

# BUZZING ACROSS SPACE

THE ILLUSTRATED CHILDREN'S GUIDE TO EBPF





BROUGHT TO YOU BY...

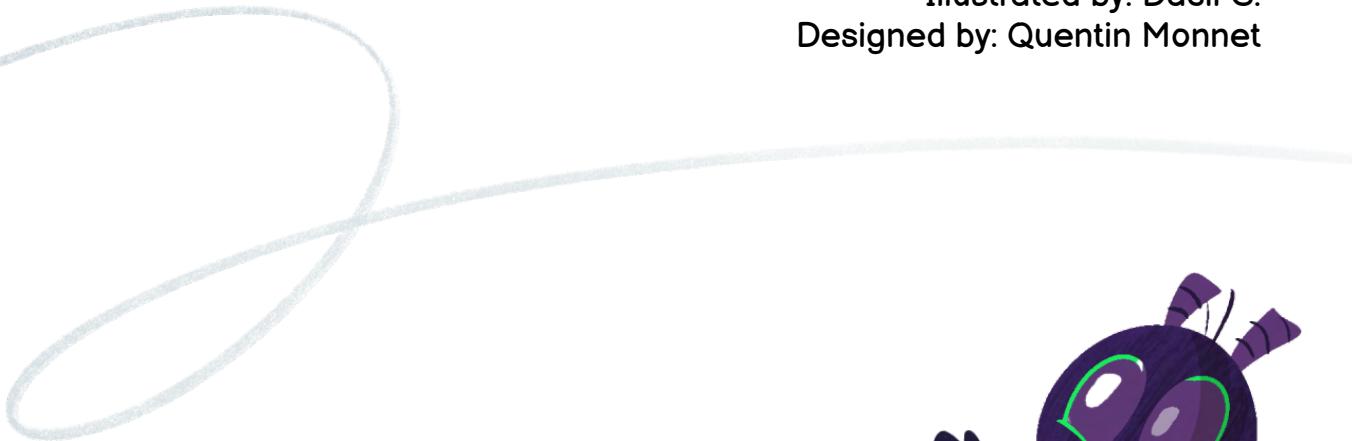
# ISOVALENT

Creators of  ciliium and  eBPF

Written by: Quentin Monnet & Bill Mulligan

Illustrated by: Dacil C.

Designed by: Quentin Monnet



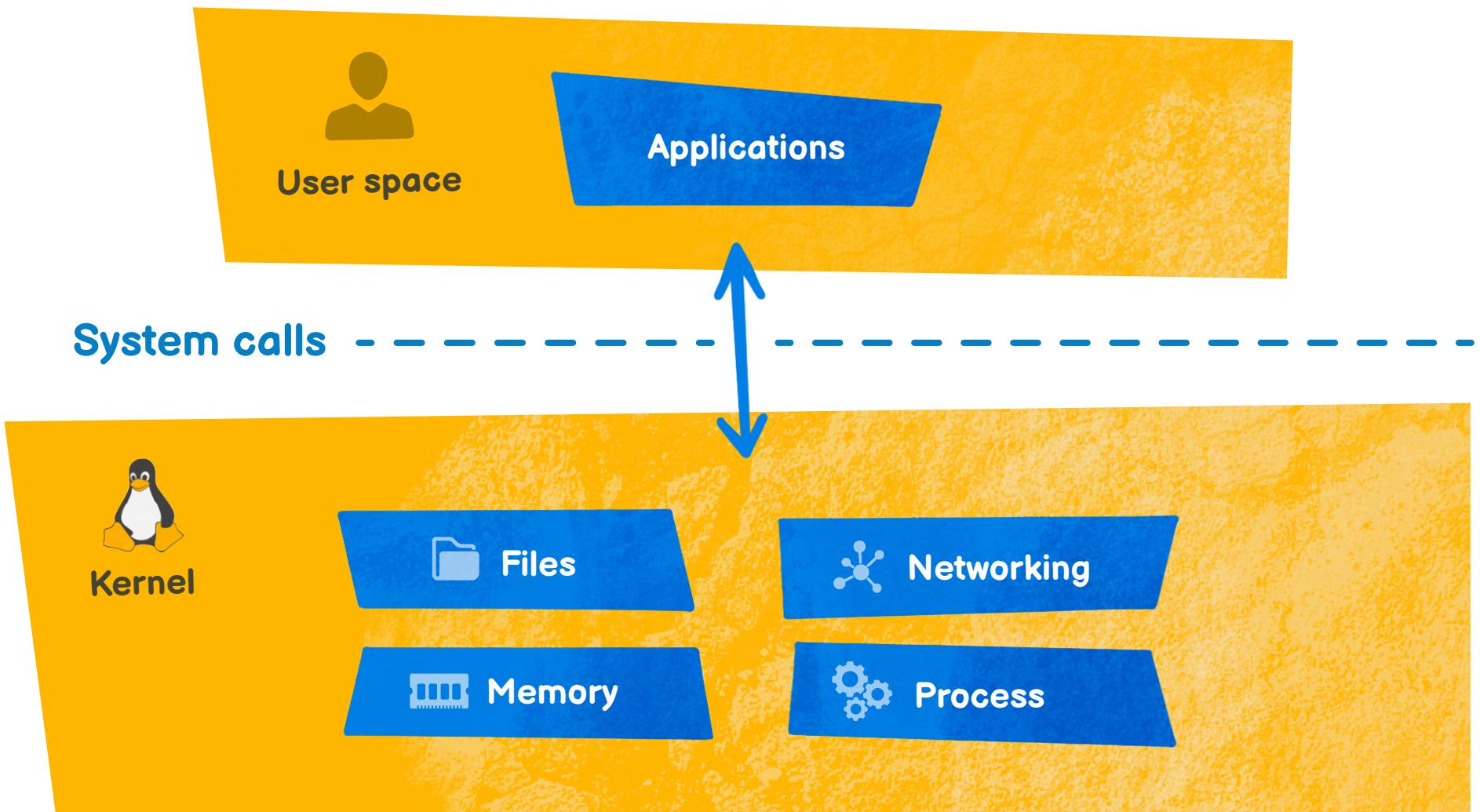
Here's a story about starships, about space travels, and about little bees who helped run the biggest vessels and benefited their entire community.

Even the smallest and youngest have a role to play. This is why education for all, and particularly for children, is essential! Whether you're young or old, you should learn some bits of knowledge about eBPF in this book. Hold on tight, we're taking off!



Far away, beyond skies, asteroids, and dust,  
The Silver Lining flew between the nebulae.  
On the front deck, at the commands, was Captain  
Tux, Who carried passengers across the galaxy.

While the crew, hard at work, assisted their captain,  
All the travelers enjoyed comfort, peace, and leisure.  
But forbidden to all was the cramped engine room,  
Its critical systems of metal and fire.

