UNIVERSIDAD PANAMERICANA

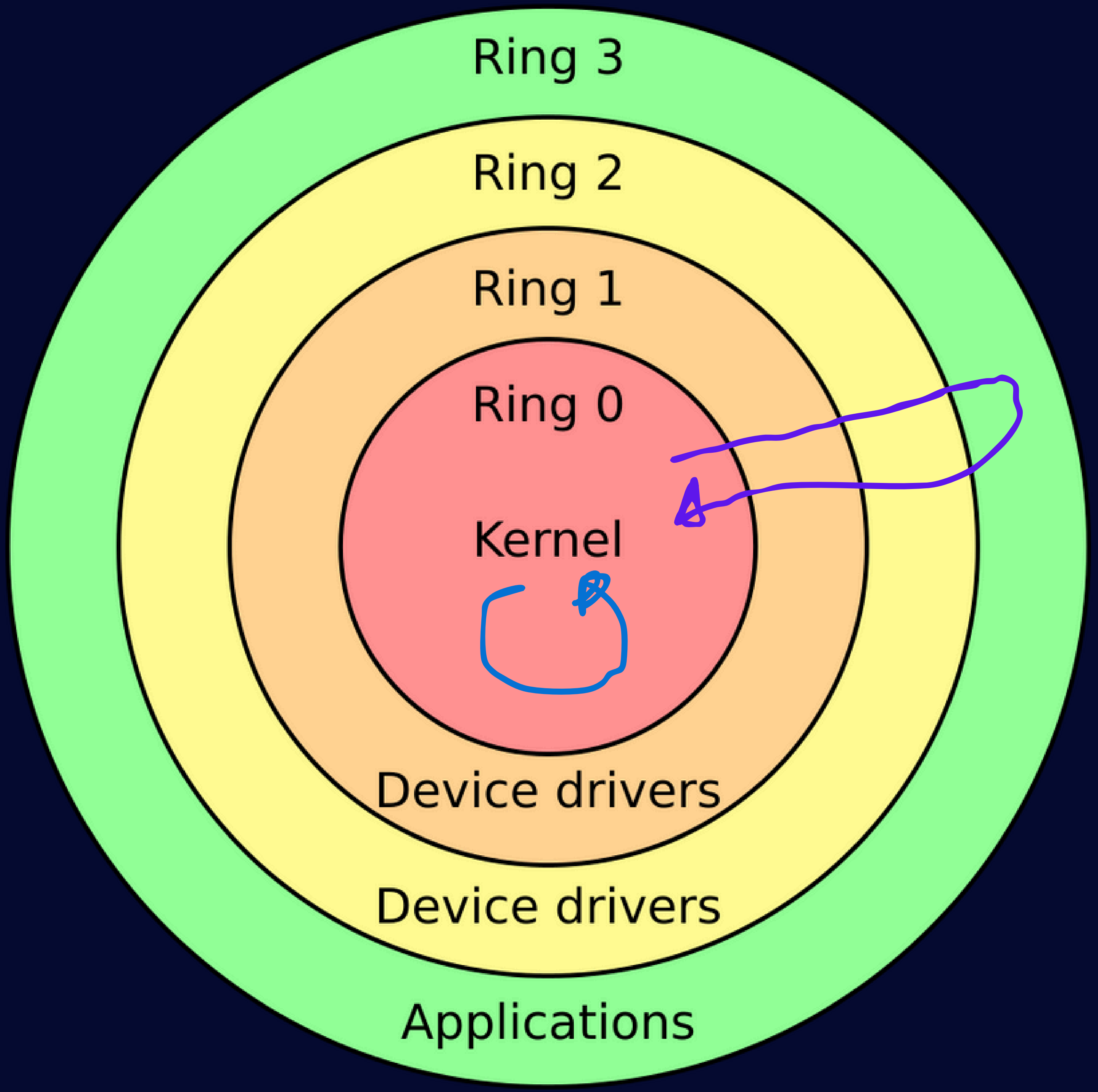
Agent-based Device Audit Monitor

01

EBPF-BASED TOOLCHAIN

KERNEL

A kernel is the core part of an operating system that manages hardware resources and allows communication between hardware and software.



eBPF

"EBPF IS A REVOLUTIONARY TECHNOLOGY THAT CAN RUN SANDBOXED PROGRAMS IN A PRIVILEGED CONTEXT SUCH AS THE OPERATING SYSTEM KERNEL."

—EBPF WEBSITE

Just a few of the things you can do with eBPF include:

