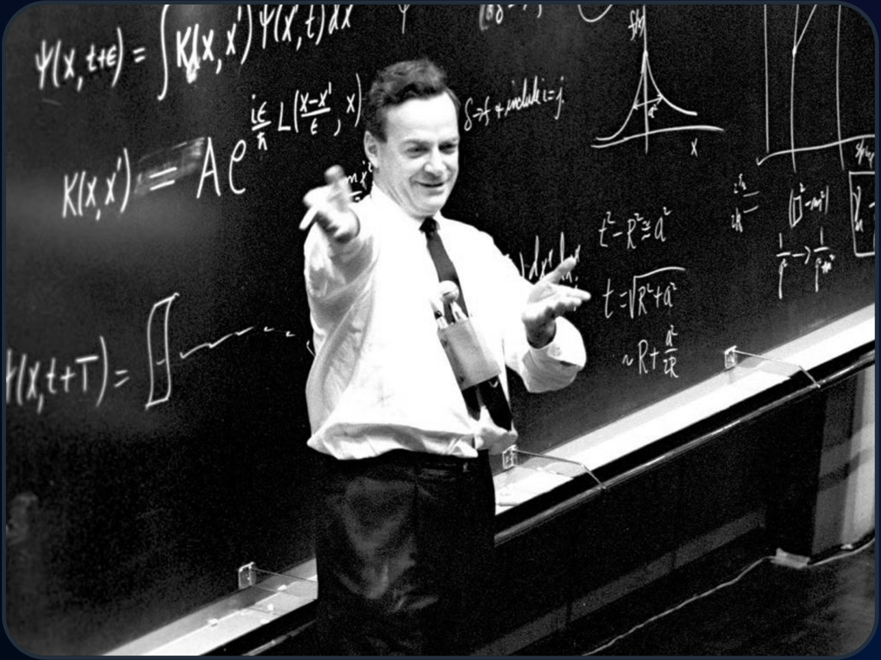


Copyright © 2018-2025 The skiftOS Developers

All rights reserved.