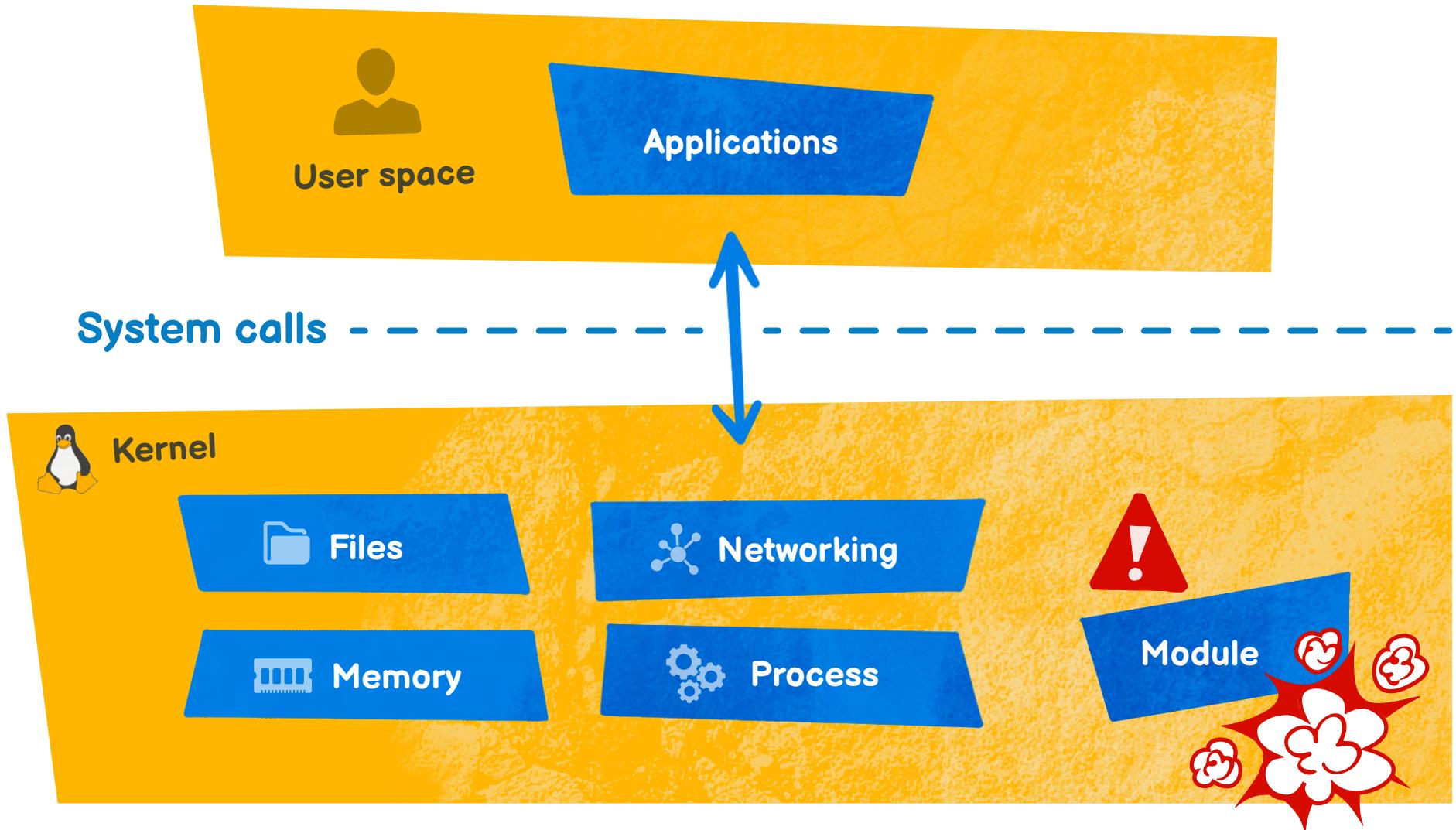


eBPF originates from Linux, an operating system that runs on billions of devices around the world and is divided into user space (where most applications run) and kernel space (which provides an interface for applications to interact with the underlying hardware). The kernel has visibility across the entire system and is highly performant, but needs to provide a stable interface to applications, so it lacks the flexibility of user space programming.



Flying for years across the galaxy and back,  
The crew learned to modify their ship and adjust.  
They changed the thrusters, the force shields, the hyper-  
stack, To explore the most exotic worlds, and beyond.

But it's tough and discouraging to upgrade systems  
In the vacuum of space, or on an aquatic moon.  
Captain Tux needed a fast way to replace items,  
Adapt quickly to meet demand and make business boom.



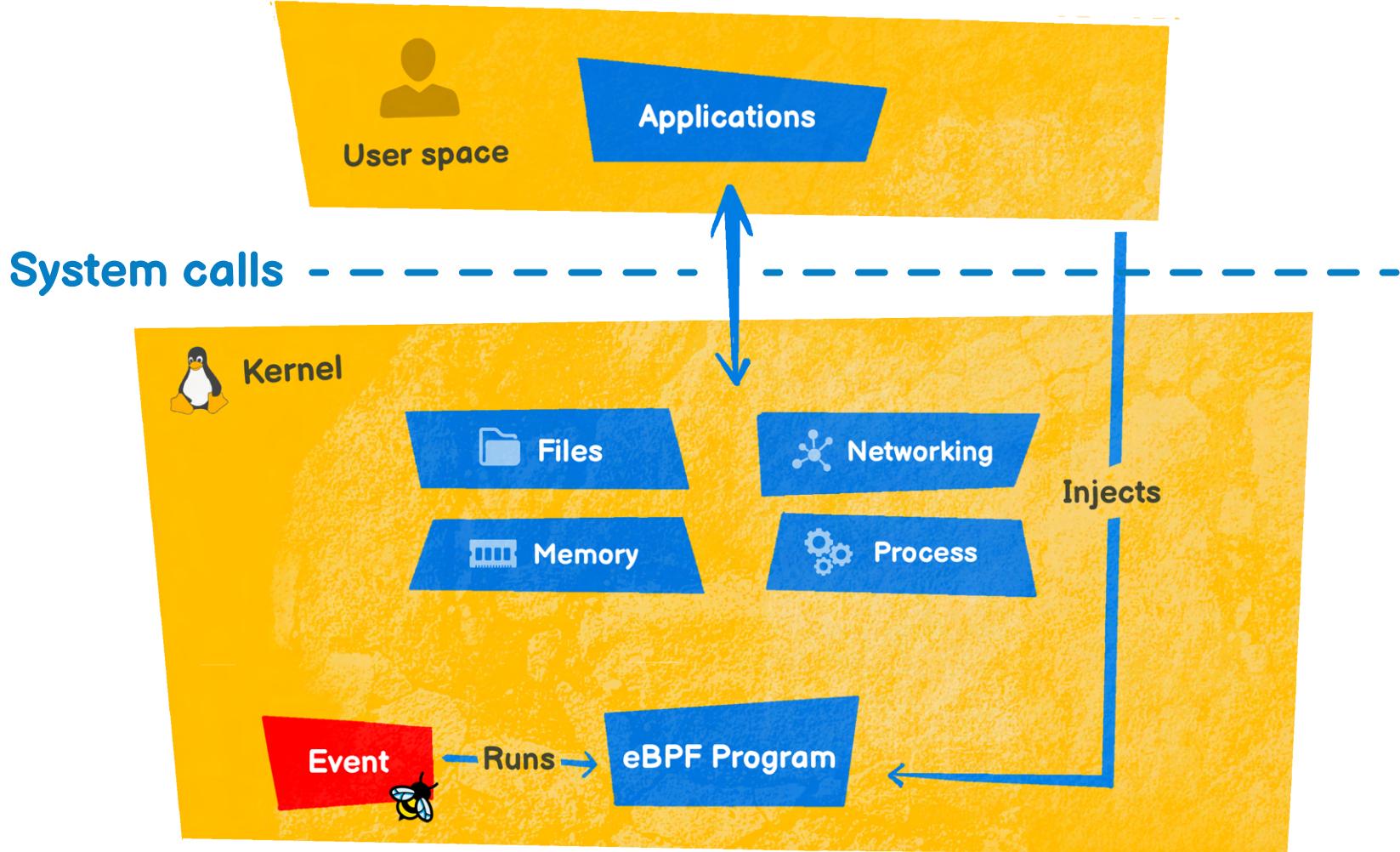
Because Linux is such a large and important project, updates to the kernel can take years to reach end users running stable long-term releases (LTS) in production.

It is possible to extend the kernel's functionalities by writing and loading kernel modules, but it comes with its own risks that they might crash or otherwise harm the kernel. If modules are not contributed upstream, they also need constant adjustments to adapt to the evolution of the kernel.



One day, a concerned Captain Tux reviewed the crew  
And remembered that bees had long been aboard,  
Working quietly, even if their chores were few. Captain  
Tux had a flash; they'd no longer get bored!

"You are small enough", he said to their spokesbee,  
"To fly straight to the engine room and work from the inside."  
"Would you like to assist?" "Of course", answered eBee,  
"We wish to help, to innovate, and we're on your side!"



eBPF comes from the original Berkeley Packet Filter (used in tcpdump), but now extends way beyond just networking, enabling users to programmatically extend almost any functionality of the operating system.