NETWORKING

High-performance networking, with built-in visibility

SECURITY & MONITORING

- Detecting and (optionally) preventing malicious activity

PERFORMANCE

Performance tracing of pretty much any aspect of a system

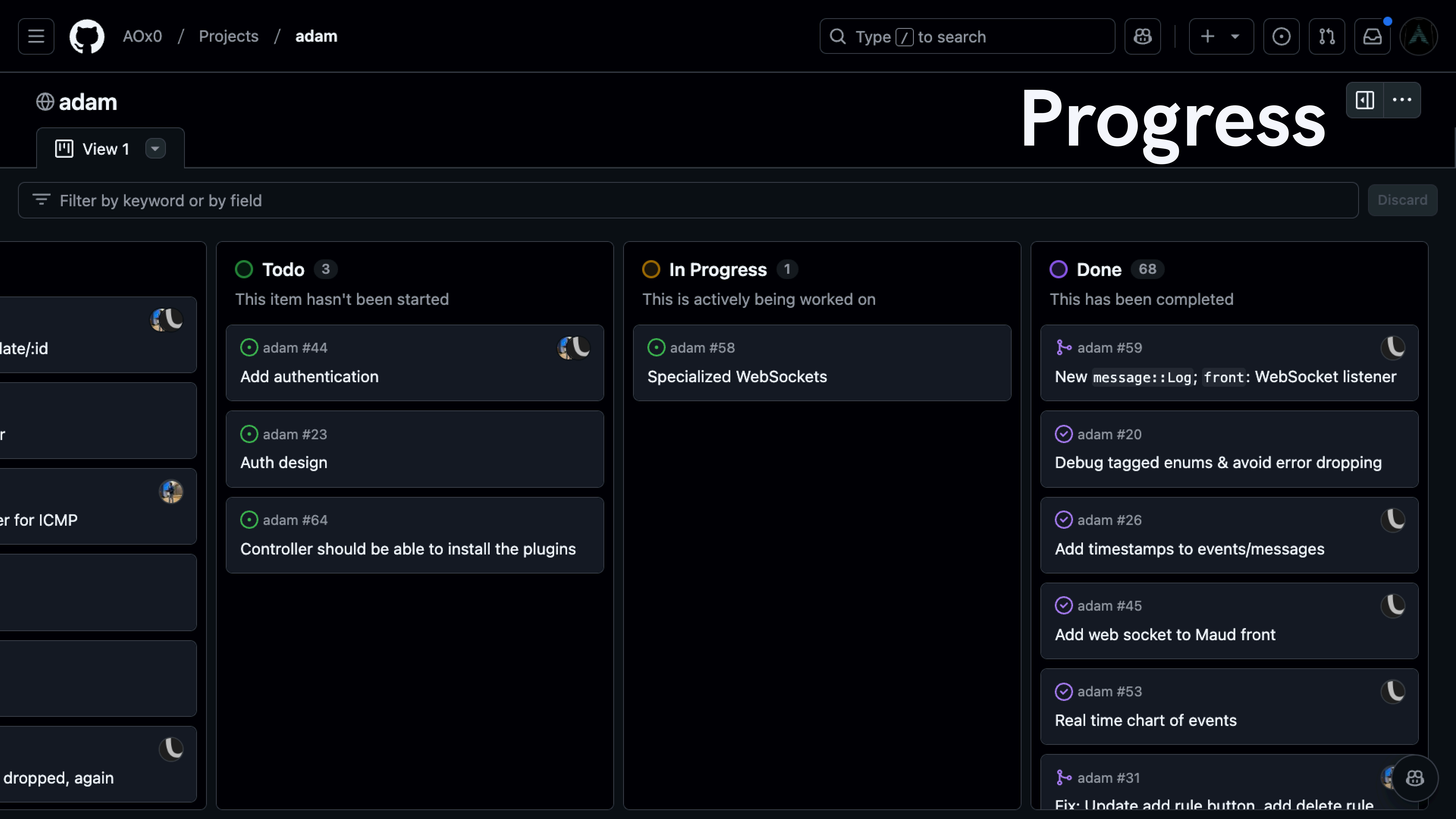
Objective

Explore the possibilities about eBPF, learn the technology

Develop a platform that allows the installation and management of eBPF applications in linux machines.

The platform allows the installation of plugins like firewalls, load balancers, process monitoring and usage of the eBPF LSM

