Navegando logs usando lnav

Caio Volpato (caioau.net)

CoffeOps

20 de dezembro de 2025

Resumo do problema

Precisamos migrar um serviço web, a migração ia alterar o registro DNS pra apontar pra nova maquina, mas alguns clientes tinham o IP hard-coded.

Então precisamos ter uma forma de identificar esses clientes e atualizar a referência.

O que é o lnav?

- Lnav é um navegador de logs pra linha de comando (cli)
- Além de filtragens tradicionais e etc ... suporta formato de logs populares permitindo rodar queries SQL (sqlite)
- As queries podem ser executadas além da forma interativa, mas automatizadas pra gerar relatórios.

Lendo os logs

O lnav suporta diversas formas pra acessar os logs, as principais:

1. Apontando pro arquivos locais: `lnav arq1 arq2`
2. Piping: `journalctl -o json | lnav`
3. Saida de comandos: `lnav -e 'make -j8'`
4. Como pager: `export PAGER='lnav -q'`
5. Arquivos remotos (http, ssh)

lnav vs instrumentação

Stacks de monitoramento como ELK coletam os logs em larga escala e armazenam em bancos de dados permitindo queries e analytics.

Ao passo que o lnav é uma ferramenta local, sem a escala das stacks, ideal pra troubleshooting de aplicações.

Instalando o lnav

- Está disponível na maioria das distros: `sudo apt install lnav`
- Pra versões mais recentes, binário no github: github.com/tstack/lnav

Tutorial

Semelhante ao vimtutor (pro vim), o lnav tem um tutorial interativo:

```
ssh tutorial1@demo.lnav.org
```