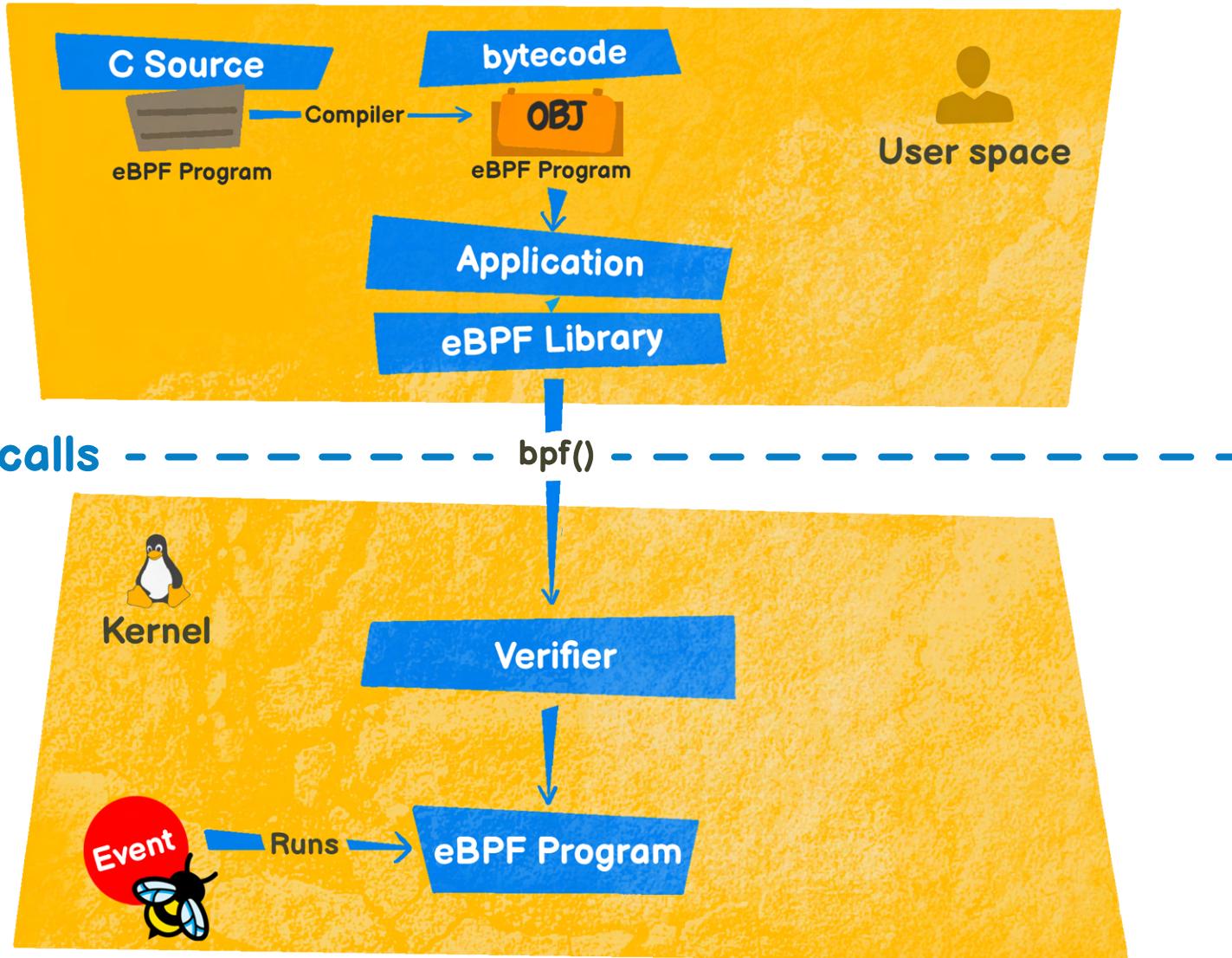
eBPF is an event-driven architecture that runs specific programs when the kernel or an application passes a certain hook point. For example, kernel probes (kprobe) or user probes (uprobe) can be added to attach eBPF tracing programs almost anywhere in kernel or user applications. eBPF is a strictly-typed assembly language with a stable instruction set.

eBPF programs can be loaded and upgraded in real time without the need to restart the kernel.

Bees of various talents took many roles in the hive.  
There were scientists, teachers, switchboard operators,  
Electricians, explorers, even a detective.  
They suited up and passed the airlock to the motors.

The engines were smokey and some bees were afraid  
To damage the engines or just to block a cog.  
“Fear not” said Captain Tux, “I will come to your aid”:  
He trained them and made sure they could see  
through the smog.





The Linux kernel expects eBPF programs to be loaded in the form of bytecode. Typically, eBPF developers write programs in C, Rust, or other languages, which are then compiled into eBPF bytecode. eBPF programs can be loaded into the Linux kernel using the `bpf()` system call, directly or through one of the available eBPF libraries.

The verifier runs in a privileged context and performs static analysis to ensure that eBPF programs are safe for the kernel, or sometimes the hardware, to run. It checks that the process loading the eBPF program holds the required capabilities (privileges), the program does not crash the system or leak sensitive memory, and that the program always runs to completion (will not sit in a loop forever, holding up further processing).

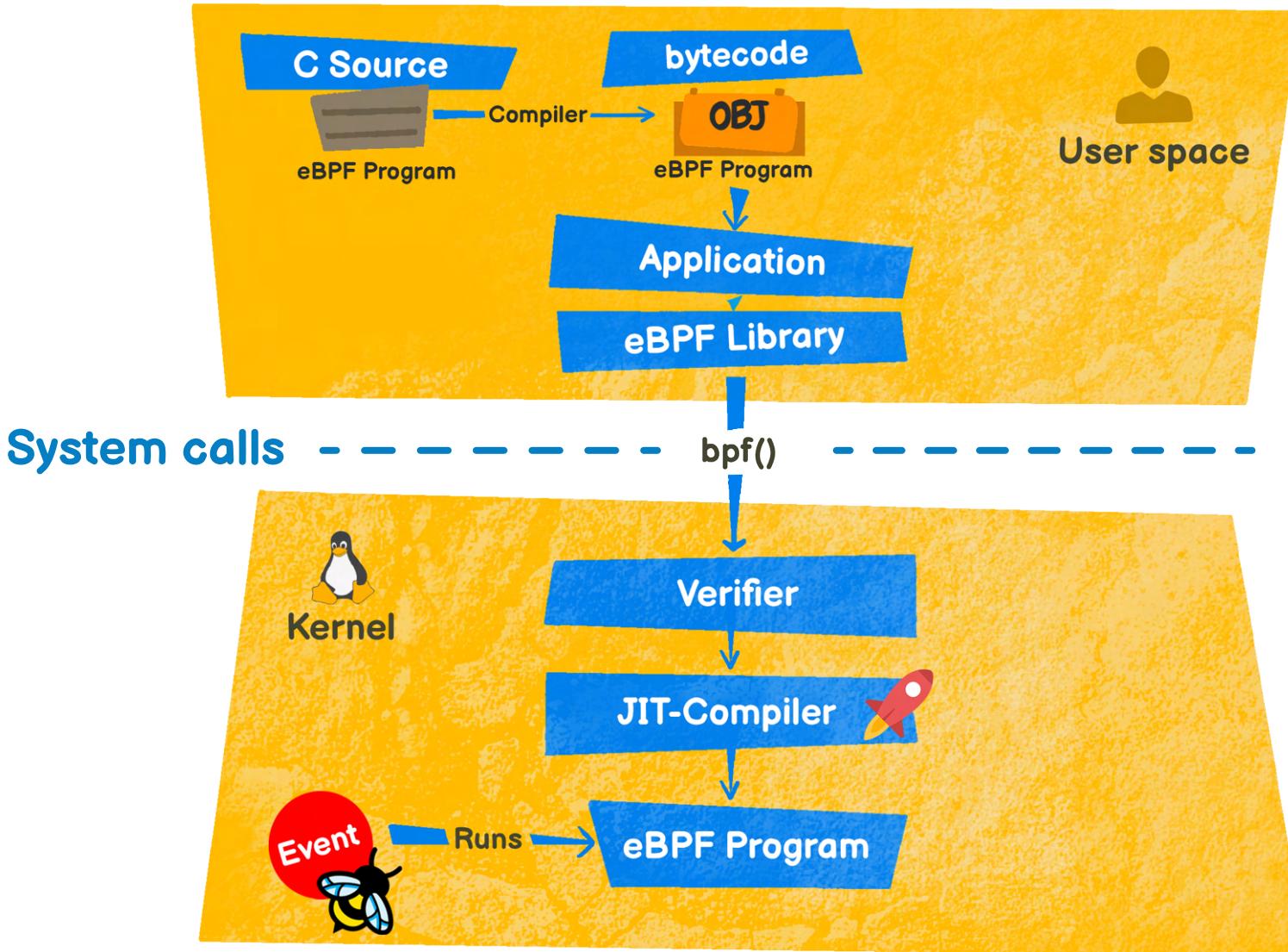
Captain Tux soon heard of possible improvement options.  
eBee's fellows were zealous bees, aiming for speed:  
"We'd go faster, for sure, with buzz code instructions."  
"This would be much simpler for us to parse and read."