Progress



late/:id

r

er for ICMP

dropped, again

Todo 3
This item hasn't been started

adam #44
Add authentication

adam #23
Auth design

adam #64
Controller should be able to install the plugins

In Progress 1
This is actively being worked on

adam #58
Specialized WebSockets

Done 68
This has been completed

adam #59
New `message::Log`; `front`: WebSocket listener

adam #20
Debug tagged enums & avoid error dropping

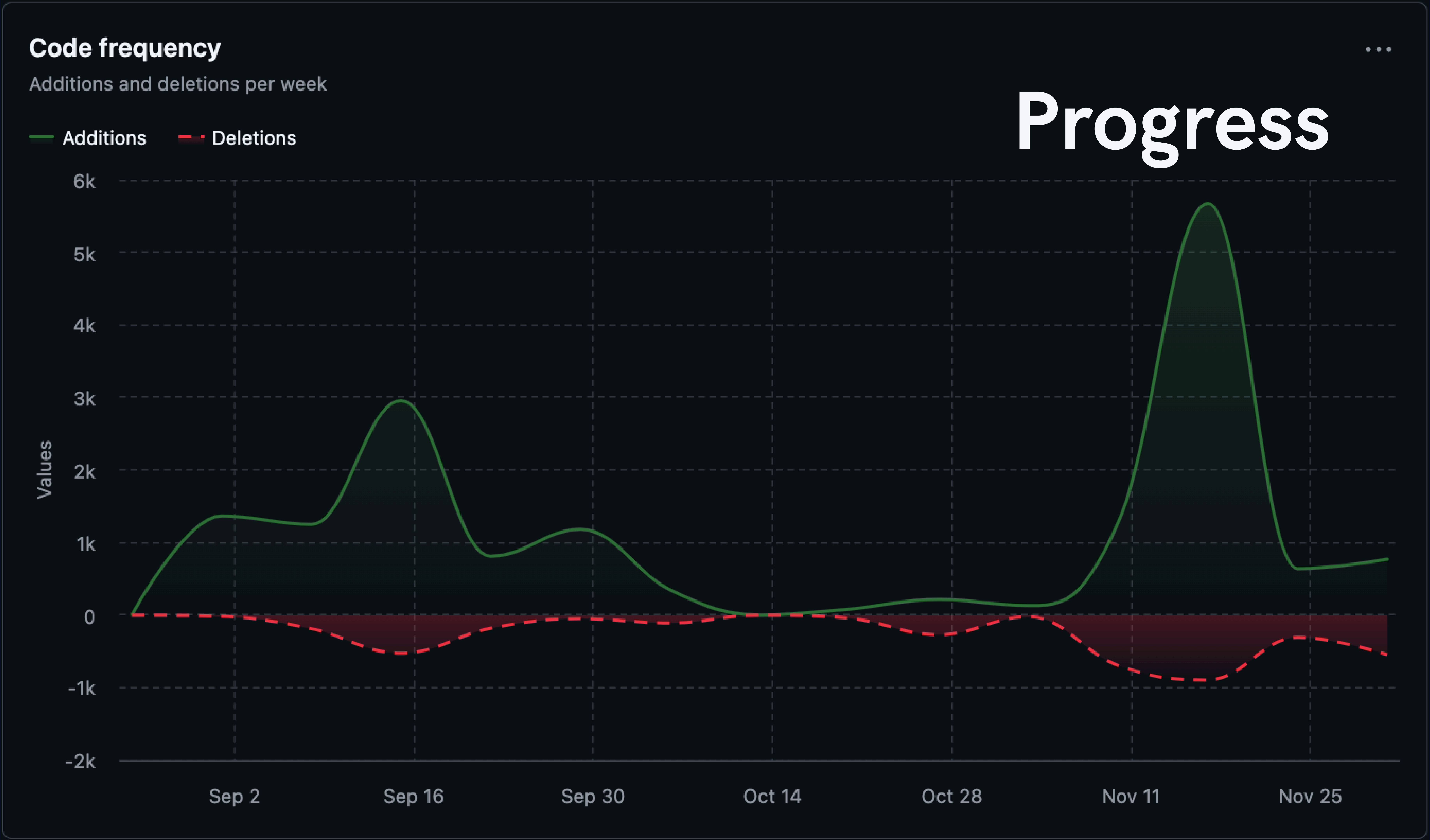
adam #26
Add timestamps to events/messages