Wholesome comment unavailable :(

GOALS



Fun

Learning

Research

PROJECT STATUS

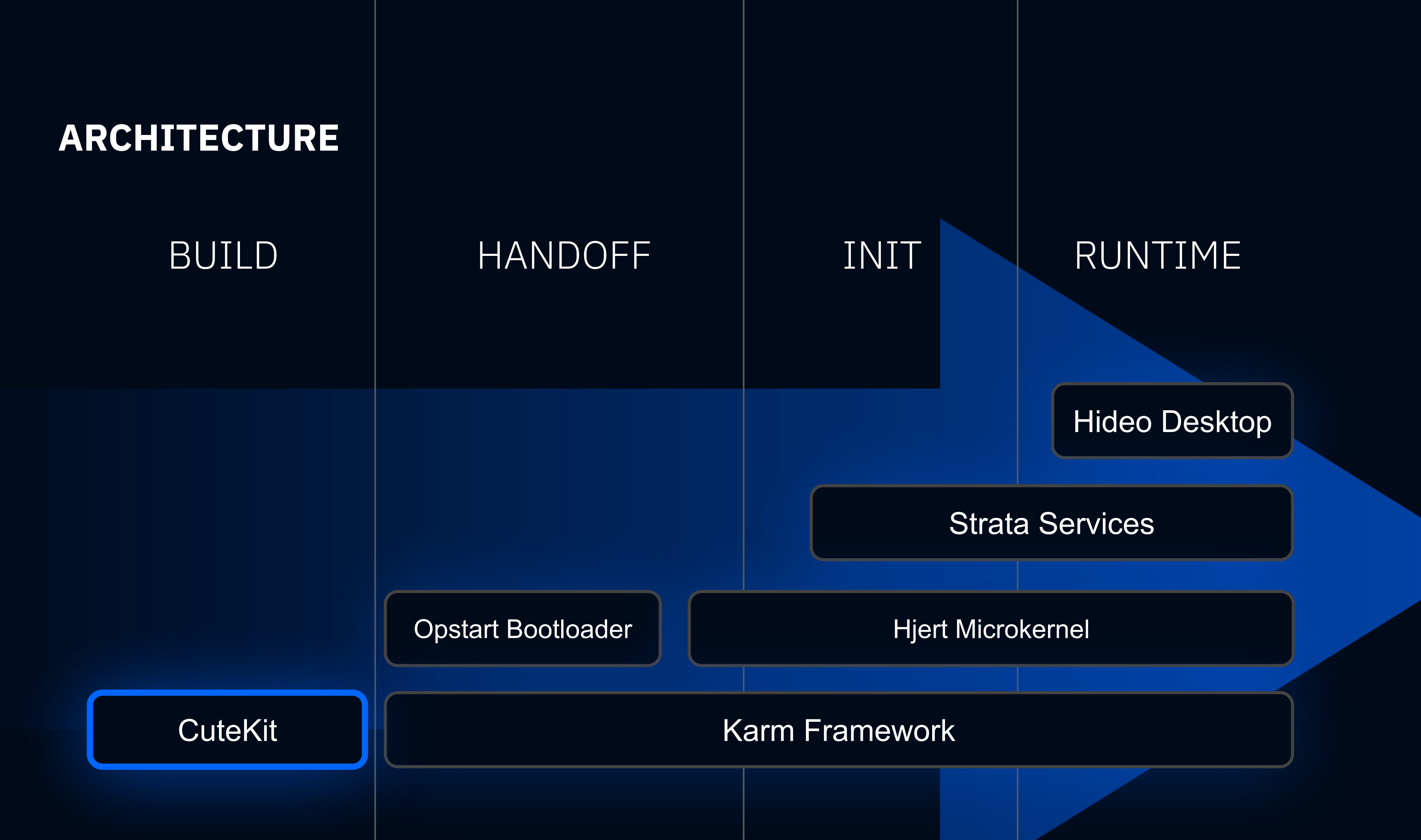
Early stage

Most components are POCs

~110k LoC

Full desktop

ARCHITECTURE



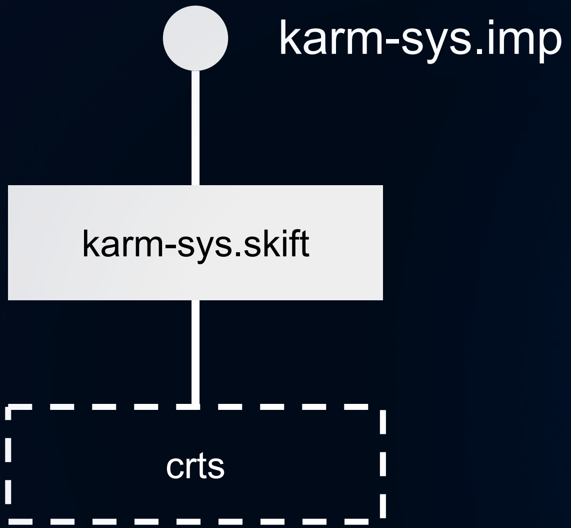
CUTEKIT

Cargo inspired

C++ 20 modules

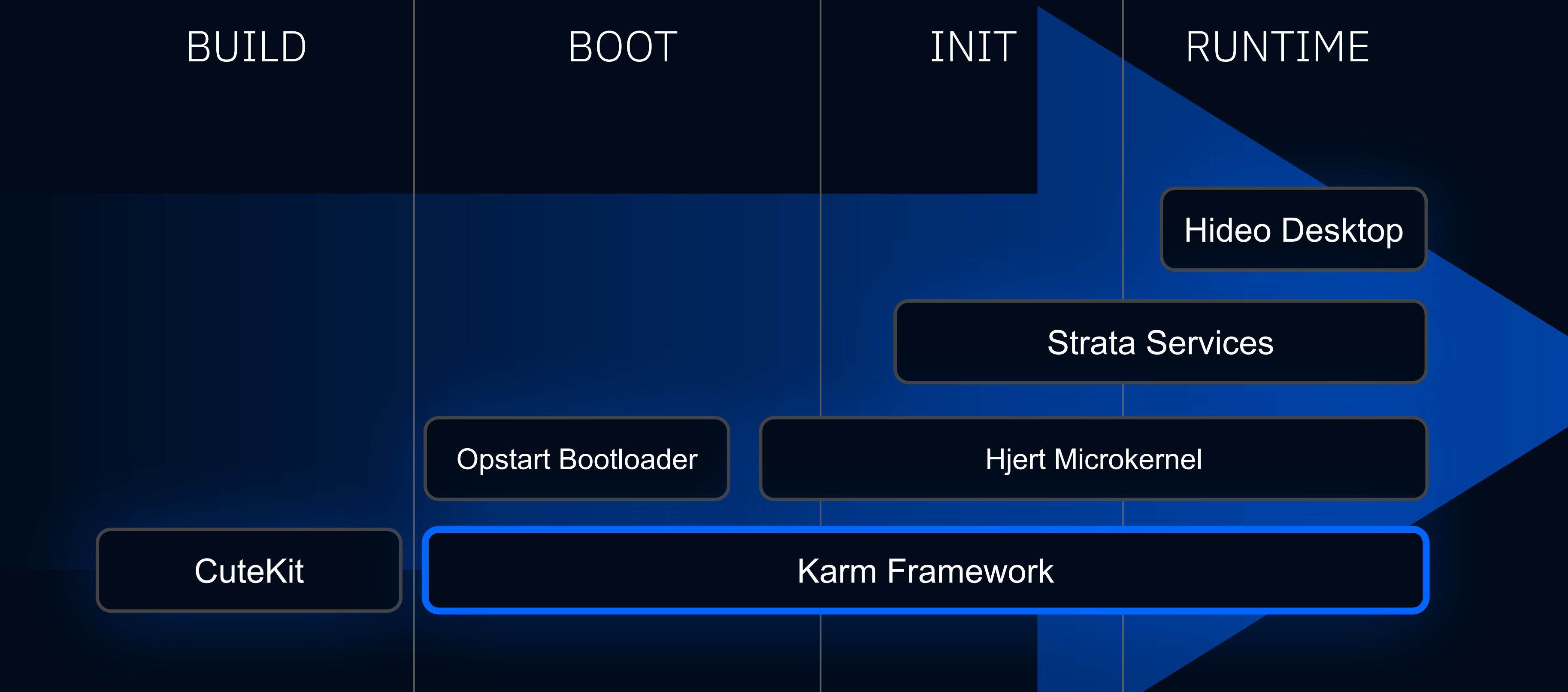
Battery included
(lint, fmt, fuzz, profile, test, build)

