BE FEARLESS USING RUST IN PRODUCTION

ROBOTXY

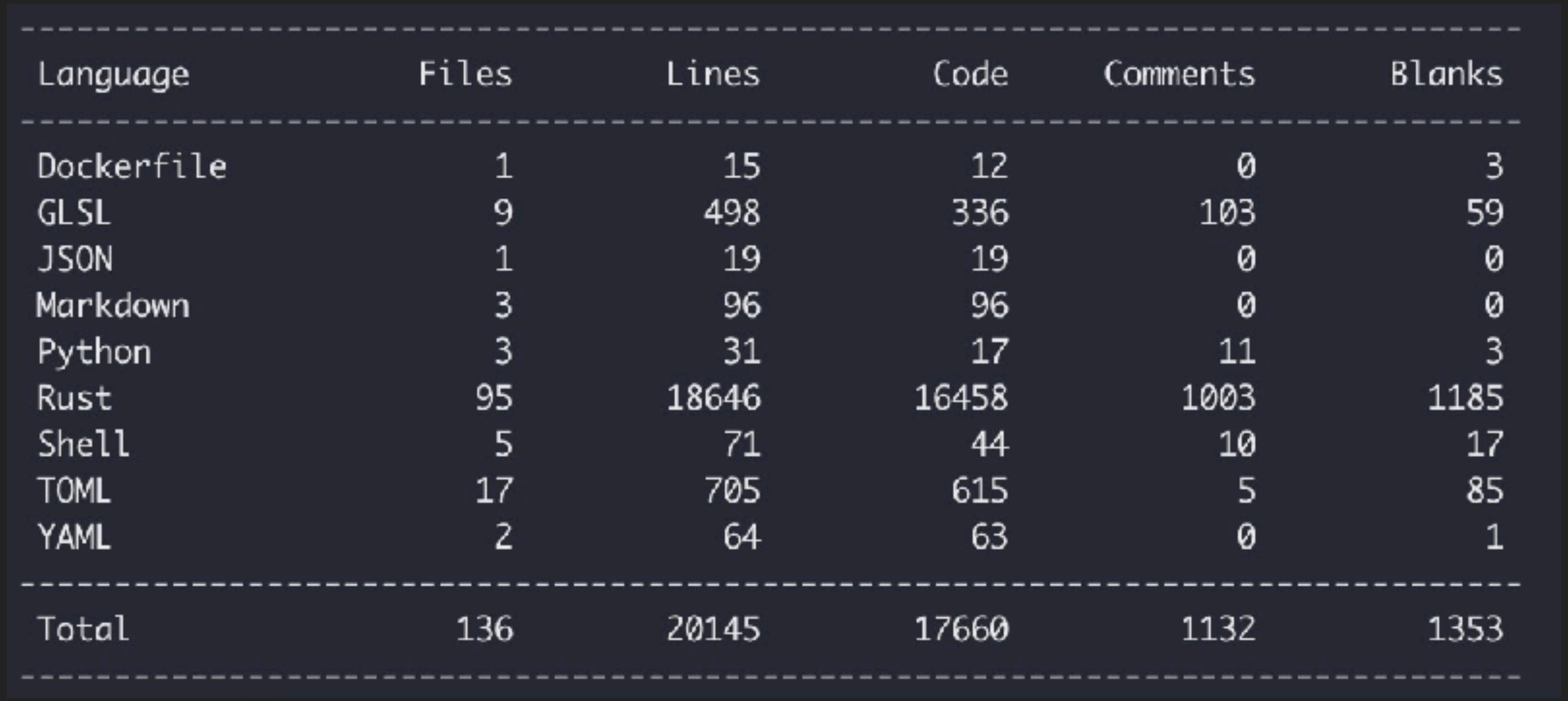
ME

- robotxy 陈雨琪
- Front-end Engineer @ Alibaba
  - Creative tools
  - Image rendering service
- https://github.com/NoXF
- xianyou.cyq@gmail.com

THIS TALK IS ABORT

STATUS

- Rust-nightly 1.35
- 40+ servers (32 core 2.50GHz 128G RAM)
- Dev since 2016



```
-------------------------------------------------------------------------------
 Language            Files        Lines         Code     Comments       Blanks
-------------------------------------------------------------------------------
 Dockerfile              1           15           12            0            3
 GLSL                    9          498          336          103           59
 JSON                    1           19           19            0            0
 Markdown                3           96           96            0            0
 Python                  3           31           17           11            3
 Rust                   95        18646        16458         1003         1185
 Shell                   5           71           44           10           17
 TOML                   17          705          615            5           85
 YAML                    2           64           63            0            1
-------------------------------------------------------------------------------
 Total                 136        20145        17660         1132         1353
-------------------------------------------------------------------------------
```

USAGE

- Rust Web Server
  - Image rendering service
  - SVG parsing & rendering
  - Video encode & decode