adam #45
Add web socket to Maud front

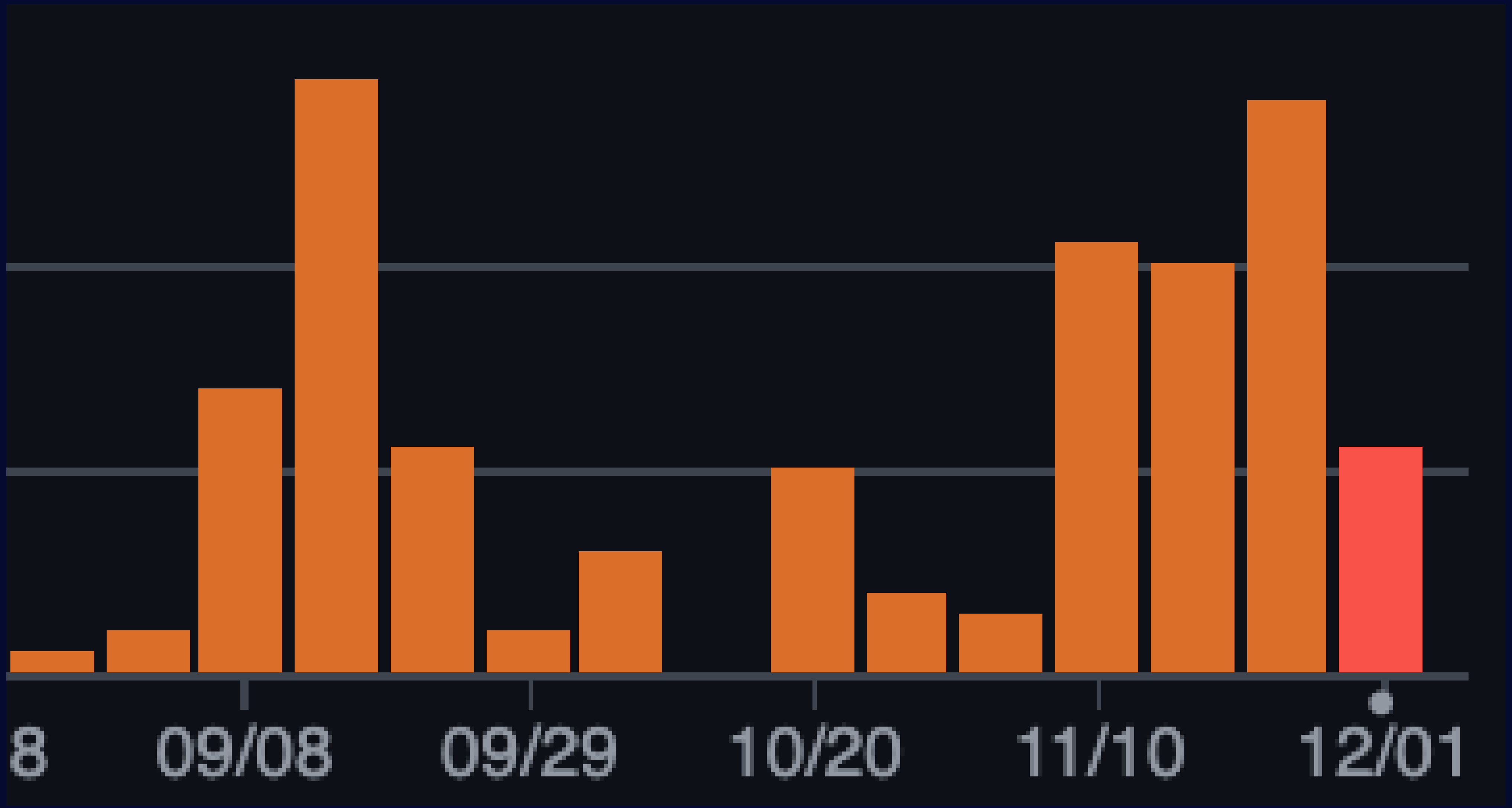
adam #53
Real time chart of events

adam #31
Fix: Update add rule button, add delete rule

Progress


Code frequency
Additions and deletions per week
Additions
Deletions
Values
6k
5k
4k
3k
2k
1k
0
-1k
-2k
Sep 2
Sep 16
Sep 30
Oct 14
Oct 28
Nov 11
Nov 25