With his laser screwdriver and a bit of copper,  
Captain Tux built a sturdy translator droid,  
And the buzzing bees worked swifter than ever.  
They organized as teams, and eBee took the lead.

SLOW →

01000110  
01100001  
01110011  
01110100



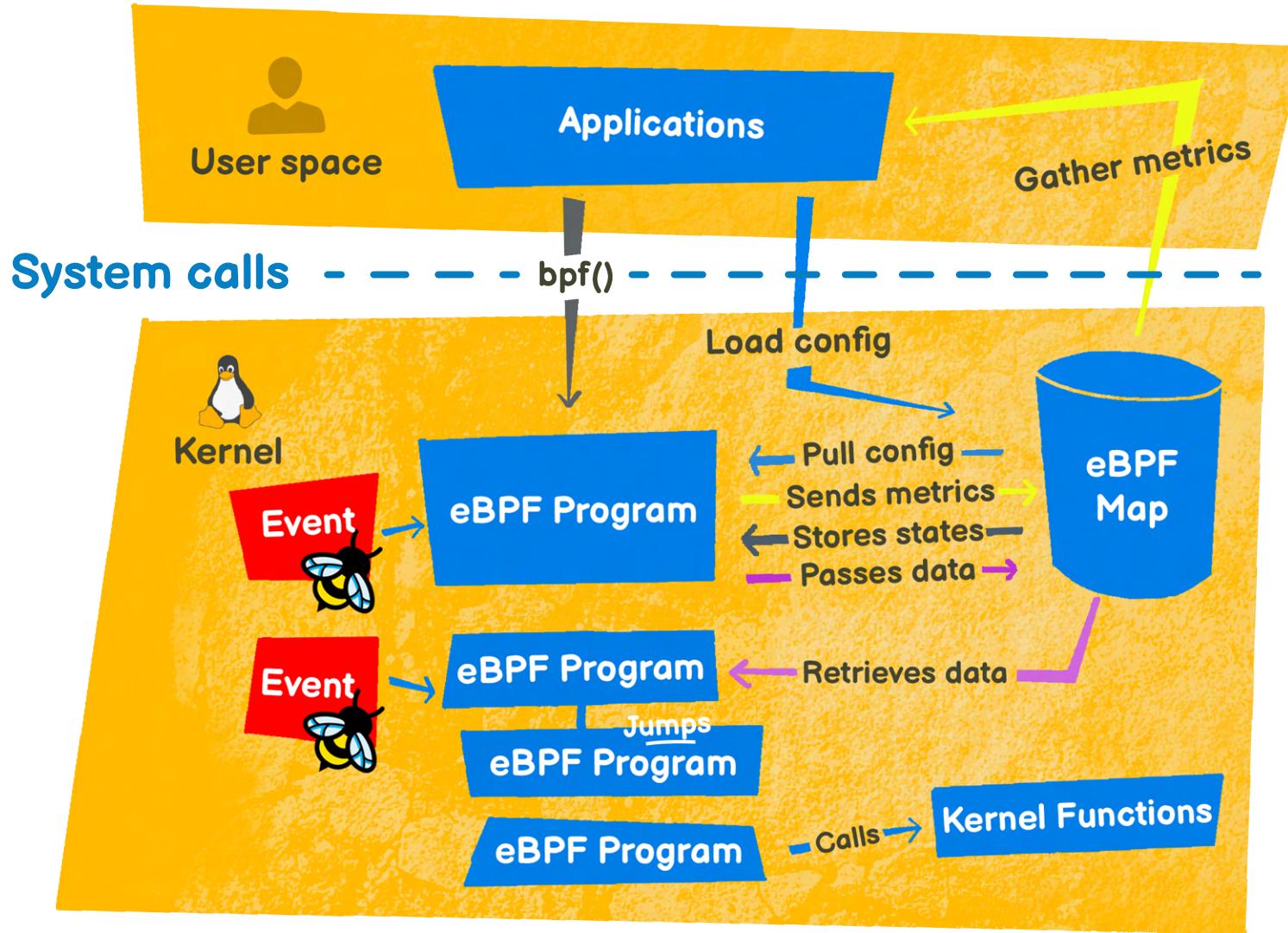


The Just-in-Time (JIT) compilation step translates the generic bytecode of the program into the machine-specific instruction set to optimize execution speed. This makes eBPF programs run as efficiently as natively compiled kernel code or as code loaded as a kernel module. Linux supports JIT compilation towards all widespread architectures.



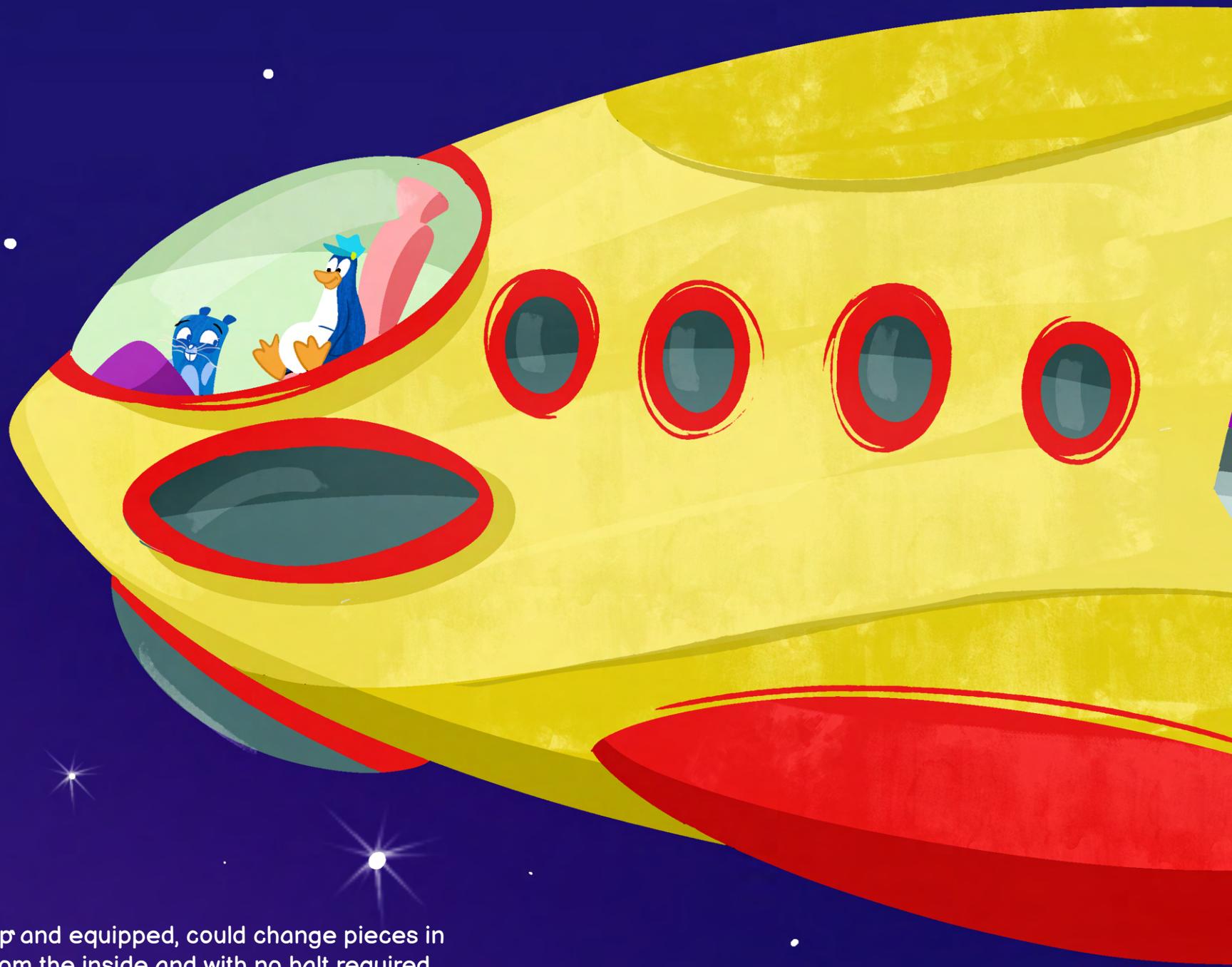
Acquiring a new taste for engine room hacking,  
The bees developed their activities more and more.  
In the narrow spaces, they began creating  
A real workshop, with shelves, tools, and gathered lore.