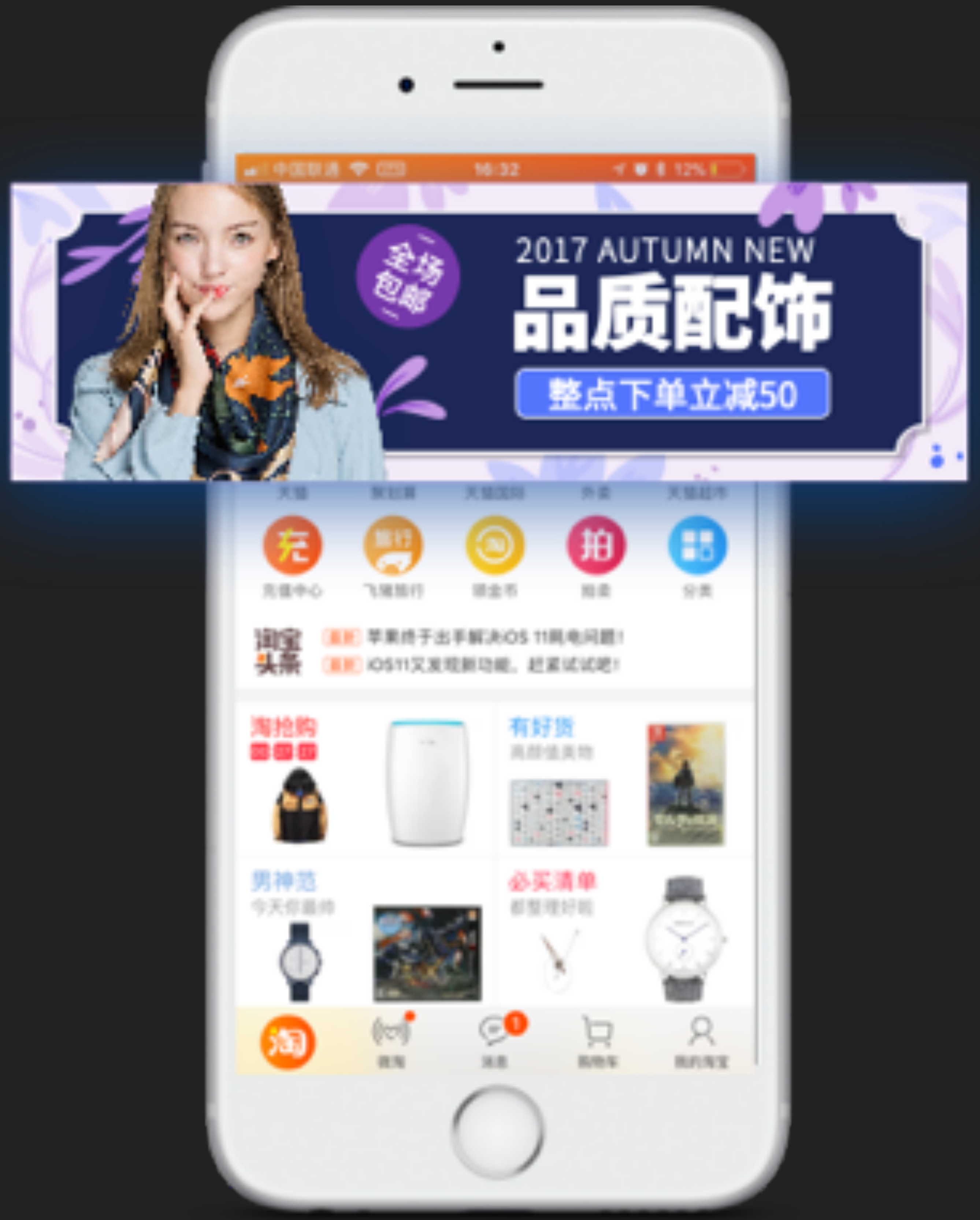
USAGE

- Compute-intensive modules for Node.js
  - Algorithm

WHAT IS MINOLTA?

High-performance and real-time image rendering service

https://alpha.alicdn.com/montage/24514-c3c2bc0b0be7980a4eaa463bf4f25ed0.jpg?content={“10”:{“attrs”:{“value":"https://img.alicdn.com/tfscom/TB1qkpoRFXXXXX.aXXXXXXXXXXX.png","mini":false,"papercut":false}},"12":{"attrs":{"value":"品质配饰"}},"14":{"attrs":{"value":"整点下单立减"}}}

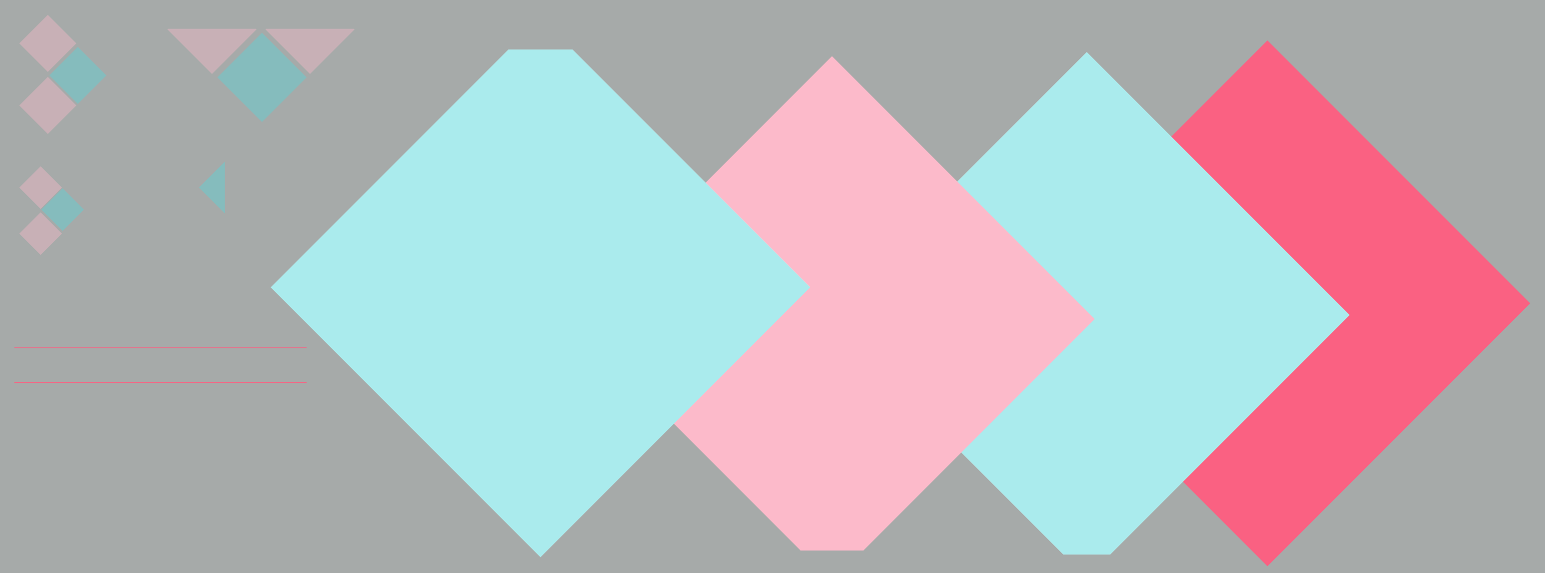


WHAT IS MINOLTA?