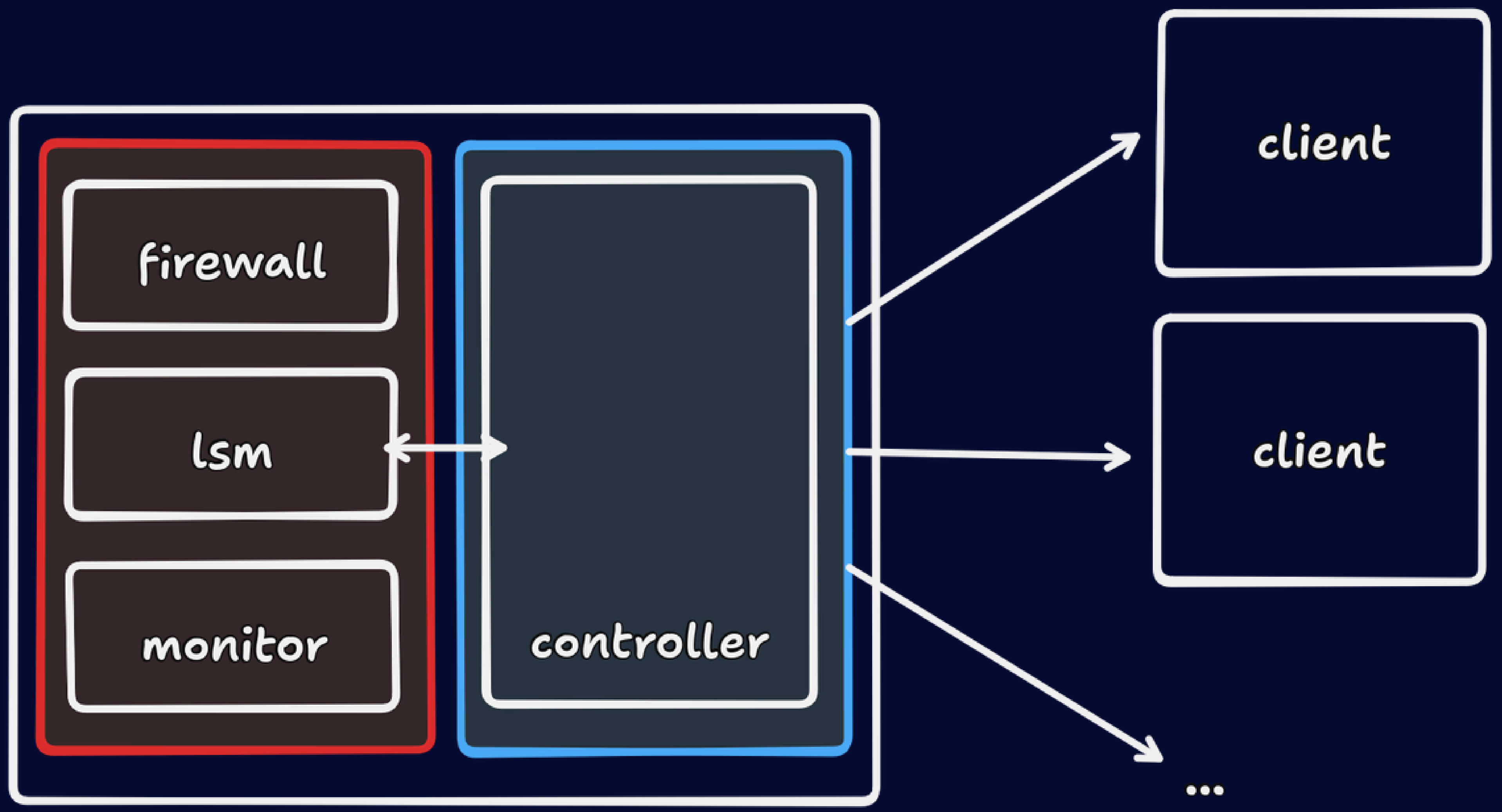
Progress



Architecture

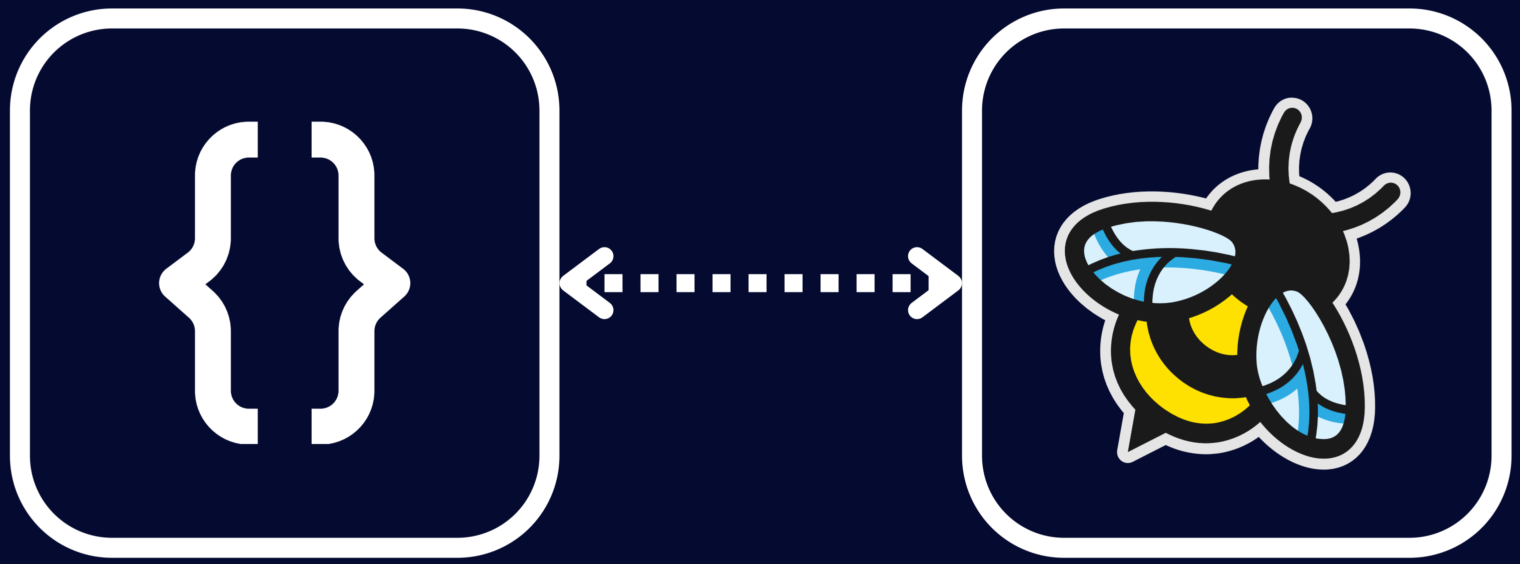
Main components:

- Per machine:
  - eBPF plugins (i.e. firewall, lsm, monitor)
  - controller: API to manage plugins
- Clients:
  - Concrete implementations that communicate to the controller(s)