CUTEKIT COMPONENTS



```
{
    "id": "karm-sys.skift",
    "type": "lib",
    "enableIf": {
        "sys": [
            "skift"
        ]
    },
    "requires": [
        "crts",
        "..."
    ],
    "provides": [
        "karm-sys.impl",
        "..."
    ]
}
```

ARCHITECTURE

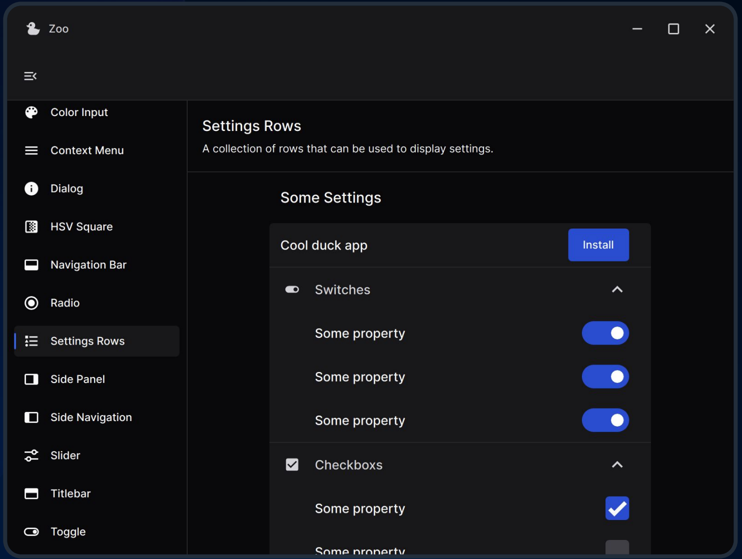


KARM

Freestanding core shared

between all components of the OS

Inspired by rust, go, and C#’s std



Settings Rows

A collection of rows that can be used to display settings.

Some Settings

Cool duck app

Some property

Some property

Some property

Some property

Some property

KARM-FLAVORED C++

C with lambda, modules, and coroutines

No exceptions

Res<> and try$(...)

strict Clang safety checks

[[clang::lifetimebound]]

-Wunsafe-buffer-usage ...

HELLO, WORLD!

```
import Karm.Core;
import Karm.Sys;
import Karm.Ui;

using namespace Karm;

Async::Task<> entryPointAsync(Sys::Context& ctx, Async::CancellationToken ct) {
    co_return co_await Ui::runAsync(
        ctx, Ui::labelMedium("Hello, world"), ct);
}
```

RPC

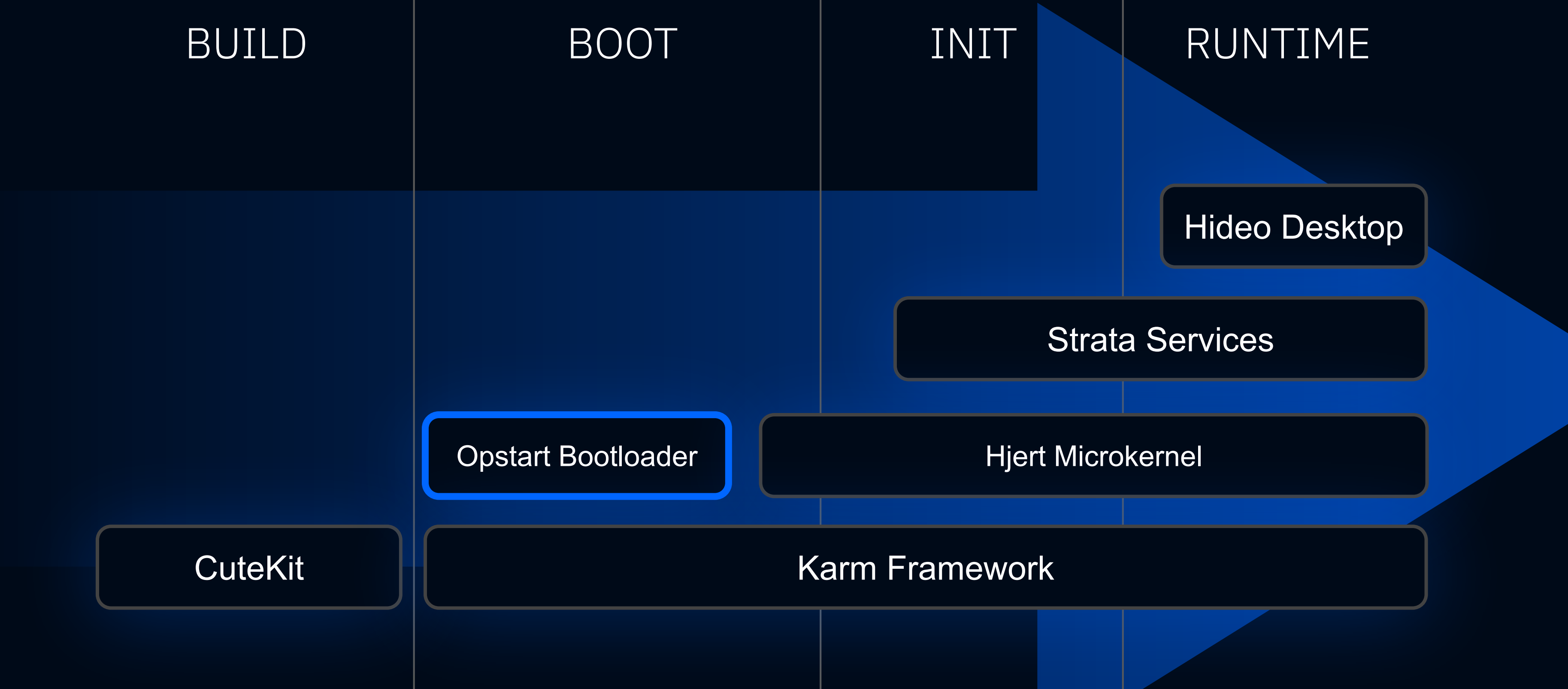
```
// Definition
export struct WindowCreate {
   using Response = Tuple<WindowId, WindowProps>;
   WindowProps want; };

// Usage
auto client = co_try$(
    co_await Sys::IpcClient::connectAsync("ipc:strata-shell"));
```