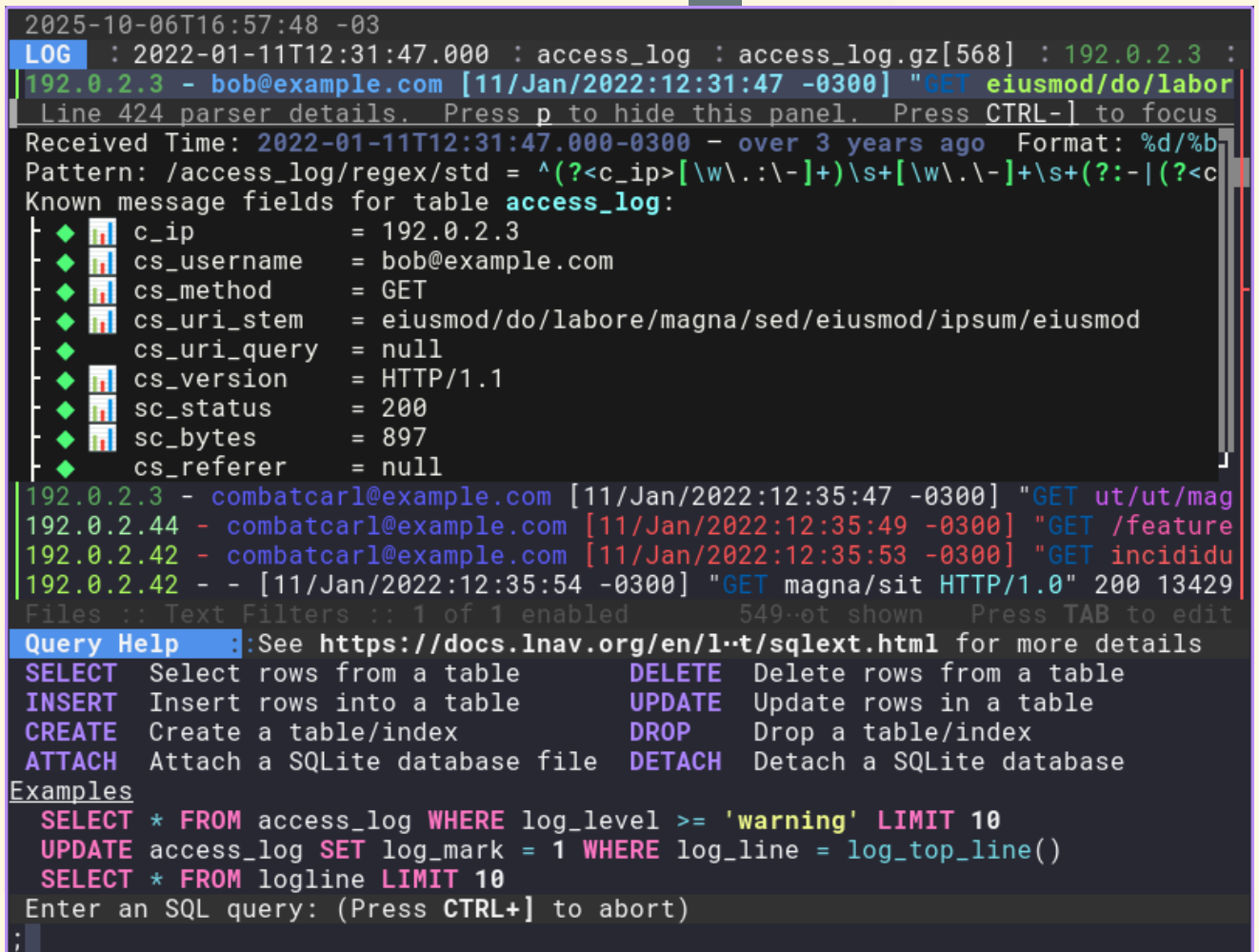
Alguns atalhos

Doc: docs.lnav.org/en/latest/hotkeys

- Sintaxe vim: `/filter` pra saltar pras linhas filtradas e `n` pra prox
- Erros: `e` pra saltar pras linhas com error (Shift-E pra anterior)
- Warning: `w`
- `D`: salta pro inicio do prox dia
- `M`: cria um "bookmark" da linha selecionada.

Interface SQL

Depois de reconhecer os logs, digite `;` pra abrir o SQL



Exemplo: Tarpit SSH (endlessh)

- O que é um tarpit?
  - Tarpit é um serviço de rede que intencionalmente insere delays, forçando clientes a esperar. Se feito corretamente "sequestra" recursos dos atacantes/bots pendurando conexões abertas.
- endlessh é um tarpit de SSH, pois segundo a RFC do ssh RFC4253
  - Depois da conexão TCP é estabelecida, um banner de identificação deve ser enviado como `SSH-2.0-OpenSSH_8.9p1`
  - Porém a RFC não estabelece um limite, então o tarpit lentamente continua o enviando, "prendendo" os clientes
- Como cliente, no openssh a opção `ConnectTimeout`

Criando um formato de log novo

- O formato de log é definido um json que define seu schema
- Em resumo é um regex pra fazer match das linhas e capturar os campos
- (Opcional) dar tipo (str, int, float etc ...) aos campos
- (Opcional) definir "level" como error, warning etc ...
- Exemplo: web servers access_log.json schema

Criando um formato de log novo (esquema fácil)

- O esquema mais fácil é pegar algumas linhas do log e colocar no regex101, criar um regex e compartilhar um link do regex101
- Basta "importar" o link do regex101 que o json schema é gerado automaticamente.
- `lnav -m regex101 import https://regex101.com/r/TEksAA/1 endlessh`
- Uma vez importado basta fazer alguns consertos (em `~/.config/lnav/formats/`) como tipos dos campos.

Resultado: queries

```
# general stats
select min(log_time) as first_log,max(log_time) as last_log,
        count(DISTINCT strftime('%Y%m%d', log_time)) as uniq_days,
        count(*) as conns,
        count(DISTINCT host),min(traps), max(traps), min(bytes)
    from endlessh
```

Resultado: Script automatizado

Além de usar o as queries de forma interativa é possível escrever "scripts" e executa-las de forma programática, para gerar relatórios.