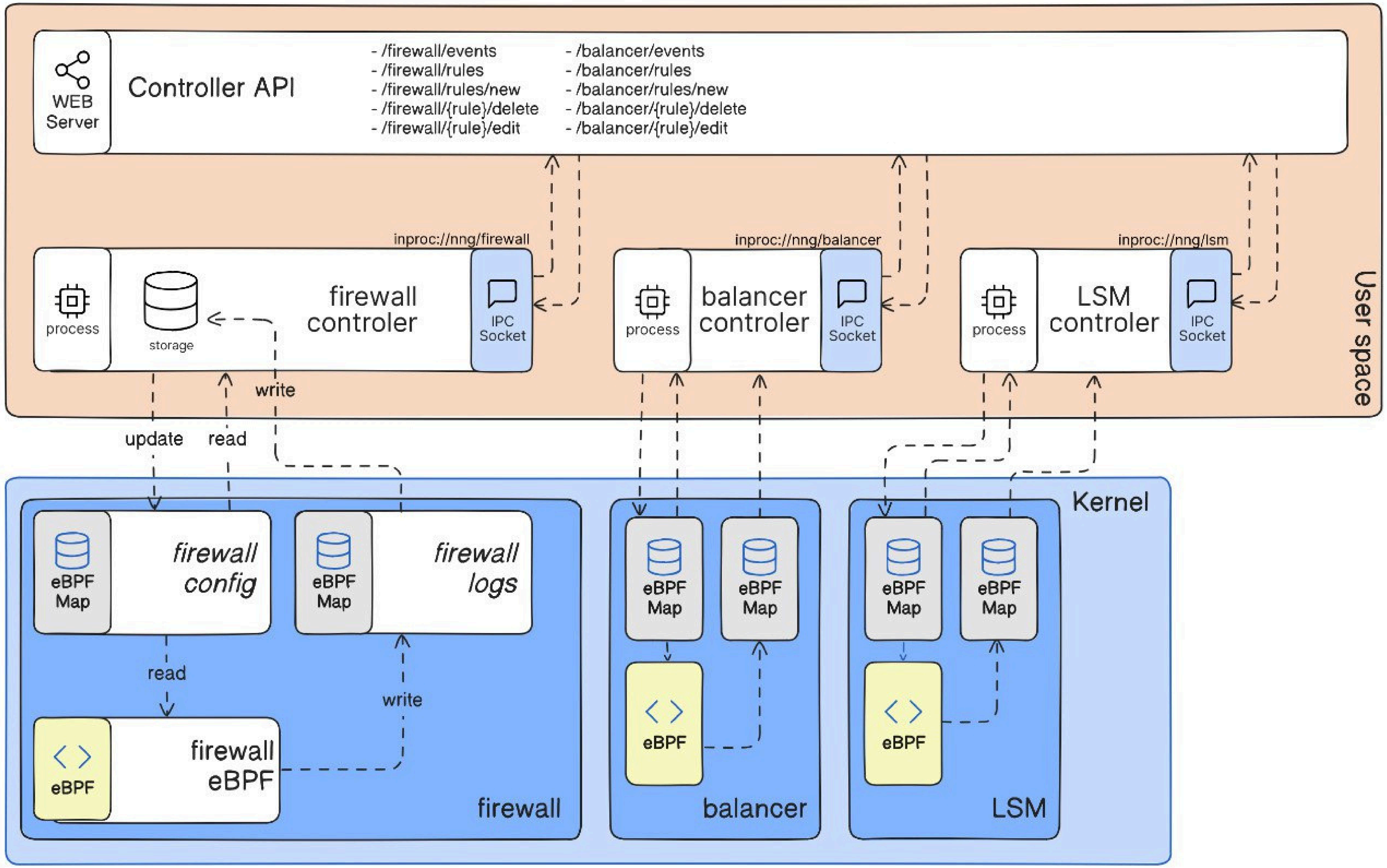


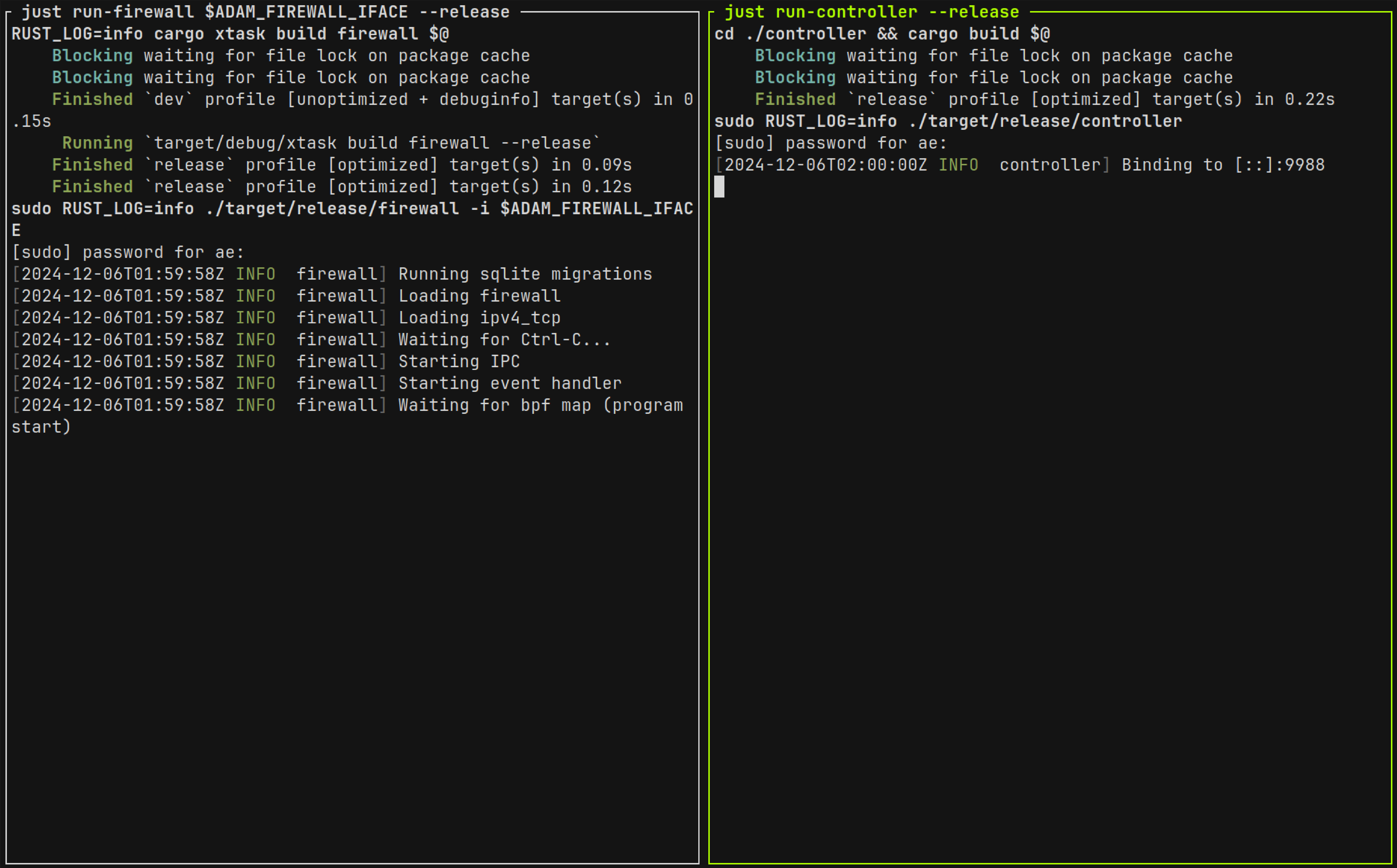
02

CURRENT ACHIEVEMENTS



EBPF PROGRAMS CONTROLLER


WEB Server
Controller API
- /firewall/events
- /firewall/rules
- /firewall/rules/new
- /firewall/{rule}/delete
- /firewall/{rule}/edit
- /balancer/events
- /balancer/rules
- /balancer/rules/new
- /balancer/{rule}/delete
- /balancer/{rule}/edit
inproc://nng/firewall
inproc://nng/balancer
inproc://nng/lsm
process
storage
firewall controler
IPC Socket
process
balancer controler
IPC Socket
process
LSM controler
IPC Socket
User space
write
update
read
Kernel
eBPF Map
firewall config
eBPF Map
firewall logs
read
write
eBPF
firewall eBPF
firewall
eBPF Map
eBPF Map
eBPF
balancer
eBPF Map
eBPF Map
eBPF
LSM



```
just run-firewall $ADAM_FIREWALL_IFACE --release
RUST_LOG=info cargo xtask build firewall $@
    Blocking waiting for file lock on package cache
    Blocking waiting for file lock on package cache
    Finished `dev` profile [unoptimized + debuginfo] target(s) in 0
.15s
     Running `target/debug/xtask build firewall --release`
    Finished `release` profile [optimized] target(s) in 0.09s
    Finished `release` profile [optimized] target(s) in 0.12s
sudo RUST_LOG=info ./target/release/firewall -i $ADAM_FIREWALL_IFAC
E
[sudo] password for ae:
[2024-12-06T01:59:58Z INFO  firewall] Running sqlite migrations