No exception!

```
auto [windowId, windowProps] = co_try$(
    co_await client.callAsync<IShell::WindowCreate>(
        {{800, 600}, App::FormFactor::NORMAL}),
cancellationToken);
```

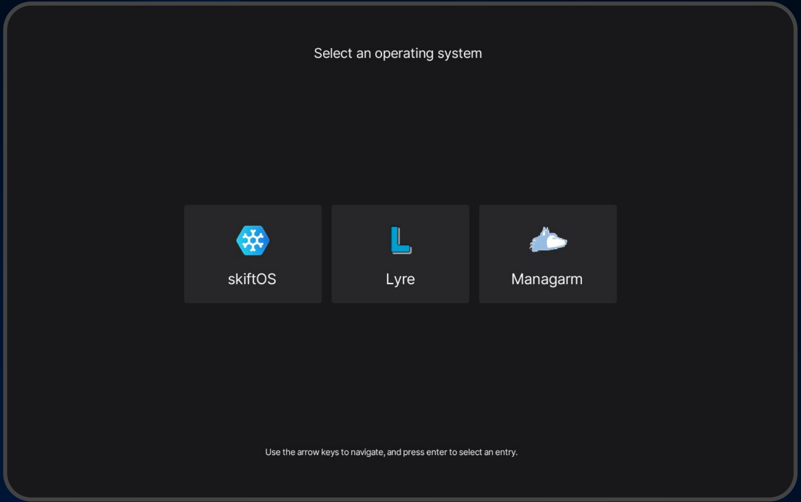
ARCHITECTURE



OPSTART

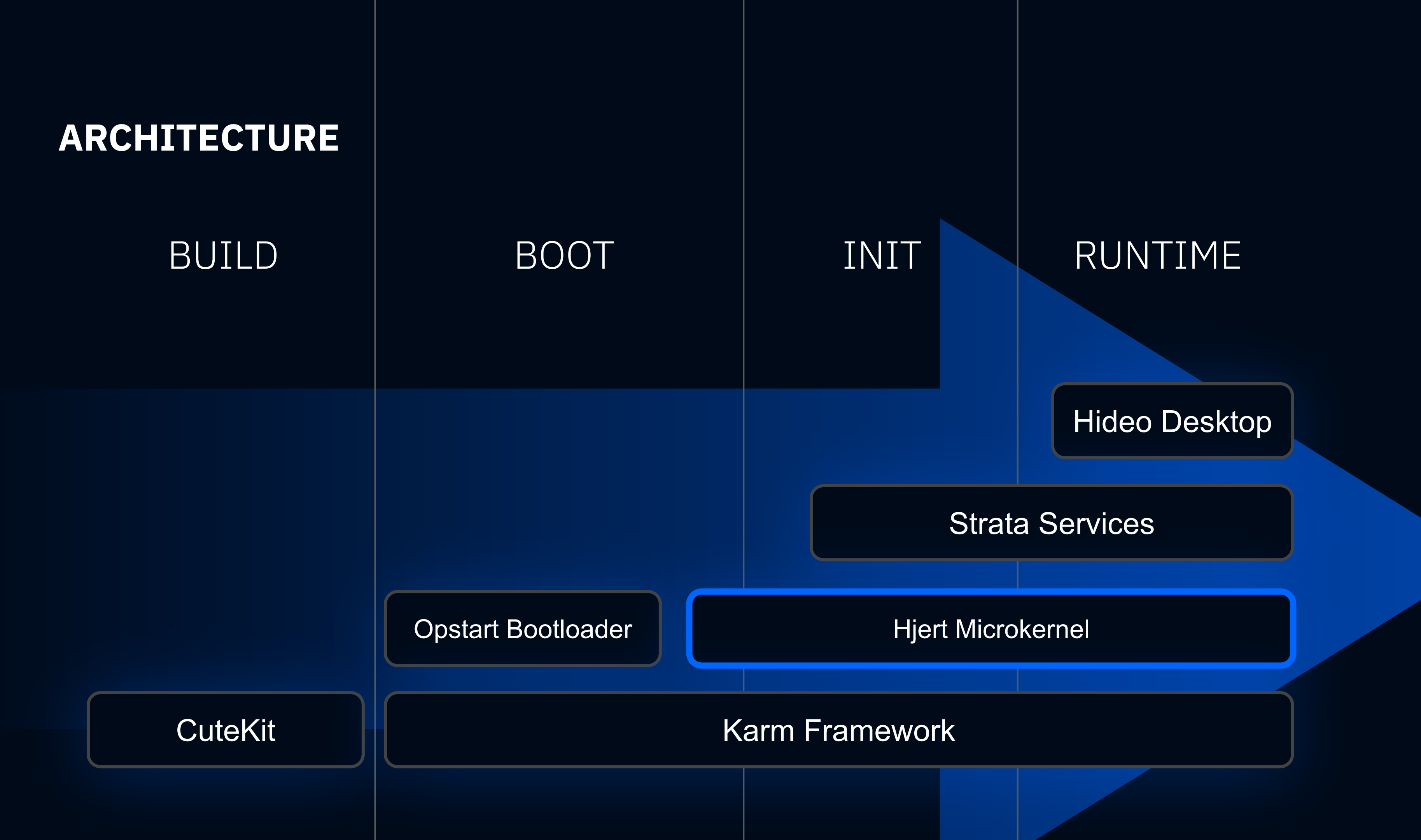
EFI bootloader

Boot protocol called handover