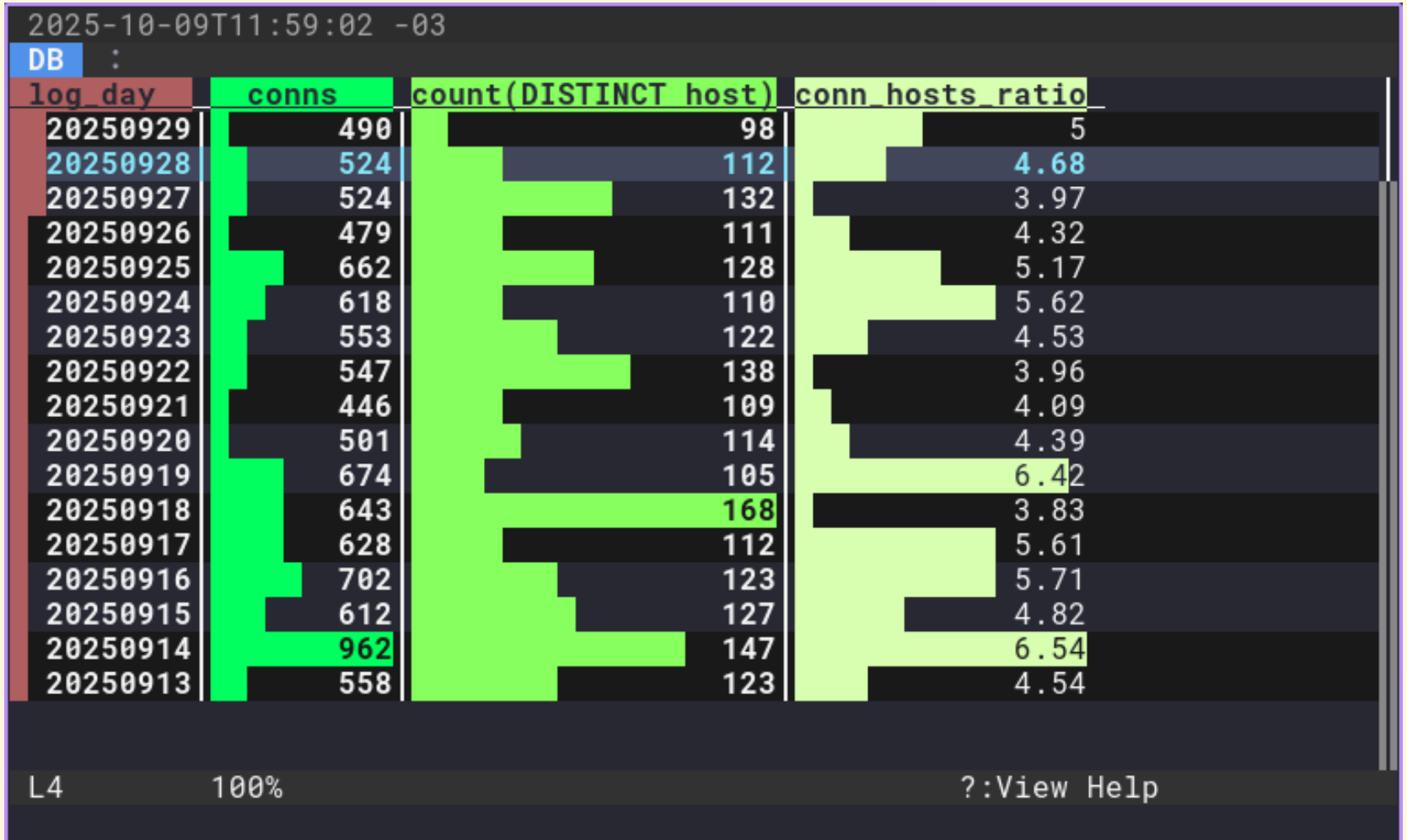
```
lnav -n -f script.lnav log.txt
```

```
# script.lnav
:redirect-to
:filter-in CLOSE

;select min(log_time) as first_log
    from endlessh;
:write-table-to -
```