The shelves were great to store all sorts of materials,  
So that one bee could pass their product to the next.  
The tools would help reuse the engine's internals  
To unlock and harness the *Silver Lining*'s powers.



Users can leverage the following objects or mechanisms when programming with eBPF:

- Share collected information, retrieve configuration options, and store state through eBPF maps to save and retrieve data in a wide set of data structures. These maps can be accessed from eBPF programs as well as from applications in user space.
- eBPF programs can make function calls into a set of dedicated kernel functions (eBPF helpers/kfuncs) to help them accomplish some specific tasks.
- For more flexibility, eBPF programs are also composable with the concept of tail and function calls.
- Various mechanisms allow eBPF programs to contain loops, while ensuring the programs always terminate.



The bees, set up and equipped, could change pieces in flight,  
Directly from the inside and with no halt required.  
No more dangerous stops on hostile satellites!  
For the first time the crew could enjoy peace of mind.

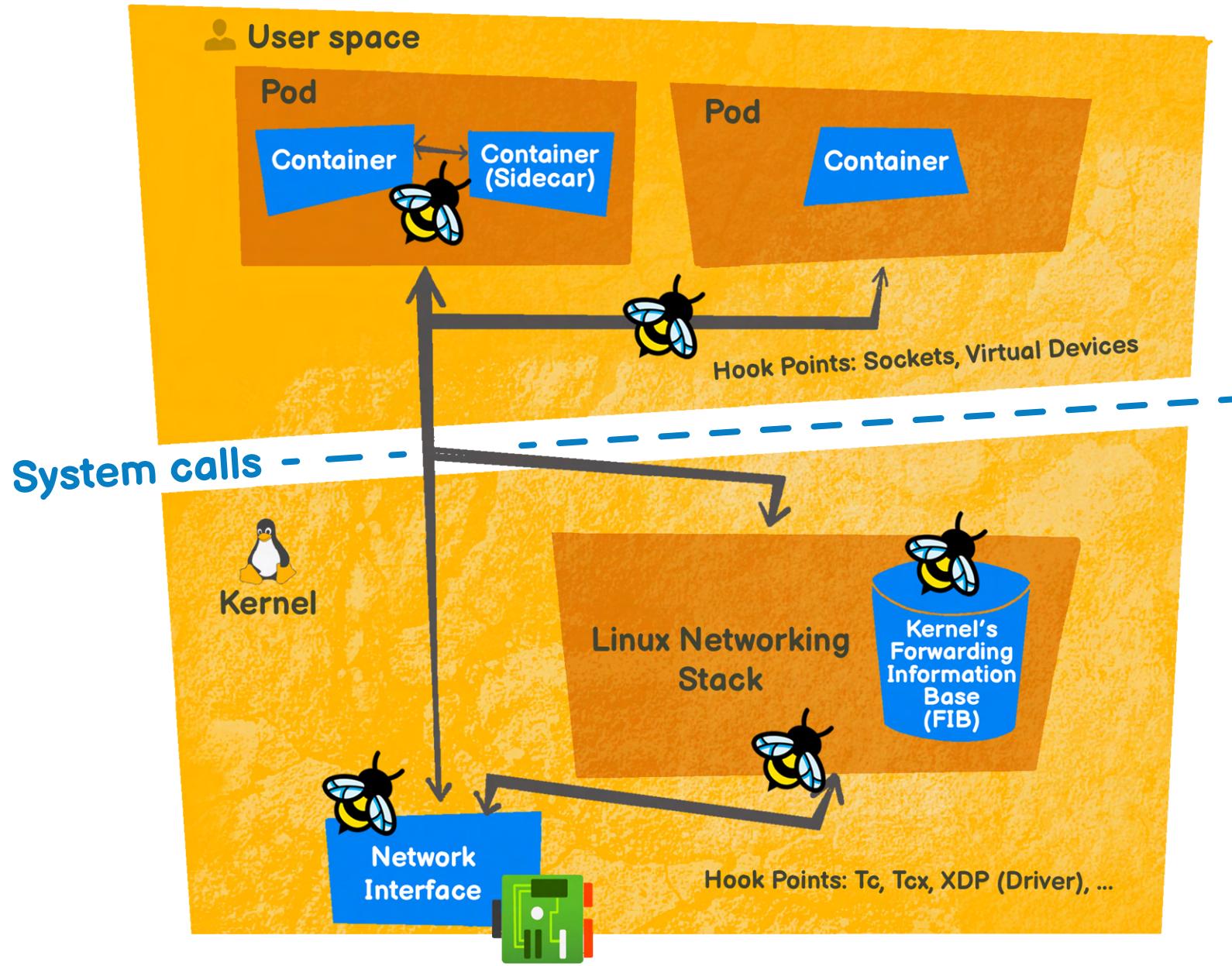


Once they had completed all pending upgrades, and then some  
The bees looked back at the fruits of their labor.  
“Let’s do more! Captain Tux is not the only one  
Who is in need of help. Let’s meet the passengers!”



The Silver Lining's passengers had long been complaining  
They couldn't send messages in their own encoding.  
When the bees jumped on the case, it marked the beginning  
Of a whole new era for data sharing and messaging.

Mail was still slow to go through the ship's processors,  
But the electrician bee had a great idea.  
And so the swarm replaced legacy receivers,  
They installed and rewired a boosted antenna.



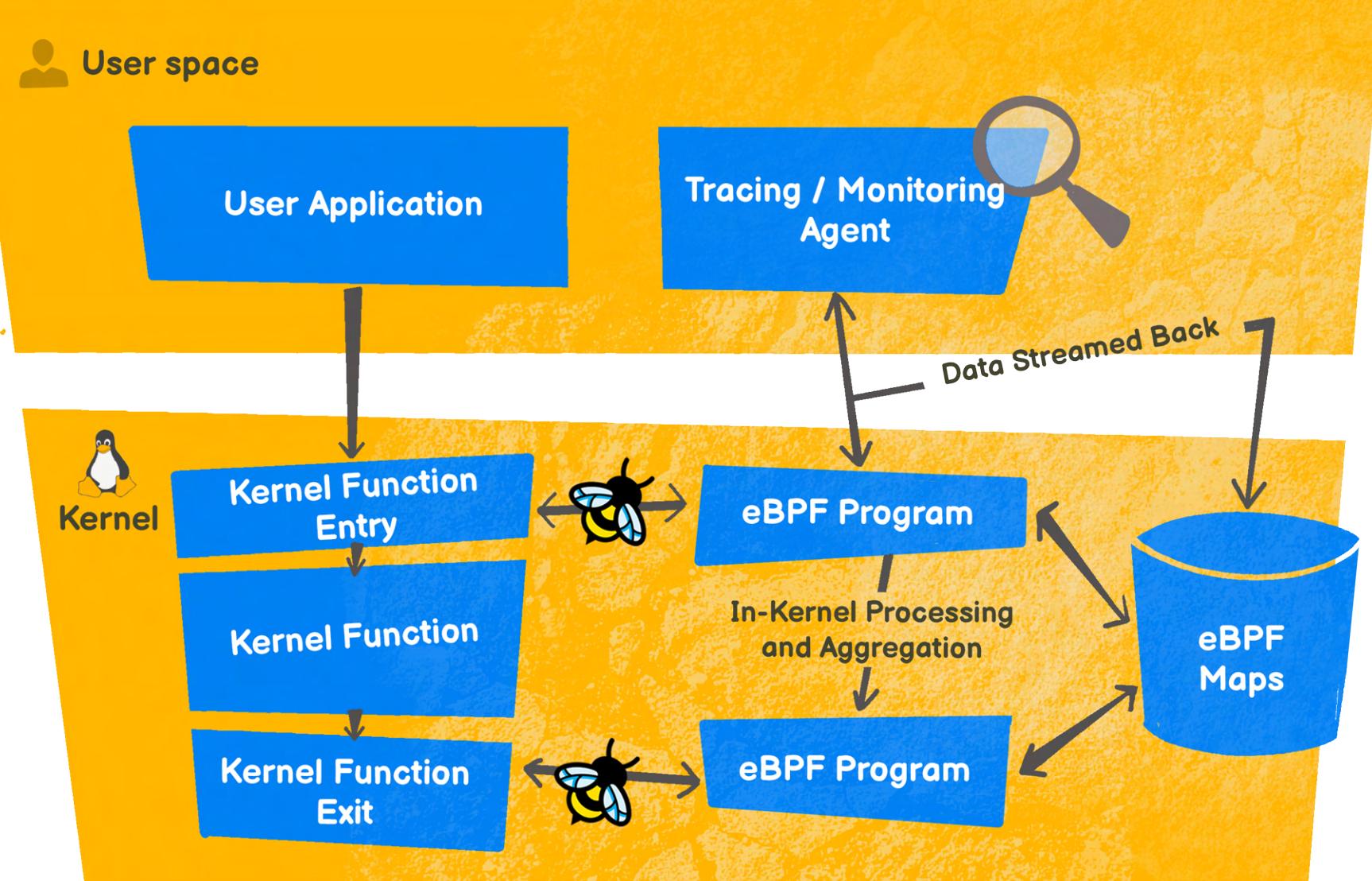
eBPF enhances networking by enabling efficient packet processing and filtering in the kernel decoupled from hardware-specific details while integrating seamlessly with the networking stack. This enhances network performance and flexibility making it ready for the cloud-native world.