Use the arrow keys to navigate, and press enter to select an entry.

ARCHITECTURE



HJERT

“Pragmatic” microkernel design

~4k lines of code, written for clarity

Core kernel responsibilities

IPC, Preemption, Memory management, IRQ dispatch

Minimal object model

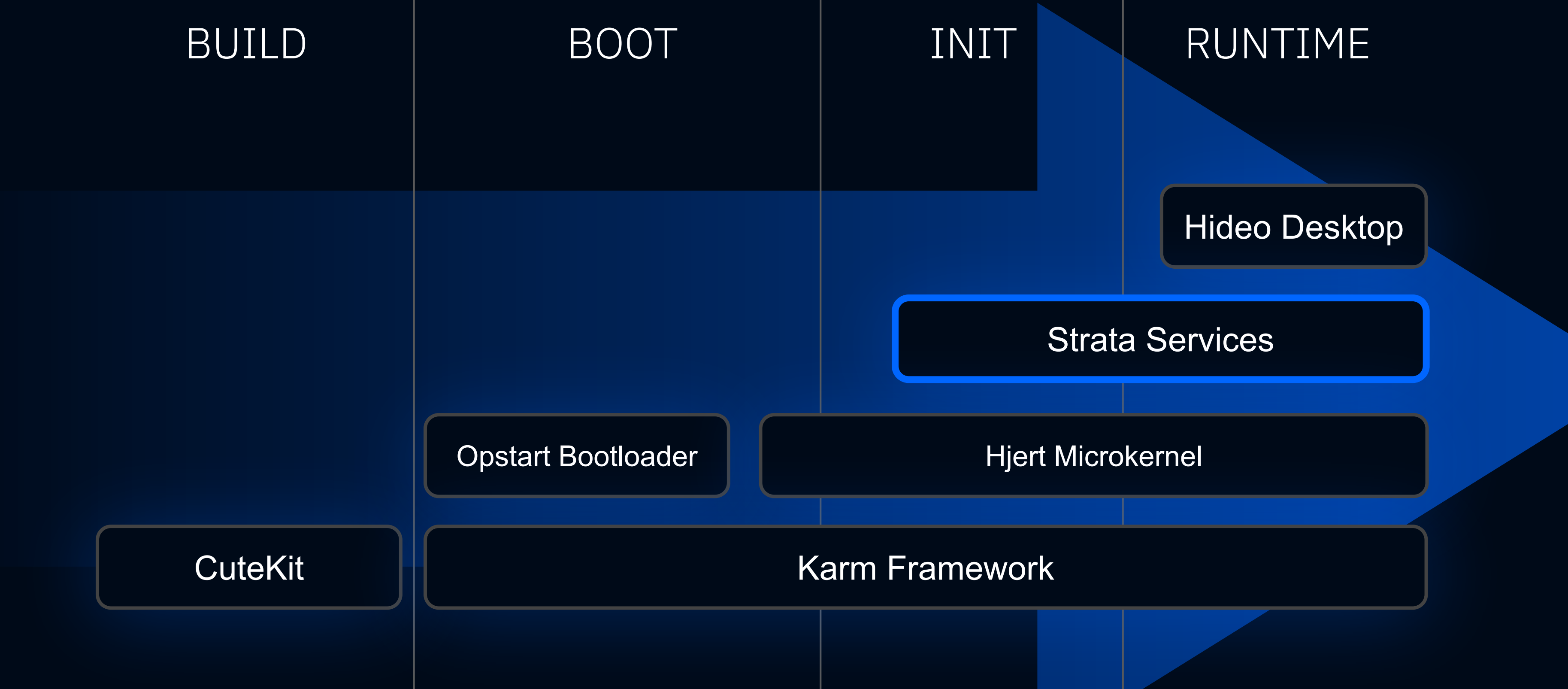
Task, Space, Vmo, Channel, Irq, Iop, Listener, ...