Resultado: gráficos

Quando executadas no modo interativo gráficos são gerados

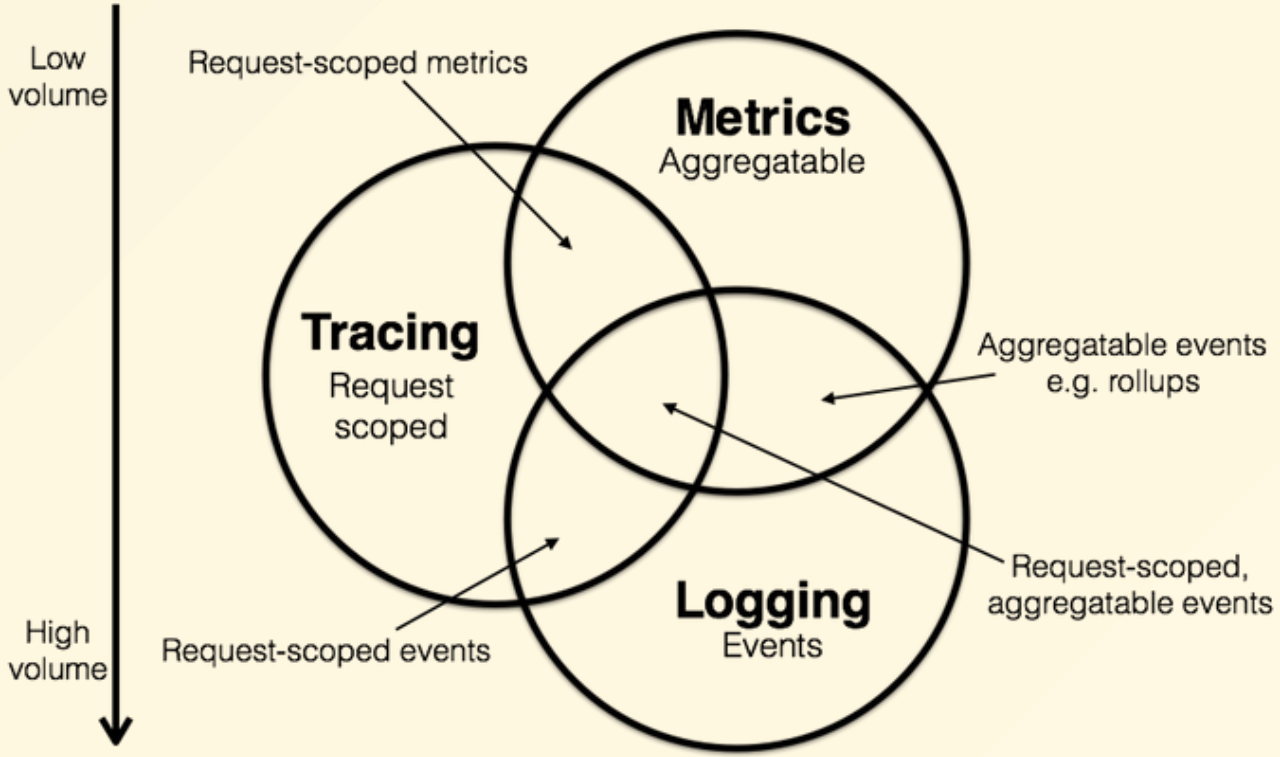


log_day	conns	count(DISTINCT host)	conn_hosts_ratio
20250929	490	98	5
20250928	524	112	4.68
20250927	524	132	3.97
20250926	479	111	4.32
20250925	662	128	5.17
20250924	618	110	5.62
20250923	553	122	4.53
20250922	547	138	3.96
20250921	446	109	4.09
20250920	501	114	4.39
20250919	674	105	6.42
20250918	643	168	3.83
20250917	628	112	5.61
20250916	702	123	5.71
20250915	612	127	4.82
20250914	962	147	6.54
20250913	558	123	4.54

como dev: crie métricas

Não dependa somente dos logs pra gerar suas métricas.

Fonte: [Peter Bourgon -- Metrics, tracing, and logging]



como dev: import slog

Afim de facilitar os "analytics" dos logs podemos usar bibliotecas de log estruturadas como slog (golang) e structlog (python)

Um cuidado é sempre que for logar o mesmo tipo de evento gere sempre os mesmos campos pra não quebrar as queries

```
slog.Info("hello from slog", "value", 42, "other_value", 2025)
```

```
2025/10/09 12:18:45 INFO hello from slog value=42 other_value=202
```

Modern Unix

Repositório: github.com/ibraheemdev/modern-unix

- cat -> bat
- grep -> ugrep/ripgrep
- top -> btop
- df -> dust
- find -> fd
- git -> lazygit
- docker -> lazydocker

nushell.sh

Um shell com features SQL

```
/usr/bin 〉ls | where size > 10mb | sort-by modified
```

#	name	type	size	modified
0	x86_64-linux-gnu-lto-dump-10	file	23.3 MiB	a year ago
1	micro	file	13.7 MiB	8 months ago
2	buildah	file	19.8 MiB	7 months ago
3	qemu-system-i386	file	13.7 MiB	5 months ago
4	qemu-system-x86_64	file	13.7 MiB	5 months ago
5	node	file	76.6 MiB	a month ago

Missing semester of your CS education

Um curso que se propõe a preencher as lacunas da formação de uma graduação teórica, ensinando tópicos como git, profiling e debugger.

missing.csail.mit.edu