[2024-12-06T01:59:58Z INFO  firewall] Loading firewall
[2024-12-06T01:59:58Z INFO  firewall] Loading ipv4_tcp
[2024-12-06T01:59:58Z INFO  firewall] Waiting for Ctrl-C...
[2024-12-06T01:59:58Z INFO  firewall] Starting IPC
[2024-12-06T01:59:58Z INFO  firewall] Starting event handler
[2024-12-06T01:59:58Z INFO  firewall] Waiting for bpf map (program
start)
```

```
just run-controller --release
cd ./controller && cargo build $@
    Blocking waiting for file lock on package cache
    Blocking waiting for file lock on package cache
    Finished `release` profile [optimized] target(s) in 0.22s
sudo RUST_LOG=info ./target/release/controller
[sudo] password for ae:
[2024-12-06T02:00:00Z INFO  controller] Binding to [::]:9988
```

WEB FRONTEND

HATEOAS

HTMX-BASED FRONTENDS SUPPORT

Introduces support for HTMX components rendering on-demand from the controller backend.

Adds a basic frontend implementation based on HTMX.

- Conditional server render based on request headers
- Enhanced API and new methods
- Makes the controller frontend-aware

Multiple Controller Management

The frontend allows to add and monitor multiple machines’ controllers

- View real-time packet flow
- Manage the registered machines in the frontend
- Manage the rules of the firewall for each machine
- Select an active machine out of all registered ones