LOGO
清凉大作战
海量美衣零点开抢
特价两件包邮
Time: 2018.6.1-6.7

PART2: MINOLTA

Layer



Image

Copy

清凉大作战 海量美衣零点开抢 特价两件包邮 Time: 2018.6.1-6.7

Special

LOGO

PART2: MINOLTA

Layer

Image

Copy

Title	subtitle	action point	decorative
清凉大作战	海量美衣零点开抢	特价两件包邮	Time: 2018.6.1-6.7

Get copy type

Special

LOGO

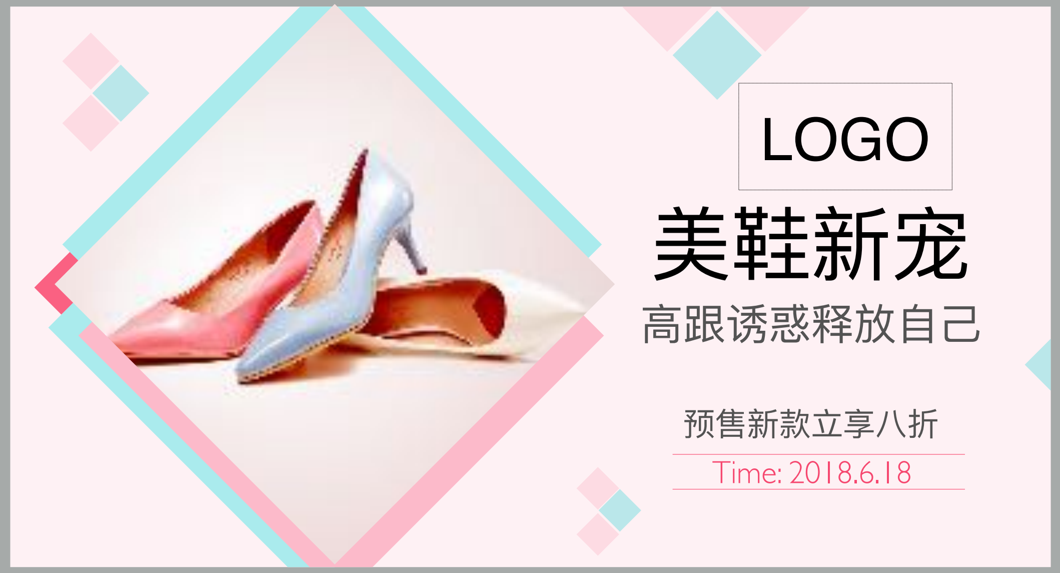
PART2: MINOLTA

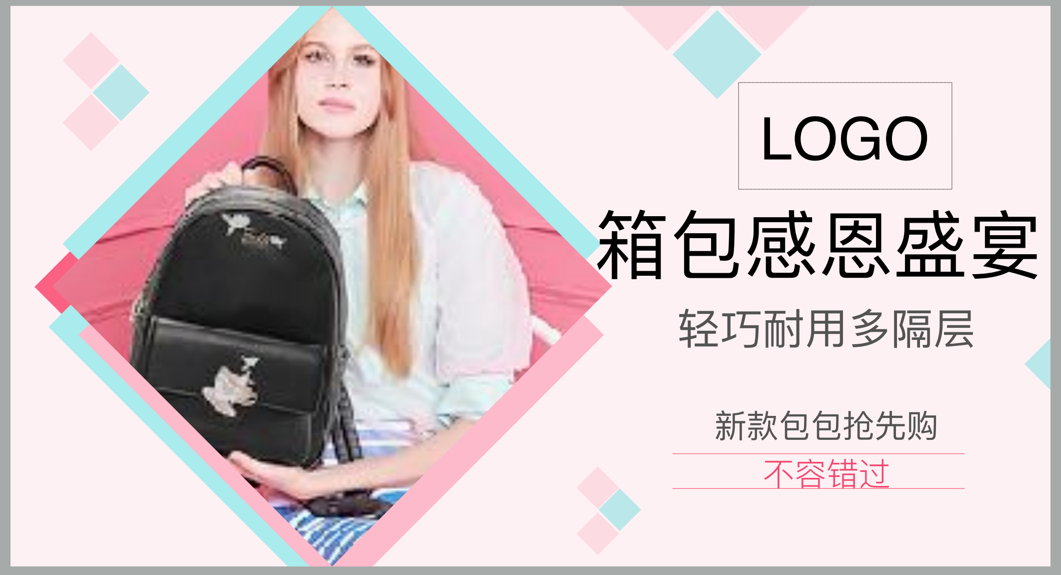
AVG. RT 10ms
3000 QPS

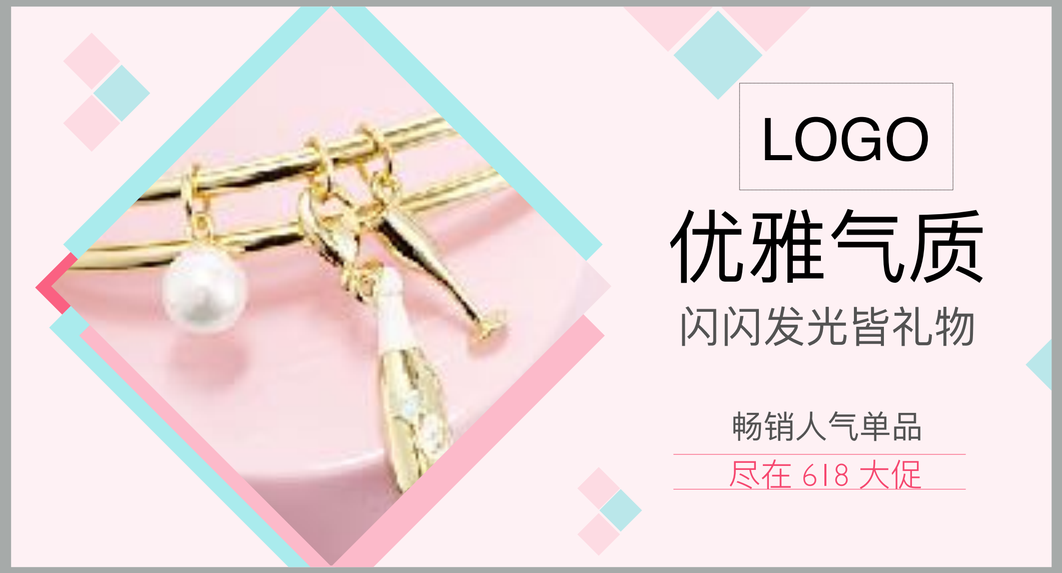
LOGO
清凉大作战
海量美衣零点开抢
特价两件包邮
Time: 2017.6.1-6.7