~25 syscalls total

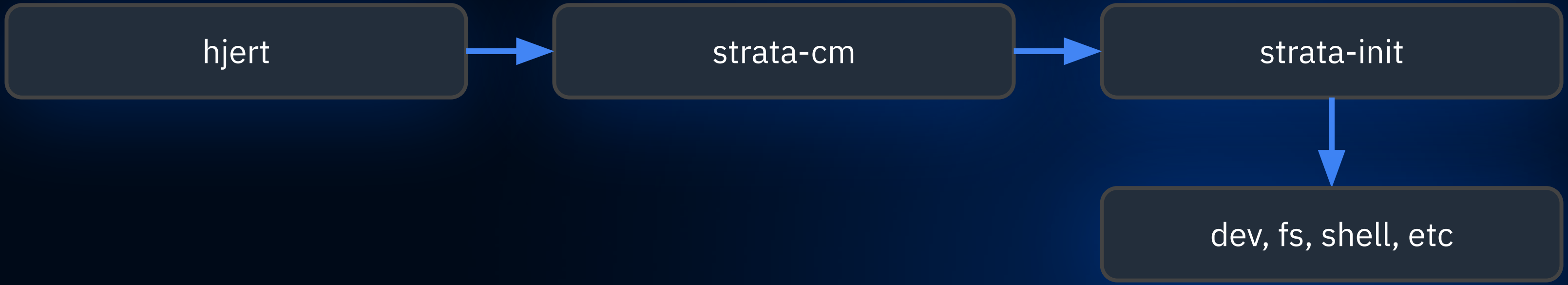
HJERT SYSCALL

```
Res<> doSend(Task& self, Hj::Cap cap, UserSlice<Bytes> buf, UserSlice<Slice<Hj::
caps) {
    return with(
        self.space(),
        [&](Bytes buf, Slice<Hj::Cap> caps) -> Res<> {
            auto obj = try$(self.domain().get<Channel>(cap));
            try$(obj->send(self.domain(), buf, caps));
            return Ok();
        },
        buf, caps
    );
}
```