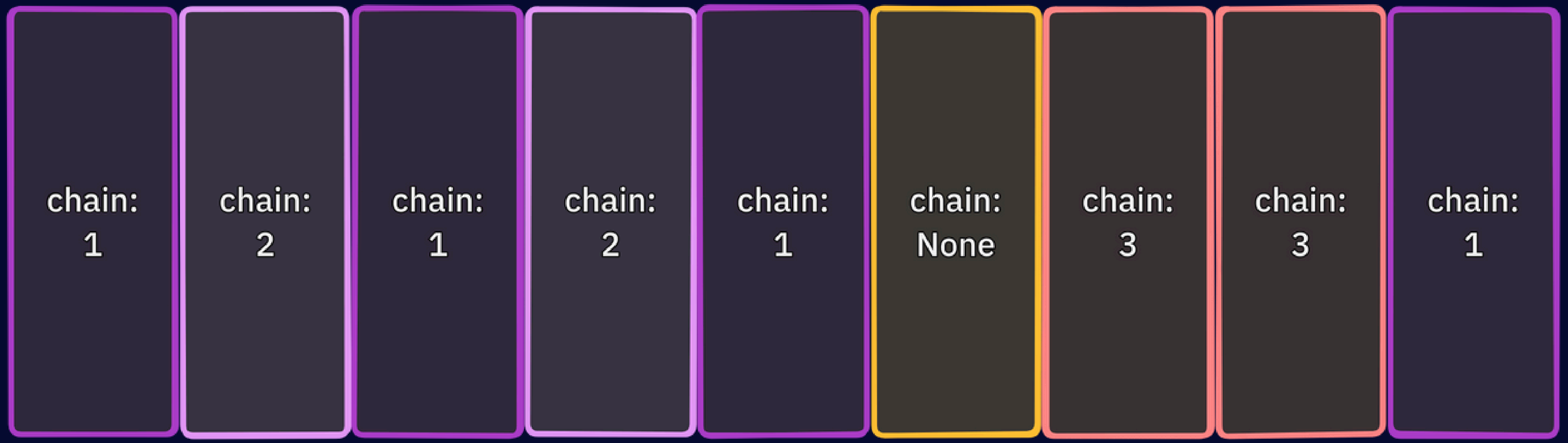
RULE CHAINS

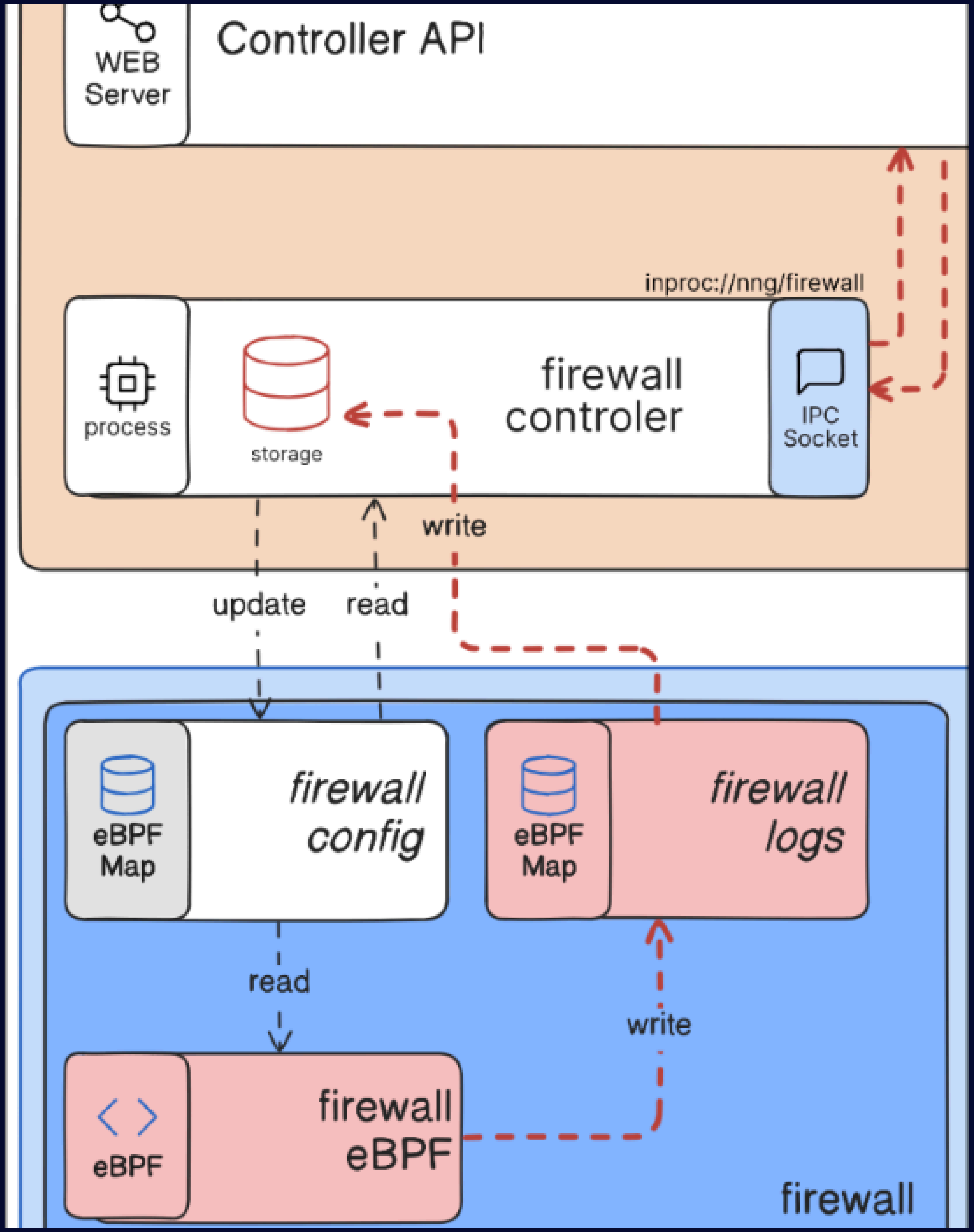
Introduces the ability to create AND relation between rules.

- Enables complex rules based on multiple conditions

Challenges:

- Requires an in-memory table to track rule matching
- Introduces complexity to kernel-level code
- Presents questions about struct serialization versions





Event persistence

STORE AND RETRIEVE EVENT LOGS FROM A SQLITE STORE.

- Query by:
  - Last N hours/days/months
  - Periods of time

- Security implications
- Enhanced API and new methods
- Makes the controller frontend-aware

Let’s make history