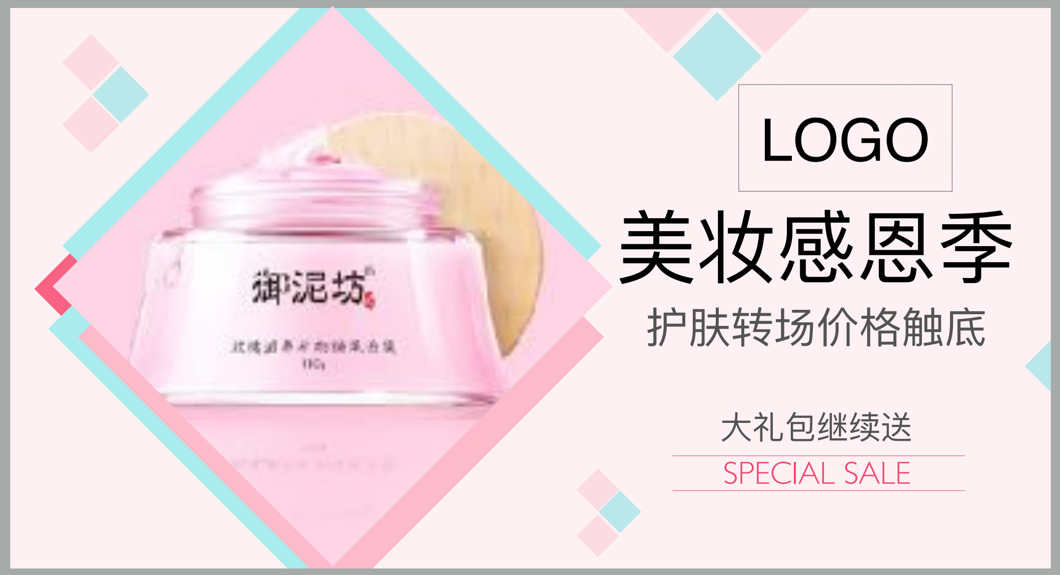
换季美不停	当季新品抢不停手	整点下单力减50	NEW ARRIVAL
美妆感恩季	护肤转场价格触底	大礼包继续送	SPECIAL SALE
美鞋新宠	高跟诱惑释放自己	预售新款立享八折	Time: 2018.6.18
时尚型男	潮搭型格兼具保暖	春季热卖抢鲜购	HOT SALE
休闲舒适	百搭潮鞋搭配无限	一件包邮	立即购买
箱包感恩盛宴	轻巧耐用多隔层	新款包包抢先购	不容错过
优雅气质	闪闪发光皆礼物	畅销人气单品	尽在 618 大促
萌衣特卖惠	百变萌搭库存有限	送好孩子漂亮衣服	马上收藏

