Inbox

- [ ] Tunning & profiling
  - [ ] 研究Arthas内部实现原理
  - [ ] 探究Arthas基本使用、与调优实践（watch, trace, dashboard, retransform）
  - [ ] Async-Profiler 与linux perf 信号
  - [ ] 生成火焰图
- [ ] hsdis + JITWatch —— 探究"代码消失术"

  JVM 为了优化把它给"抹掉"了（Dead Code Elimination），或者你想看 volatile 到底怎么生成的 CPU 屏障
  - [ ] 配合 hsdis 插件，让 JVM 在运行时把汇编代码（Assembly） dump 出来
- [ ] Wireshark/Tcpdump —— Netty 的"上帝视角"
  - [ ] Netty 的 Zero-Copy（零拷贝）到底有没有减少上下文切换
  - [ ] Linux Epoll
  - [ ] Linux io uring
- [ ] eBPF + Java
  - [ ] 在内核态用 eBPF 直接拦截 JVM 的符号调用
  - [ ] 使用 bpftrace 或 ply 编写脚本。你可以绕过 JVM，去统计 Linux 内核调度 Java 线程的延迟（Scheduling Latency）
- [ ] Java FFI

  Java 17+ 推出的 Foreign Function & Memory API
  - [ ] 不使用任何 Java 集合类，而是用 MemorySegment 在**堆外（Off-heap）**手动申请一段物理内存，像 C 语言一样用 Offset 操作数据，然后再把这个内存段传给一个用 Rust 写的动态链接库进行处理
- [ ] AOT
  - [ ] graalVM AOT 编译

- [ ] Byte Buddy / ASM：代码生成的“造物主”
  - [ ] 编写一个程序，在运行时偷偷修改另一个类的逻辑。比如：拦截所有带 @ToString 注解的类，在字节码层面给它们动态注入一段加密逻辑
- [ ] Debugger 实现原理
  - [ ] ptrace 等系统调用
- [ ] 使用JOL分析java 对象 Today
  - [ ] 空对象到底占多大 Today
  - [ ] 观察内存对齐及Padding情况 Today
  - [ ] 对象加 synchronized 锁时，JOL 可以让你实时看到 Mark Word 的二进制位是如何从 01（无锁）变成 00（轻量级锁）甚至 10（重量级锁）的 Today
  - [ ] 对比 ArrayList<Integer> 和 int[] 在存放 100 万个数字时，内存占用差了多少倍 Today
  - [ ] Layout 与 Segment API
- [ ] 用 Java 的 JFR (Java Flight Recorder) 事件流，自己写一个能实时推送系统热点到你 EndeavourOS 任务栏的小程序
  - 在实现这个工具的过程中，你会为了减少对被观测程序的性能影响，不得不去研究什么是 SafePoint，什么是 Zero-overhead Profiling
  - [ ] SafePoint
  - [ ] Zero-overhead Profiling
- [ ] Java 性能监控插件”，集成在 Zsh 里
  - [ ] Java 代码怎么通过 attach 机制挂载到进程
  - [ ] Instruction（指令集）怎么解析
  - [ ] 研究 Linux 的 /proc 系统和 perf 事件
- [ ] 基于 Netty 和微服务架构的“个人工作台系统”
  - 它能通过 eBPF 监控你系统的负载。
  - 它能通过 Netty 和你的所有设备通信。
  - 它的核心模块用 Rust 编写，通过 Panama API 挂载在 Java 主程序上

- [ ]