Projects using eBPF for networking include Cilium and Katran, for example.



So many messages! At times, some would get lost,  
Or the engines would struggle under the pressure.  
So the detective bee added gauges and sensors  
To measure and observe, and keep things in order.

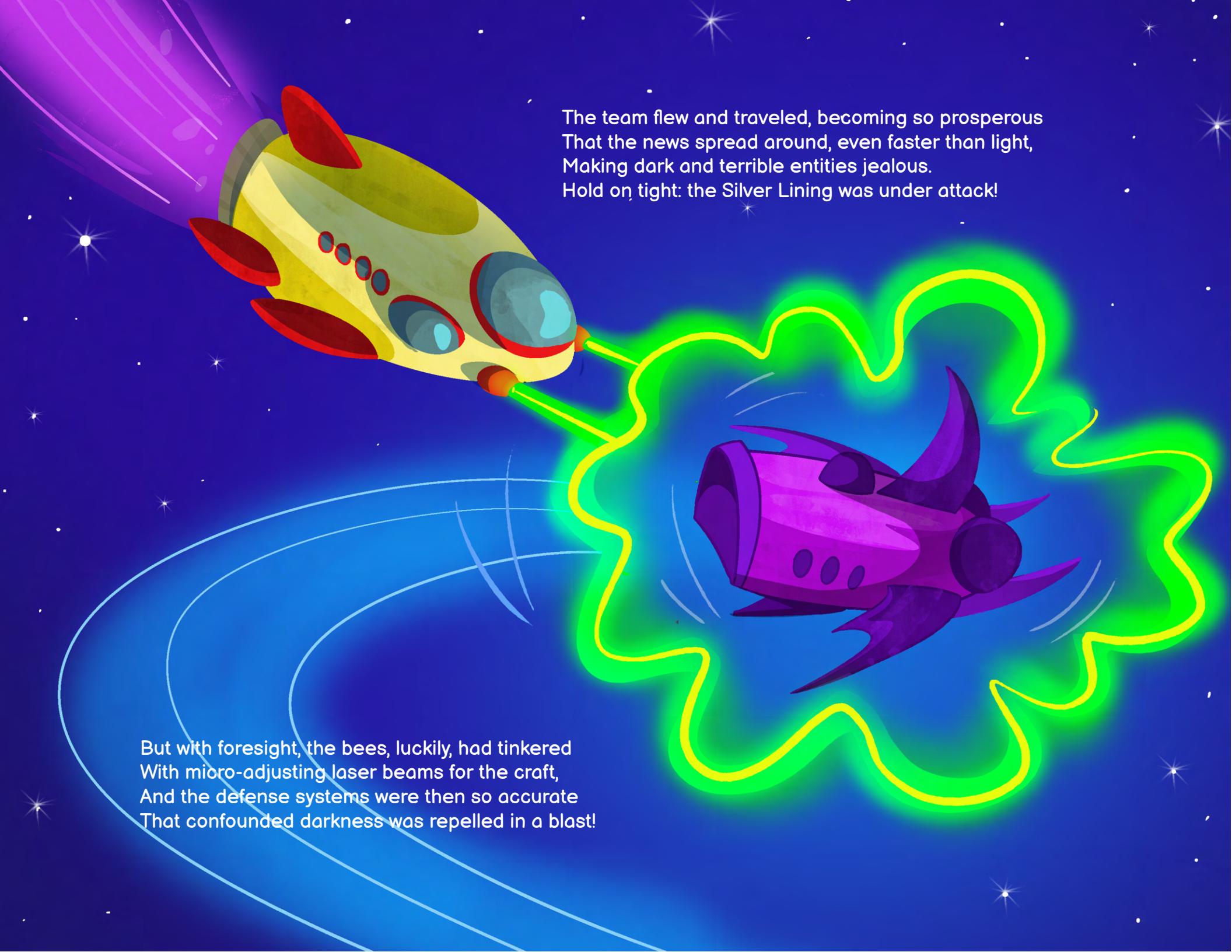
The busy engine room was the ideal place  
To monitor all things happening on the ship,  
And the bees optimized, process after process,  
All the things that they could, to improve every trip.



By running in the kernel, eBPF enables broad system observability. Collection and in-kernel aggregation of metrics allows flexible and efficient generation of observability events and data structures from a wide range of possible sources without having to export samples.

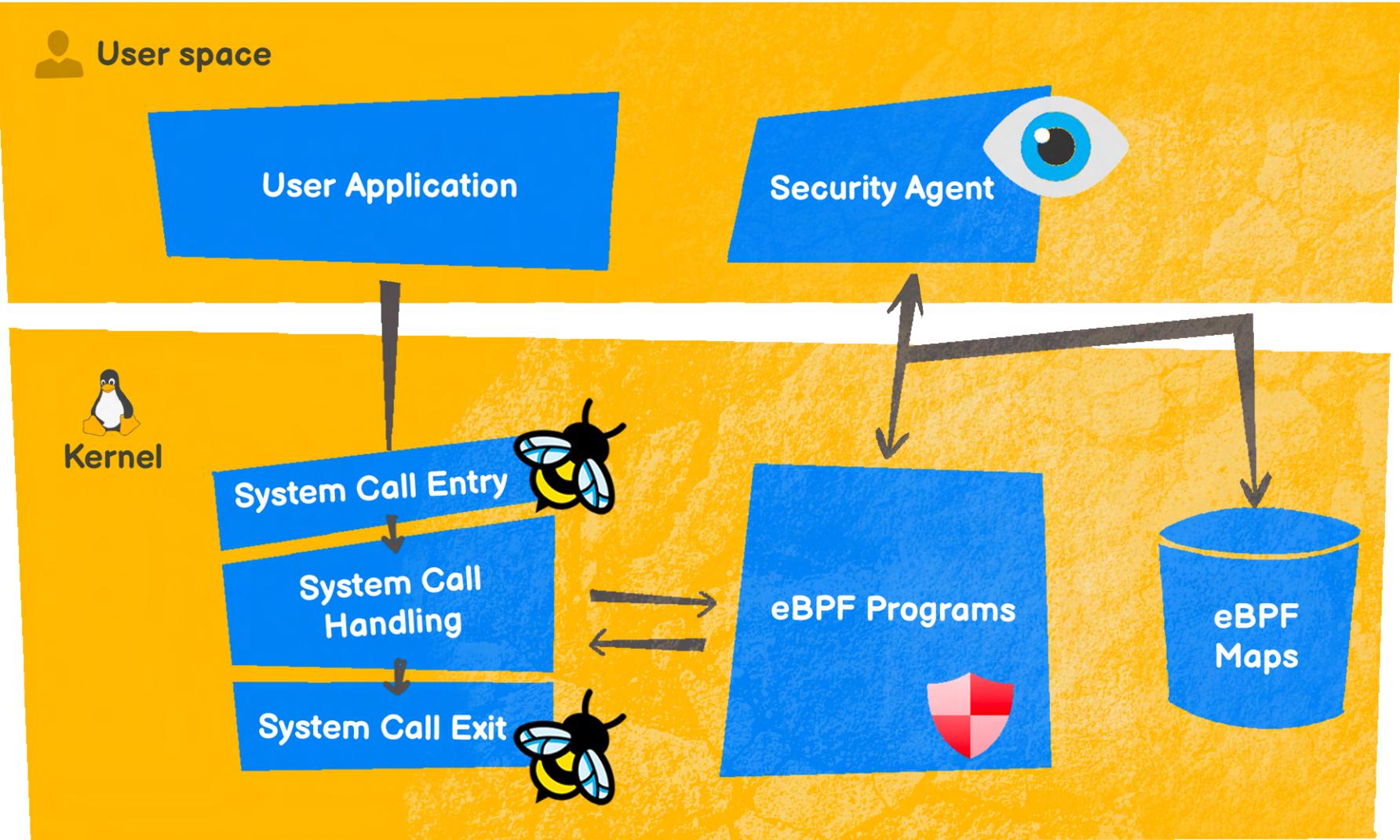
Attaching eBPF programs to trace points as well as kernel and user application probe points gives powerful introspection abilities for the kernel and user space applications and useful insights to troubleshoot system performance problems.