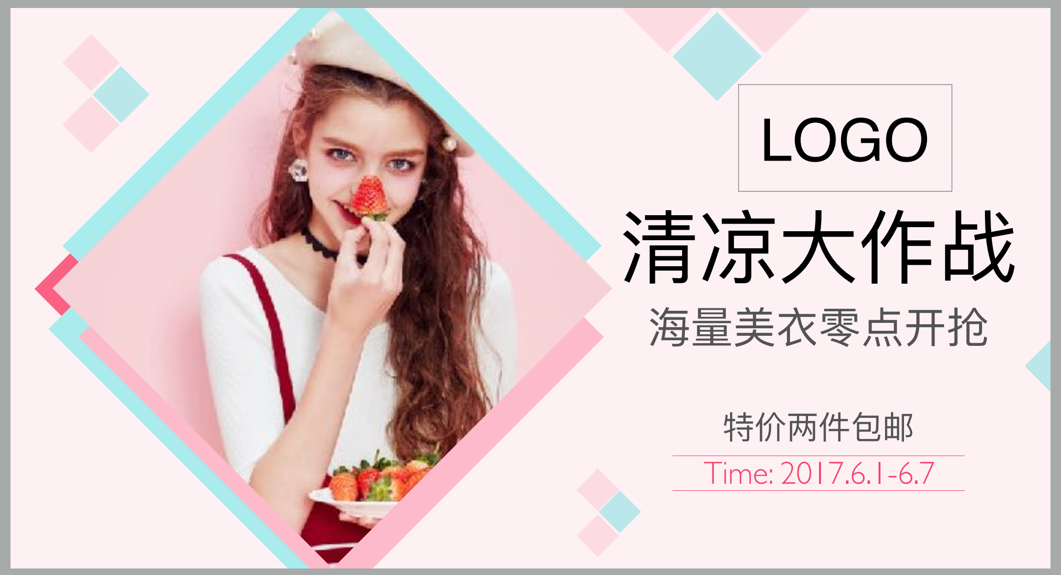
PART 2: MINOLTA

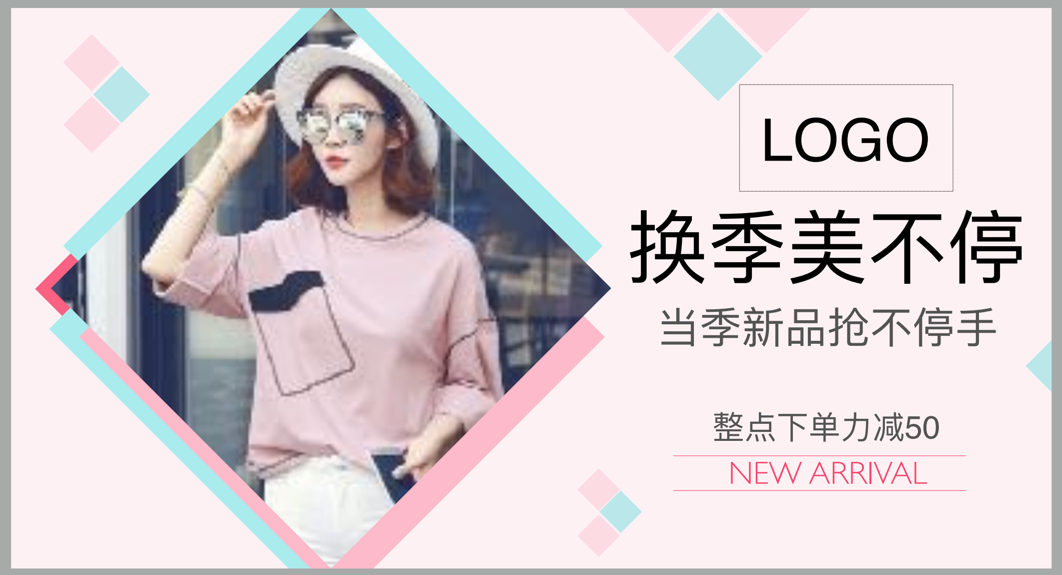


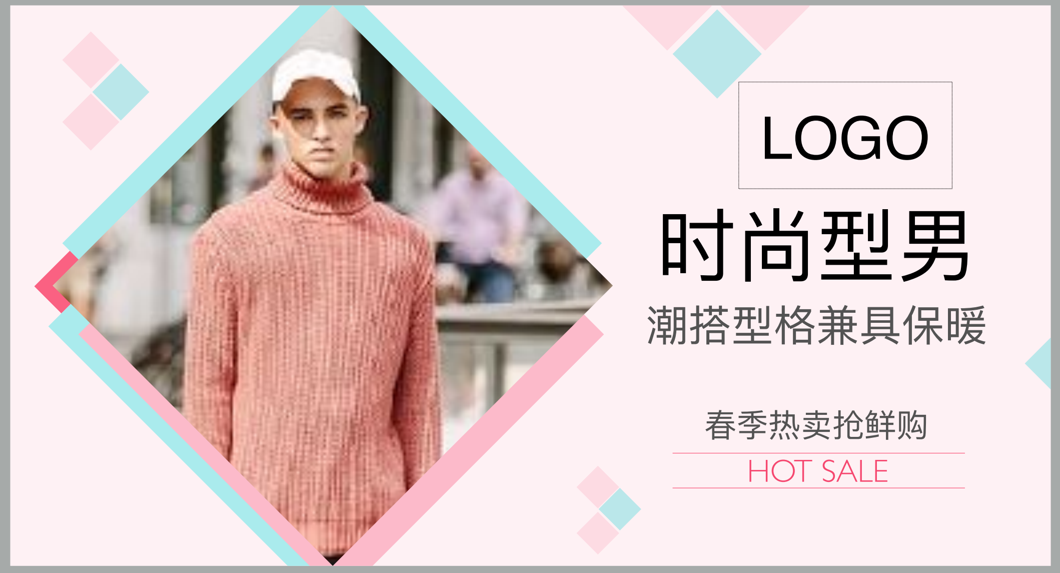


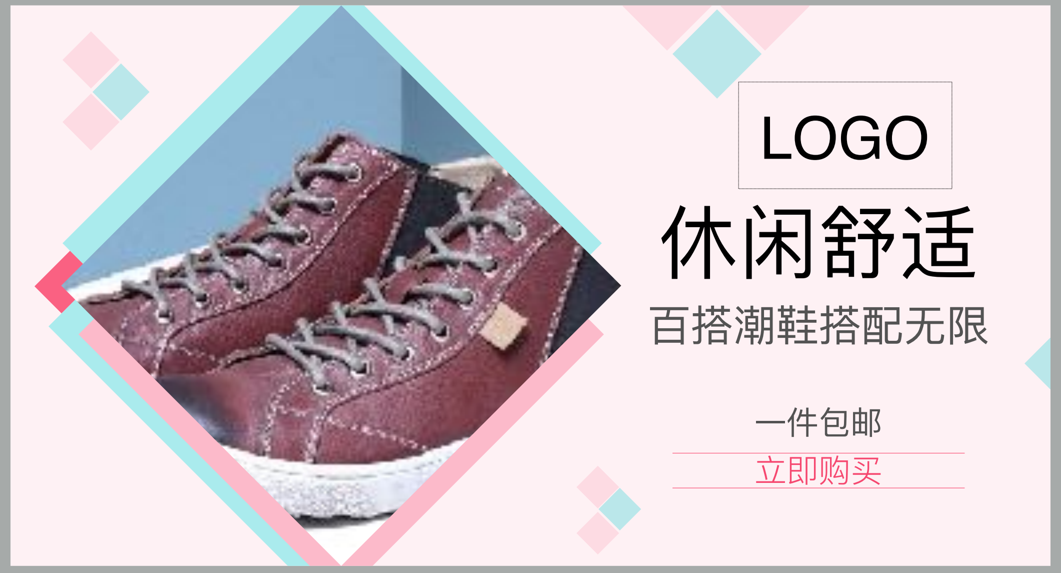


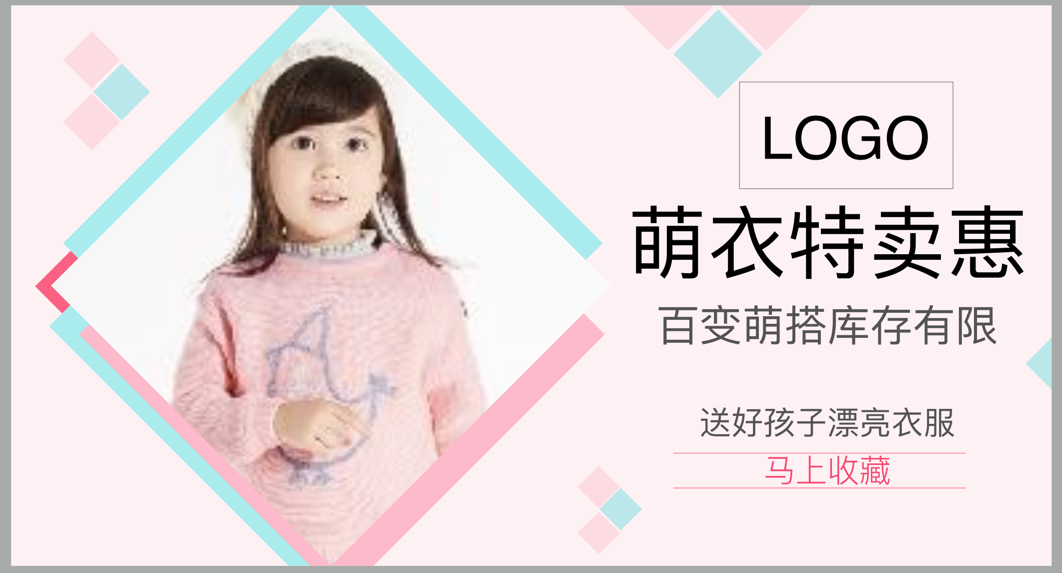










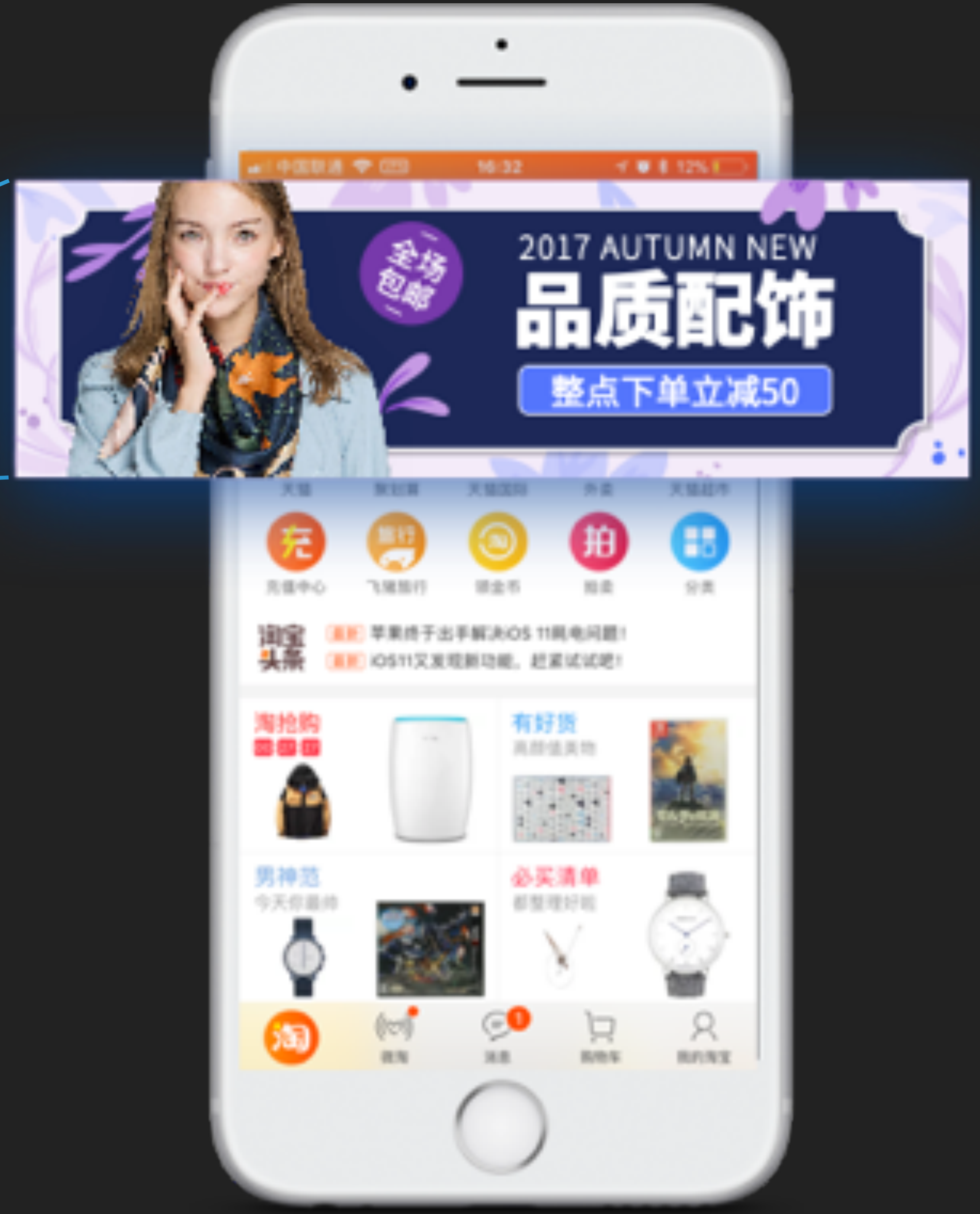


NODE

640x200

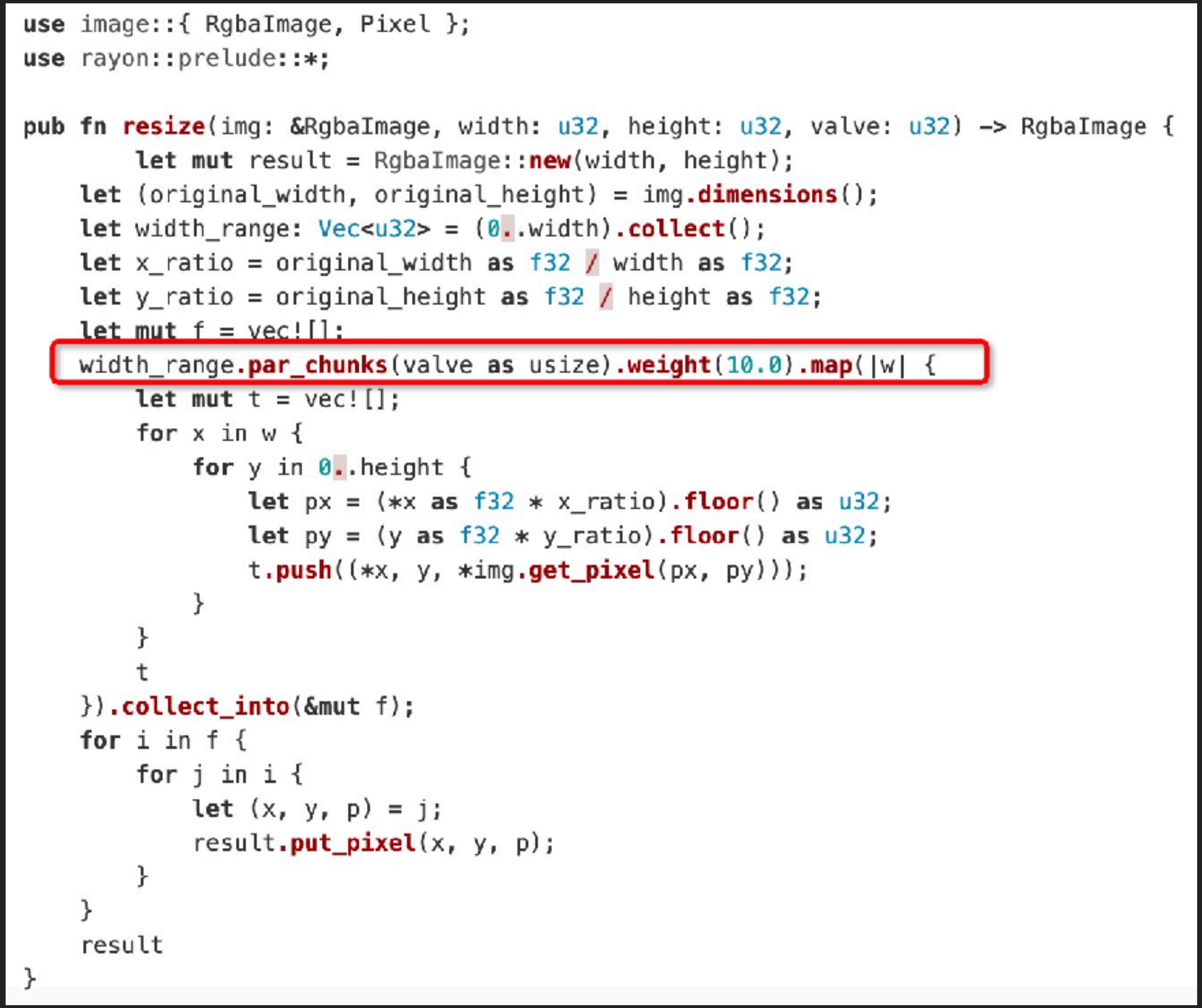
QPS: 60

RT: 200ms



NODE & NEON & RAYON

- Image resize in parallel
- Pixels blend in parallel
- GM(ImageMagick) commands



```rust
use image::{ RgbaImage, Pixel };
use rayon::prelude::*;

pub fn resize(img: &RgbaImage, width: u32, height: u32, valve: u32) -> RgbaImage {