- [ ] Others Research

  - [ ] 编译openjdk 源码并修改 System.out.println 的底层 C++ 实现，让它在每次打印时自动带上当前的系统温度或内核版本

  - [ ] 有栈协程、无栈协程

    - Rust 里的 async/await（基于无栈协程，State Machine）和 Java 21 的虚拟线程（有栈协程，基于 Continuation

  - [ ] hsdis

  - [ ] 为什么在 Java 里，几个无意义的 long 字段（JOL 能看出来的 Padding）能让程序的性能提升好几倍

  - [ ] LMAX Disruptor

    - [ ] 环形队列（RingBuffer）、缓存行对齐（Cache Line Padding）

  - [ ] 去研究 CPU 缓存架构（L1/L2/L3）以及 MESI 协议。你会发现，Java 也能写出比烂代码写的 C 语言快得多的程序

  - [ ] Java 智能 Agent

    - [ ] java.lang.instrument。写一个 JAR 包，在任何 Java 程序启动时挂载它

      统计所有方法的执行时间，或者自动拦截所有的数据库查询并实时推送到你的 Linux 桌面通知。
      - 理解字节码、JVM 内部通信机制、以及终端交互设计

  - [ ] 写一个 Java 程序，利用 eBPF 捕获网络包，然后通过 Netty 异步推送到你自建的 VictoriaLogs 里，最后用 Grafana 画出一条完美的全链路监控线

  - [ ] 看 Linux 源码里 fs/io_uring.c 是怎么写的，再看 Java 是怎么通过原生方法调用（Native Method）把数据塞进那个共享内存环的

  - [ ] CRaC (Coordinated Restore at Checkpoint)：Java 的“瞬间移动”

    - [ ] Linux 内核的 CRIU (Checkpoint/Restore In Userspace) 技术

      - Project CRaC。它允许你在 Java 应用运行到“巅峰状态”（JIT 已经预热完毕）时，给整个 JVM 做一个快照（Checkpoint），保存到磁盘上
      - 下次启动时，你不是从 main 方法开始跑，而是直接从快照恢复（Restore）。原本需要 10 秒启动的应用，现在只需要 50 毫秒

- [ ] 研究 Happens-Before 原则背后的偏序关系（Partial Ordering），以及 Weak Memory Model (弱内存模型)

  - 当你理解了什么是 Sequential Consistency，什么是 Store Buffers 和 Invalidate Queues 时，你会发现无论是 Java、Rust 还是 C++，在并发层面的本质全是一样的数学逻辑

- [ ] 写一个工具，它能自动检测运行中的 Java 应用是否有 Log4j 类似的漏洞，并在不重启、不修改源码的情况下，动态地把漏洞代码"置换"掉

- [ ] 用 Java / Netty 写一个你自己的日志采集 Agent，要求性能超越 Vector（Rust 写的那位）。

  - 第一步： 用 Java / Netty 写一个你自己的日志采集 Agent，要求性能超越 Vector（Rust 写的那位）。
  - 第二步： 为了超越它，你必须用上 io_uring，用上 Disruptor（环形缓冲区），甚至要用 JOL 来优化你的 Buffer 对象布局以减少 Cache Miss。
  - 第三步： 用 Async-Profiler 生成火焰图，看看 Java 的开销在哪里；用 hsdis 看看 JIT 是不是把你最核心的循环给内联了（Inlining）。
  - 第四步： 当你的 Java 程序在某些指标上真的跑赢了 Rust 或者是 C 的时候，你还会有"缺乏动力"的感觉吗？

- [ ] jemalloc 算法

  - Netty 的核心根本不是 Java，而是对 OS 内存布局和 CPU 缓存行 的极致压榨

- [ ] 类脑计算（Neuromorphic Computing）

  - 不再琢磨怎么写复杂的 Java 业务逻辑，而是在琢磨怎么用电路模拟神经元的脉冲发射（Spiking Neural Networks）

- [ ] 拉普拉斯妖（Laplace's Demon）

- [ ] #Source Thingsboard TBMQ Sources Jan 14

- [ ] #Source Netty Jan 14

- [ ] Jan 14

- [ ] Android GKD, shizuku 2025-10-28

- [ ] 定时任务与惊群问题及随机延迟 2025-10-27

- [ ] #NeoVim #NvChad　Basics neovim nvchad

  - space + e : 侧边栏文件树
  - space + ff : 文件搜索
  - space + sv: 垂直分屏
  - space + sh: 水平分屏
  - space + sc: 关闭当前分屏