Projects such as BCC, bpftrace, Hubble, Parca, or Pyroscope use eBPF for tracing and monitoring purposes.



The team flew and traveled, becoming so prosperous  
That the news spread around, even faster than light,  
Making dark and terrible entities jealous.  
Hold on tight: the Silver Lining was under attack!

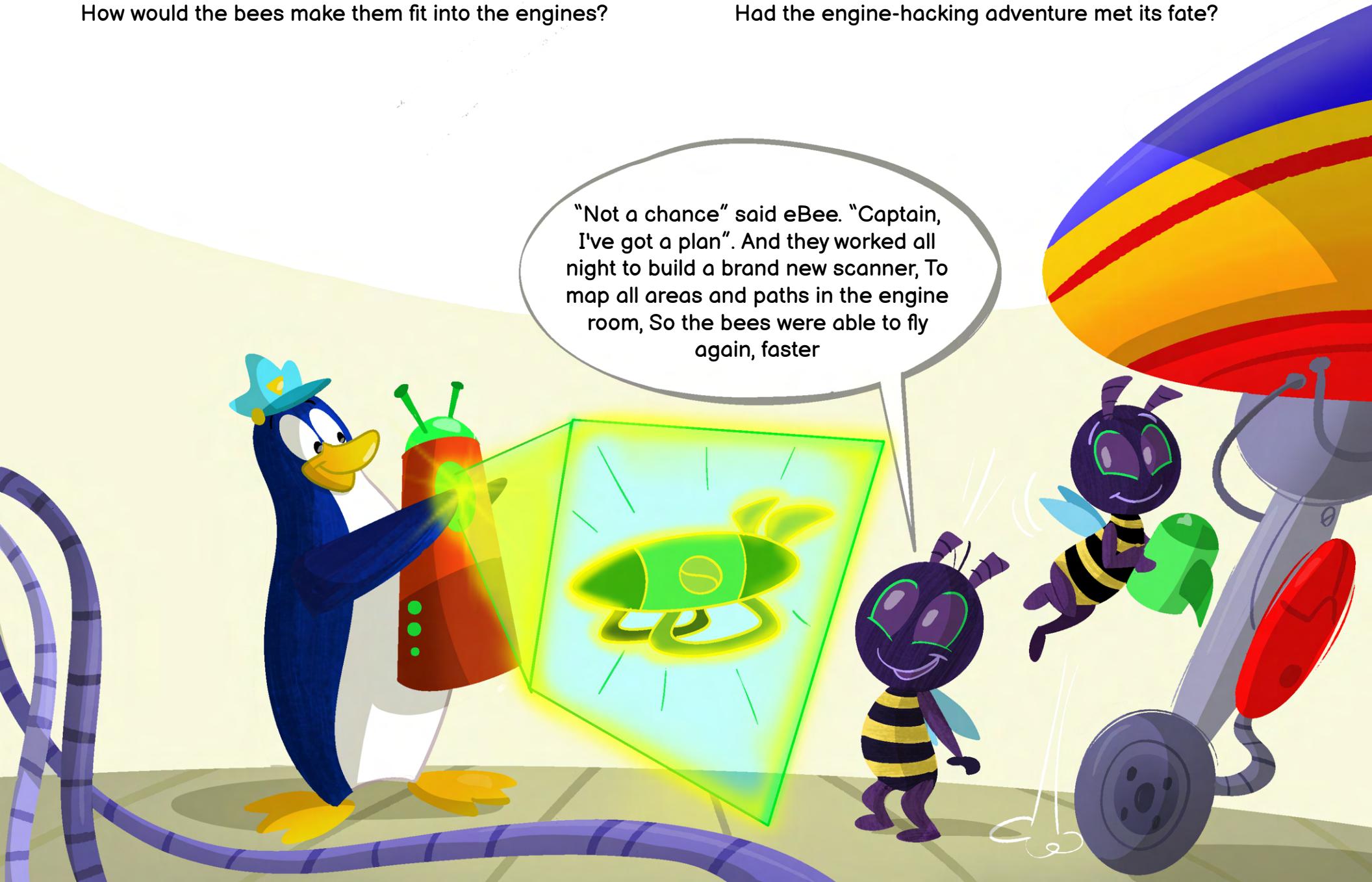
But with foresight, the bees, luckily, had tinkered  
With micro-adjusting laser beams for the craft,  
And the defense systems were then so accurate  
That confounded darkness was repelled in a blast!