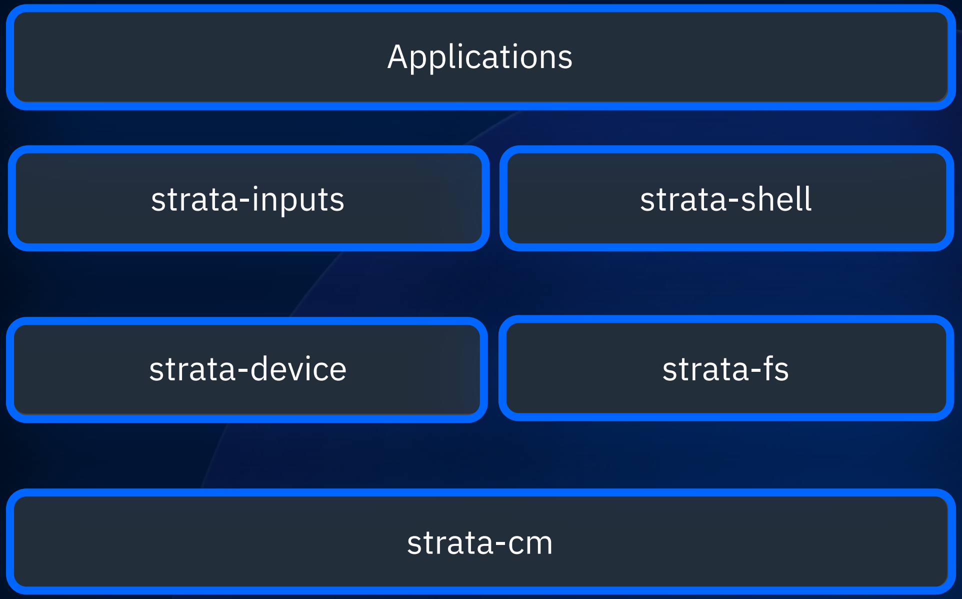
ARCHITECTURE



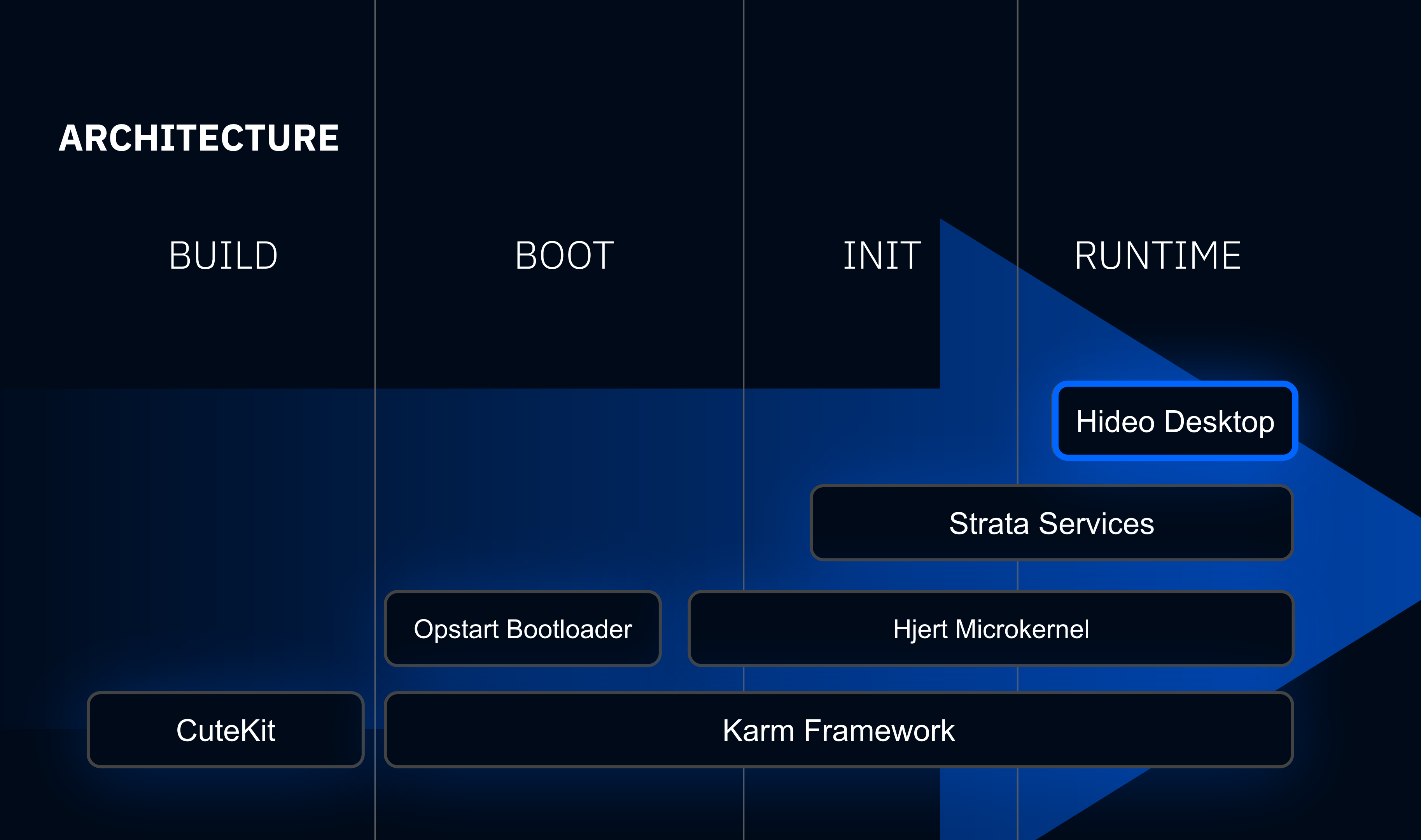
STARTING STRATA



STRATA


Applications
strata-inputs
strata-shell
strata-device
strata-fs
strata-cm

ARCHITECTURE



HIDEO

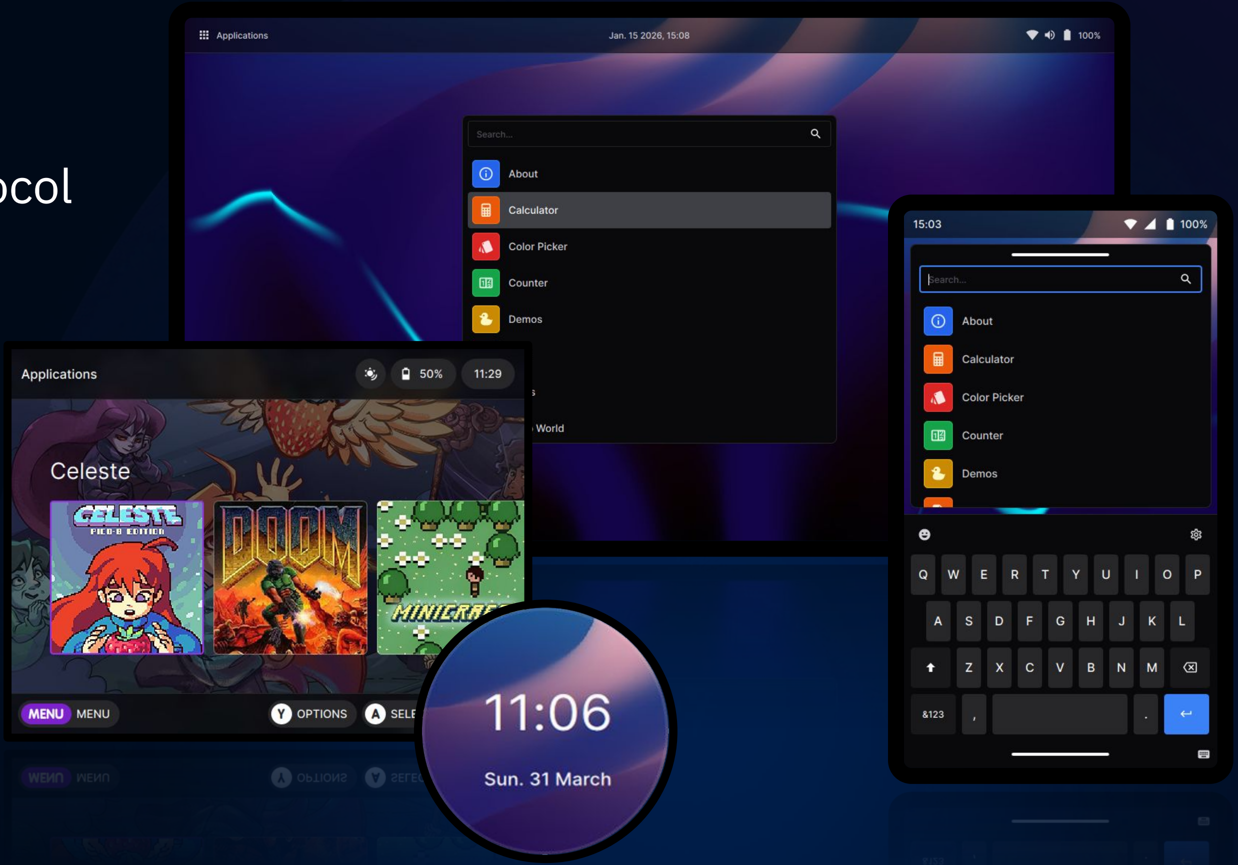
Custom display protocol

Adaptive UI

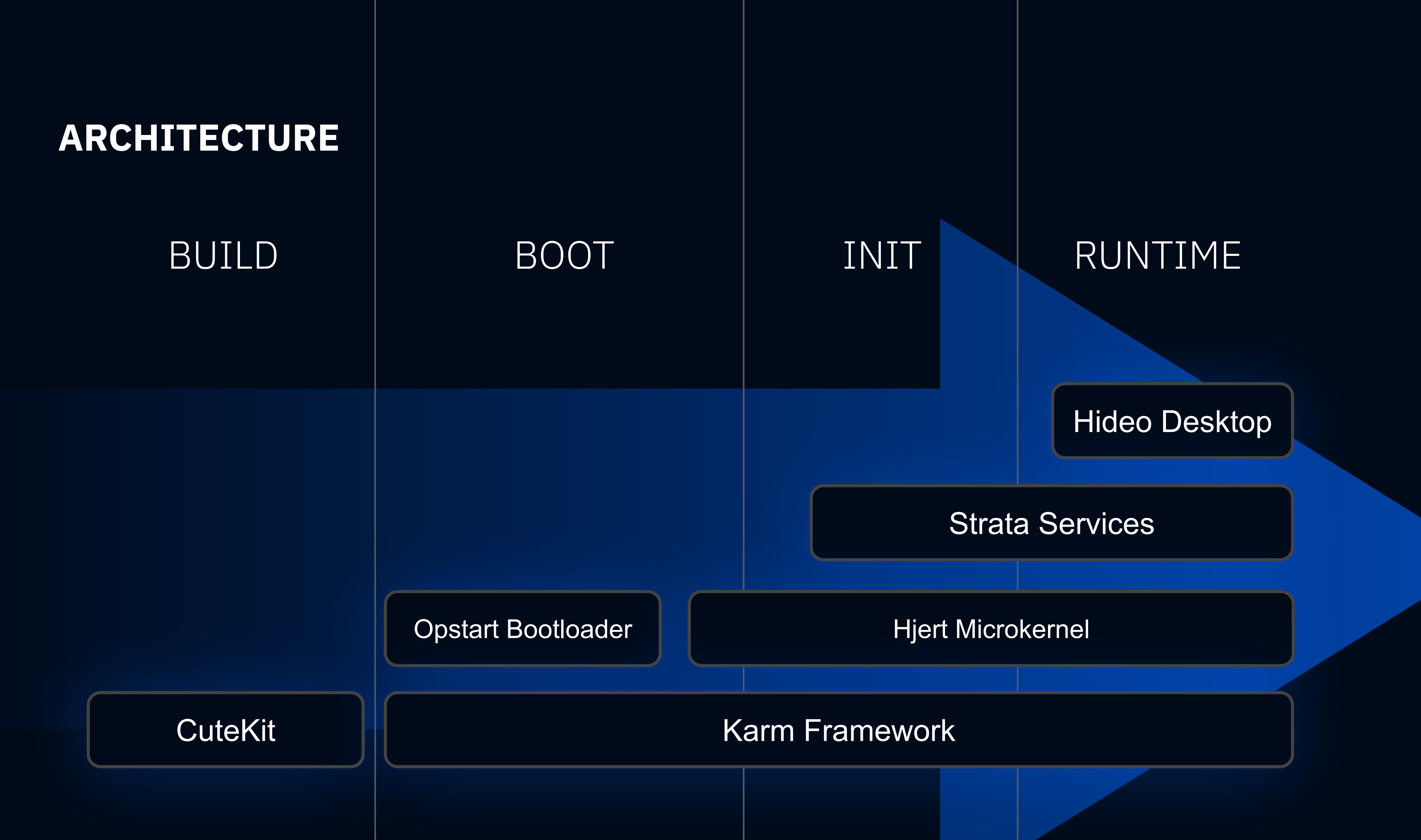
Vector Based

Subpixel antialiasing

TrueType



ARCHITECTURE



ARCHITECTURE

BROWSER

HTML/CSS only

No JS (for now)

Pretty fast

DEMO TIME!

BEYOND

Full virtio support

Networking

On disk file system

Sound Server

Namespacing

HTML/CSS in the bootloader :^)

ACKNOWLEDGEMENTS

Mathilde

Lou, Jordan, and All of DEVSE, and OSDEV

All of you

skift OS

Slides and links

RPC CANCELATION

$$\begin{tikzcd}
\boxed{\text{App}} \arrow[r] & \boxed{\text{Fs}} \arrow[r] & \boxed{\text{Dev}}
\end{tikzcd}$$

PACKAGING

All code and assets live in /bundles/<package>

Bundle contents are private by default

Only /bundles/<package>/public is exposed for consumption

Bundles are packaged into a single BootFS volume

BootFS is page-aligned and intentionally simple

Design inspired by Fuchsia