  - :PackerSync: 同步更新插件
  - PackerSync　" 同步更新插件

- :PackerInstall " 安装插件
- :PackerUpdate " 更新插件
- :PackerClean " 删除未用插件

- [ ] docker compose 2025-09-24

- profiles: group service by logic and then can start by specify --profile option
- multi-stage build so get small image

- [ ] Thingsboard "Login As Tenant Admin" 原理 2025-09-13

```text
This mechanism allows administrators to generate tokens for users who haven't set passwords yet, enabling the "Login as Tenant Admin" functionality even for inactive users. The system bypasses password authentication entirely in this administrative impersonation flow, which is why it works even when user X has no password set.
```

- UserController.java:179
- UserCredentials vs Password
- UserCredentials.java:36-44： not only password
- SecurityUser constructor SecurityUser.java:50-54 doesn't require a password - it only needs the enabled status and principal information.
- Token Generation Without Password
- tokenFactory.createTokenPair(securityUser) call UserController.java:180 generates JWT tokens based on the SecurityUser object, not on password validation. This is the key difference from normal login authentication.

- [ ] 只在本地应用里利用 Spring Cloud Context 提供的 @RefreshScope 和 /actuator/refresh 来实现运行时刷新 2025-09-10

- [ ] ConfigDataLoader / ConfigDataLocationResolver 2025-09-10

- [ ] JFR Event Streaming API 2025-08-29

- [ ] Jackson Mixins 2025-08-28

- [ ] https://spring.io/projects/spring-modulith

- [ ] Netflix DGS [Home - DGS Framework](https://netflix.github.io/dgs/)

- [ ] [</> htmx - high power tools for html](https://htmx.org/)

- [ ] [About Mermaid | Mermaid](https://mermaid.js.org/intro/)

- [ ] [Getting started | Ajv JSON schema validator](https://ajv.js.org/guide/getting-started.html)

- [ ] [GitHub - dbt-labs/dbt-cli](https://github.com/dbt-labs/dbt-cli)
- [ ] [Pushpin | Reverse proxy for realtime APIs](https://pushpin.org/)
- [ ] [GitHub - trinodb/trino: Official repository of Trino, the distributed SQL query engine for big data, formerly known as PrestoSQL (https://trino.io)](https://github.com/trinodb/trino)
- [ ] [GitHub - syntasso/kratix: Kratix is an open-source framework for building platforms](https://github.com/syntasso/kratix)
- [ ] [GitHub - pola-rs/polars: Dataframes powered by a multithreaded, vectorized query engine, written in Rust](https://github.com/pola-rs/polars)
- [ ] [Apache Arrow | Apache Arrow](https://arrow.apache.org/)
- [ ] opentelemetry
- [ ] [GitHub - line/armeria: Your go-to microservice framework for any situation, from the creator of Netty et al. You can build any type of microservice leveraging your favorite technologies, including gRPC, Thrift, Kotlin, Retrofit, Reactive Streams, Spring Boot and Dropwizard.](https://github.com/line/armeria)
- [ ] two
- [ ] one
- [ ] Welcome to TickTick

  You have joined the hundreds of thousands of others who are using TickTick to make life easier. TickTick lets you keep track of your tasks, draw up a schedule, and share projects with anyone around you. TickTick currently supports a number of operating system platforms including iPhone, iPad, Mac App, Windows, Apple Watch, Android phone, Android Wear, Web, Chrome Extension, Chrome App, and Firefox Extension. With these 10 platforms, we believe TickTick can always help you achieve more everyday!

What can you do with TickTick?

Add tasks and reminder. Never lose track of your to-do list.

Set flexible repeating intervals for recurring tasks.

Subscribe third-party calendars.

Use folder, list, priority, and tag to make tasks more organized.

Make check items within tasks.

Collaborate on shared lists with your family, friends, and colleagues.