        let mut result = RgbaImage::new(width, height);
    let (original_width, original_height) = img.dimensions();
    let width_range: Vec<u32> = (0..width).collect();
    let x_ratio = original_width as f32 / width as f32;
    let y_ratio = original_height as f32 / height as f32;
    let mut f = vec![];
    width_range.par_chunks(valve as usize).weight(10.0).map(|w| {
        let mut t = vec![];
        for x in w {
            for y in 0..height {
                let px = (*x as f32 * x_ratio).floor() as u32;
                let py = (y as f32 * y_ratio).floor() as u32;
                t.push((*x, y, *img.get_pixel(px, py)));
            }
        }
        t
    }).collect_into(&mut f);
    for i in f {
        for j in i {
            let (x, y, p) = j;
            result.put_pixel(x, y, p);
        }
    }
    result
}
```

NODE & NEON & RAYON

640x200	640x200
QPS: 60	QPS: 60
RT: 200ms	RT: 20ms

RUST

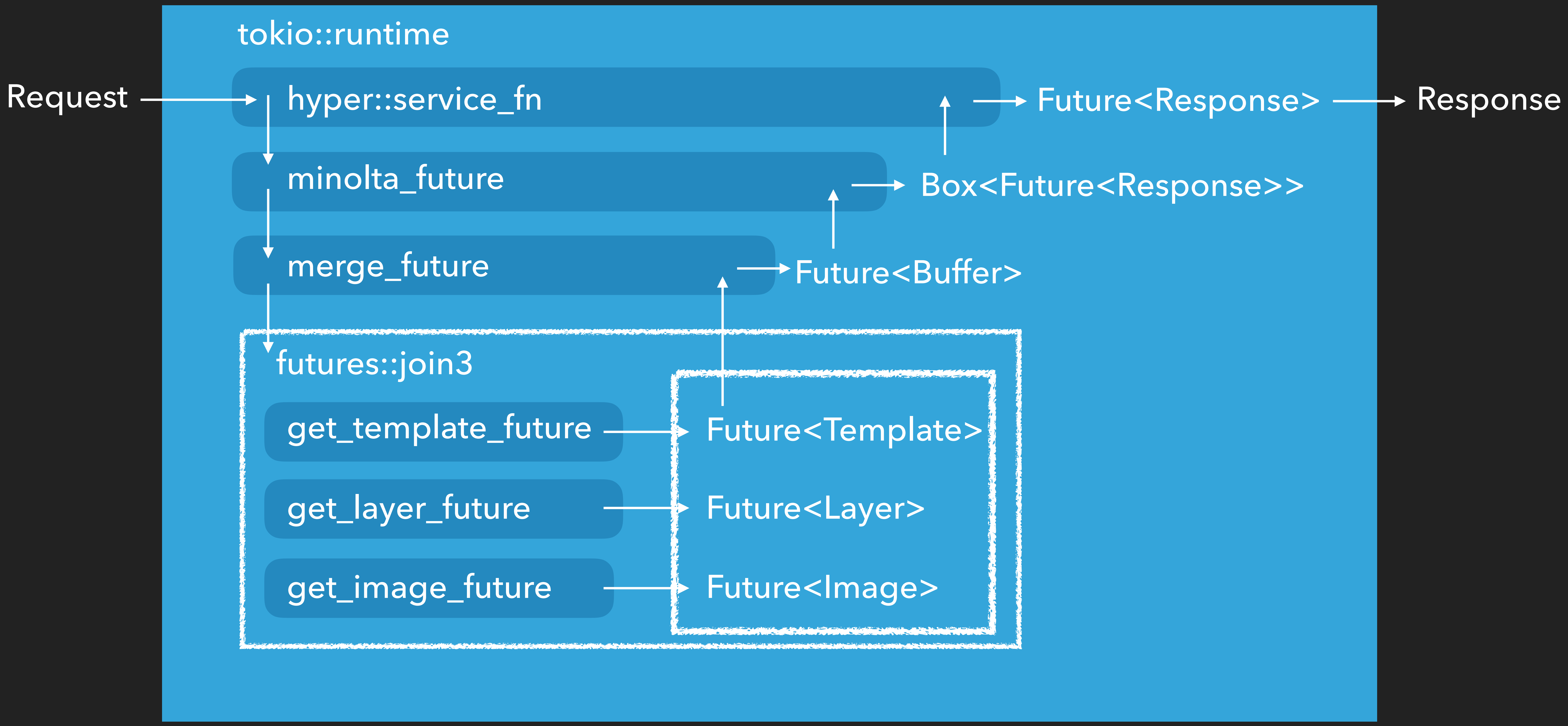
- Asynchronous everything
- Cache everywhere

ASYNCHRONOUS

“Async everything. By leveraging the Tokio project, all Gotham web framework types are async out of the box. Our async story is further enhanced by Hyper, a fast server that provides an elegant layer over stringly typed HTTP.”