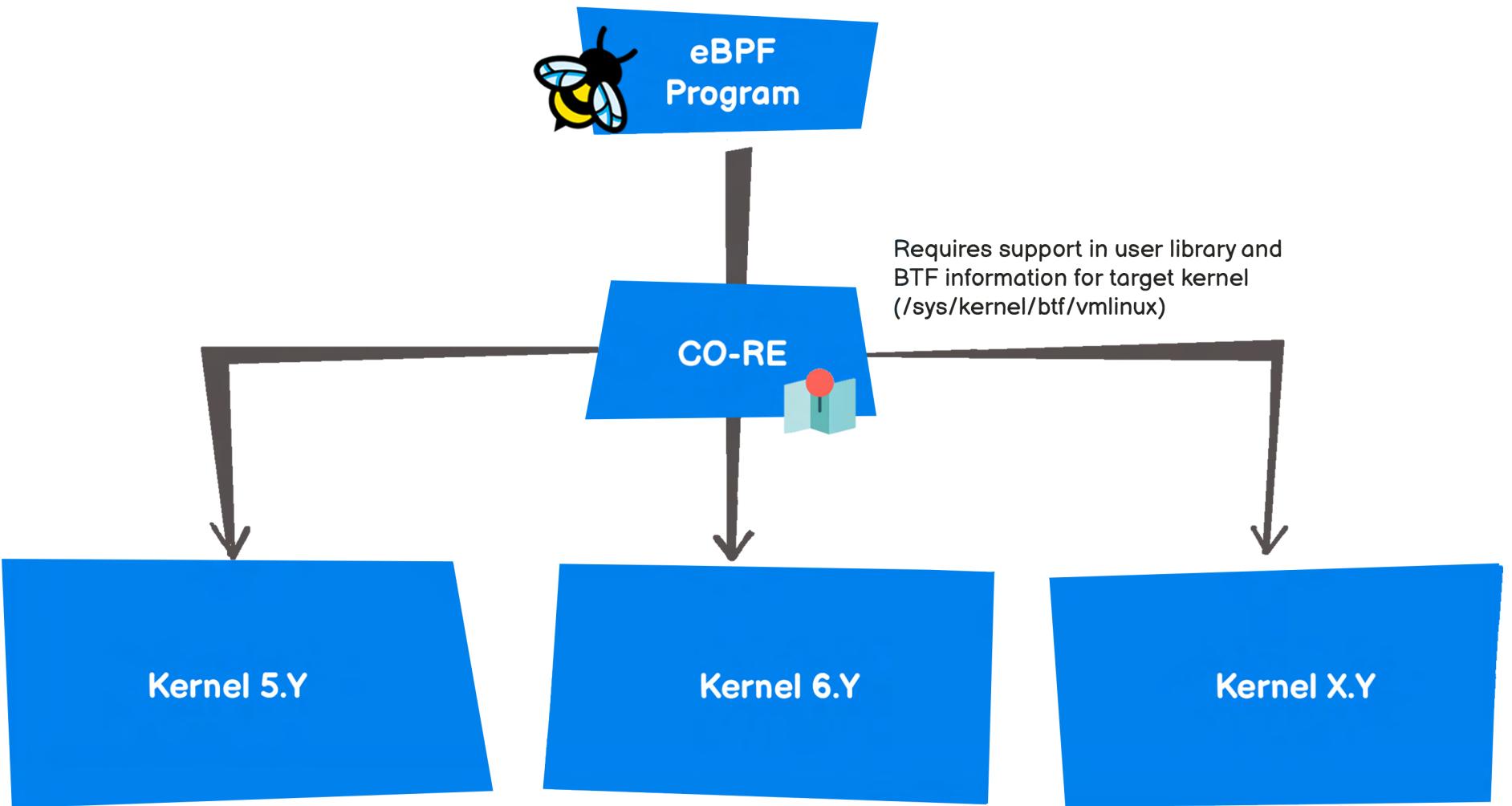


eBPF can be used in conjunction with Linux Security Modules (LSM) to allow runtime instrumentation of the LSM hooks. eBPF combines seeing and understanding all system calls with a packet and socket-level view of all networking. This creates security systems operating with richer context and a better level of control than traditional solutions. Projects using eBPF for security purposes include Falco, Tetragon, or Tracee.

Alas! During the fight a thruster was damaged.  
Once at the space garage, Captain Tux found spare pieces,  
But they lacked instructions, and the model had changed.  
How would the bees make them fit into the engines?

And then what about the quantic guidance units?  
New versions all had a completely different shape.  
eBee's friends couldn't fly between the narrower bits.  
Had the engine-hacking adventure met its fate?





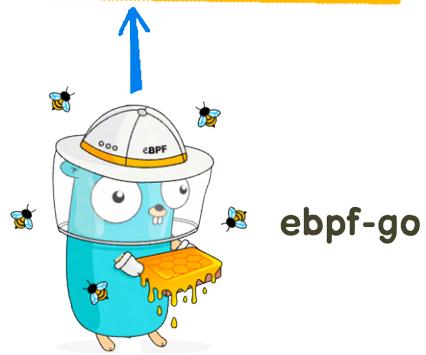
Kernel types and data structures are in constant flux. eBPF "Compile Once – Run Everywhere" (CO-RE) leverages BPF Type Format (BTF) to make programs portable across kernel versions, especially if they rely on data structures internal to the system.



Before the ship flew off, intrigued by the rumors,  
Several creatures came to meet the spokesbee.  
They wondered if the bees would like to join efforts  
And collaborate on craft, speeders, and ferries.

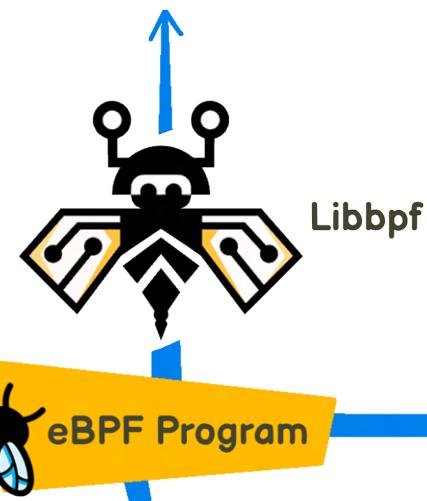
eBee was delighted and agreed that the hive  
Would split, some bees going to new ships and new teams.  
And the word spread further of transports that would thrive  
With bees under the hood, making them go full steam.

## Golang Application



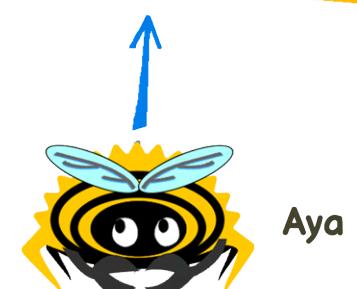
ebpf-go

## C/C++ Application

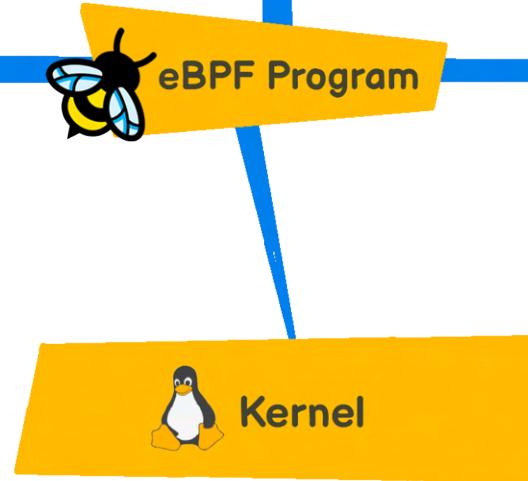


Libbpf

## Rust Application



Aya



eBPF now has a variety of libraries written in Golang, Rust, C++, and others that help loading, compiling, and debugging eBPF programs across both user and kernel space.

eBPF is at the heart of many projects. See the full landscape at <https://ebpf.io>.



eBPF

As fame grew and bees spread, they felt the desire  
To keep contact and share news, knowledge, tips and tricks,  
To better tell around how their skills were for hire.  
And the united bees formed the Hive Alliance.

Under this new banner, bees now help innovate  
Teams on various vessels, keeping them fast and sound.  
Far away, beyond skies, asteroids, and dust,  
When he thinks of the bees, Captain Tux is so proud!

After years of proven production experience, eBPF has been adopted for Windows and other privileged execution contexts.

The eBPF Foundation now brings together a cross-platform community of eBPF-related projects. Part of the Foundation, the BPF Steering Committee is responsible for the technical direction and overall vision of eBPF and as of 2023, there is work in progress with the IETF (Internet Engineering Task Force) for standardizing eBPF.



## Credits

The eBPF logo was designed by Vadim Shchekoldin. It is copyright The Linux Foundation, on behalf of the eBPF Foundation.

See <https://ebpf.foundation/brand-guidelines/>

Other bees are based on designs by Vadim Shchekoldin. They are copyright Isovalent, and licensed under the Creative Commons Attribution-NonCommercial 4.0 (CC-BY-NC-4.0). See <https://github.com/isovalent/ebeedex>

Captain Tux is based on Tux (the Linux mascot) designed by Larry Ewing <[lewing@isc.tamu.edu](mailto:lewing@isc.tamu.edu)> using The GIMP

The gophers are based on the Go Gopher designed by Renee French. They are licensed under Creative Commons Attribution 3.0 (CC-BY-3.0).

Cappy (the turtle), Captain Kube (the owl), Hazel (the hedgehog), Phippy (the giraffe), and Tai (the elephant) are copyright The Linux Foundation, on behalf of the Cloud Native Computing Foundation. They are licensed under Creative Commons Attribution 4.0 International (CC-BY-4.0). See <https://phippy.io>

The crab is based on Ferris, the unofficial Rust mascot, designed by Karen Rustad Tölva. It is licensed under Creative Commons Zero (CC0-1.0, Public Domain Dedication). See <https://rustacean.net/>

The LLVM wyvern is copyright Apple, Inc. See <https://llvm.org/Logo.html>

The gnu is based on the GNU Head designed by Etienne Suvasa. It is licensed under Creative Commons Attribution-ShareAlike 2.0 (CC-BY-SA-2.0). See <https://www.gnu.org/graphics/agnuhead>

The Cilium logo (hexagons) is a trademark of the Linux Foundation.

The eBPF for WIndows logo (puzzle pieces) is copyright Microsoft Corp. See <https://github.com/microsoft/ebpf-for-windows>

Many thanks to Daniel Borkmann, Karen Chu, Thomas Graf, Marga Manterola, Liz Rice, and Joe Stringer for their reviews and precious feedback.

And kudos to the whole eBPF community, this is all thanks to you! ❤