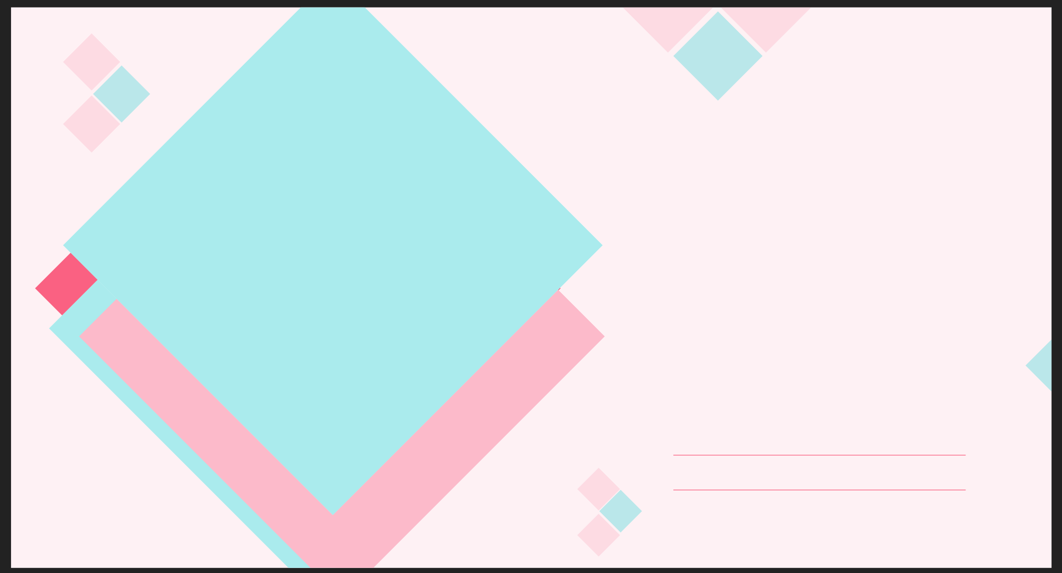
––Gotham https://github.com/gotham-rs/gotham

ASYNCHRONOUS



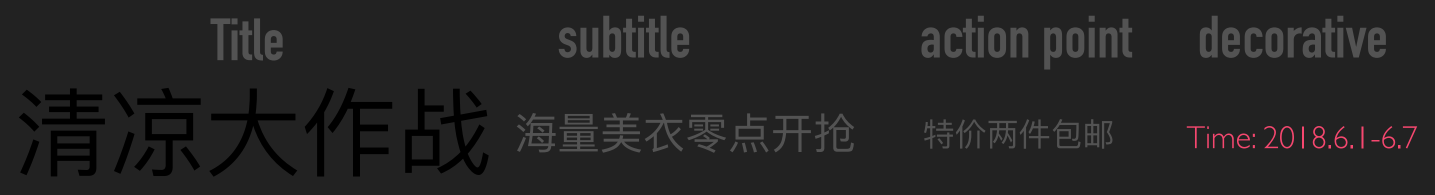
CACHE EVERYWHERE

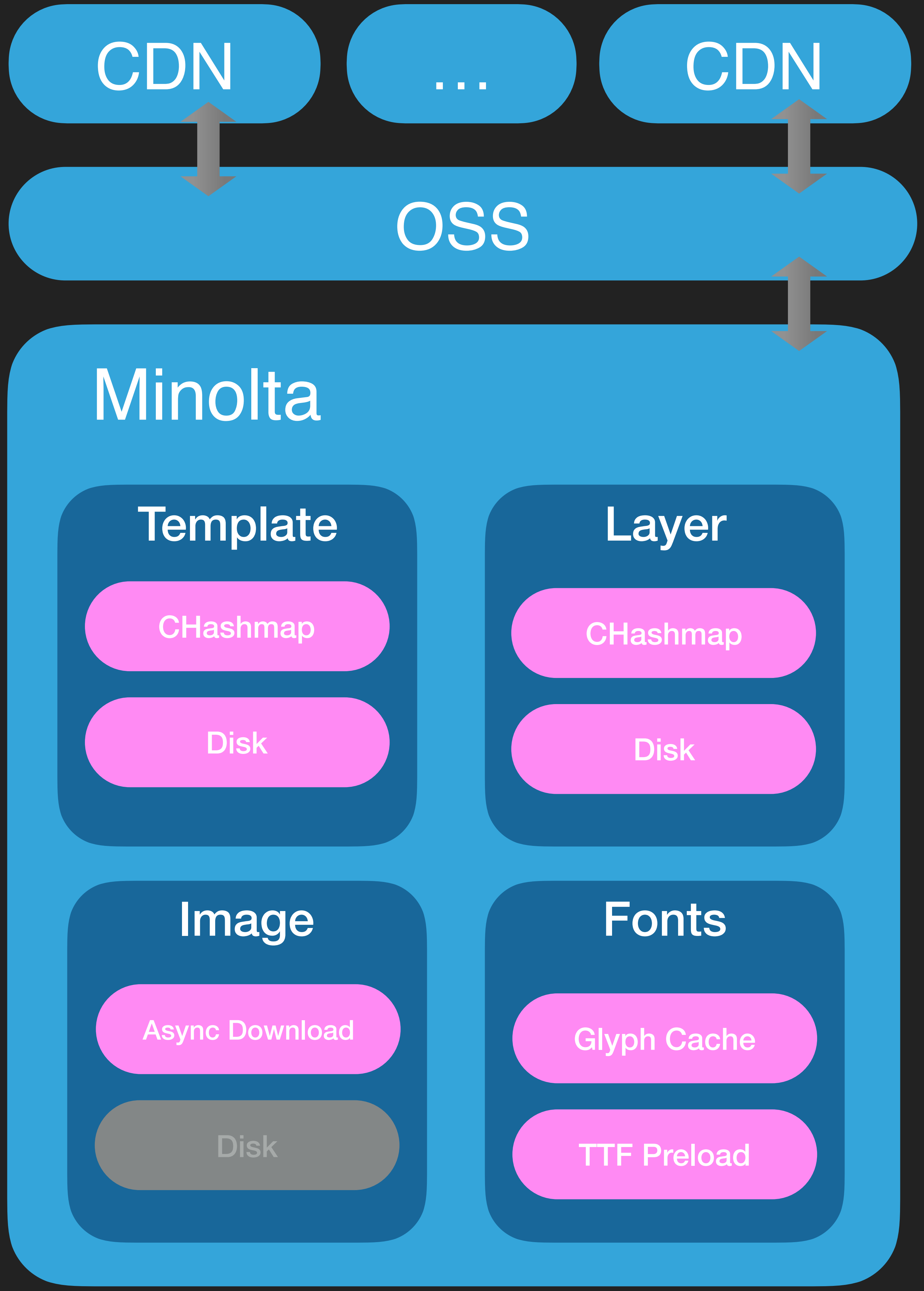
Layer



Image

Copy

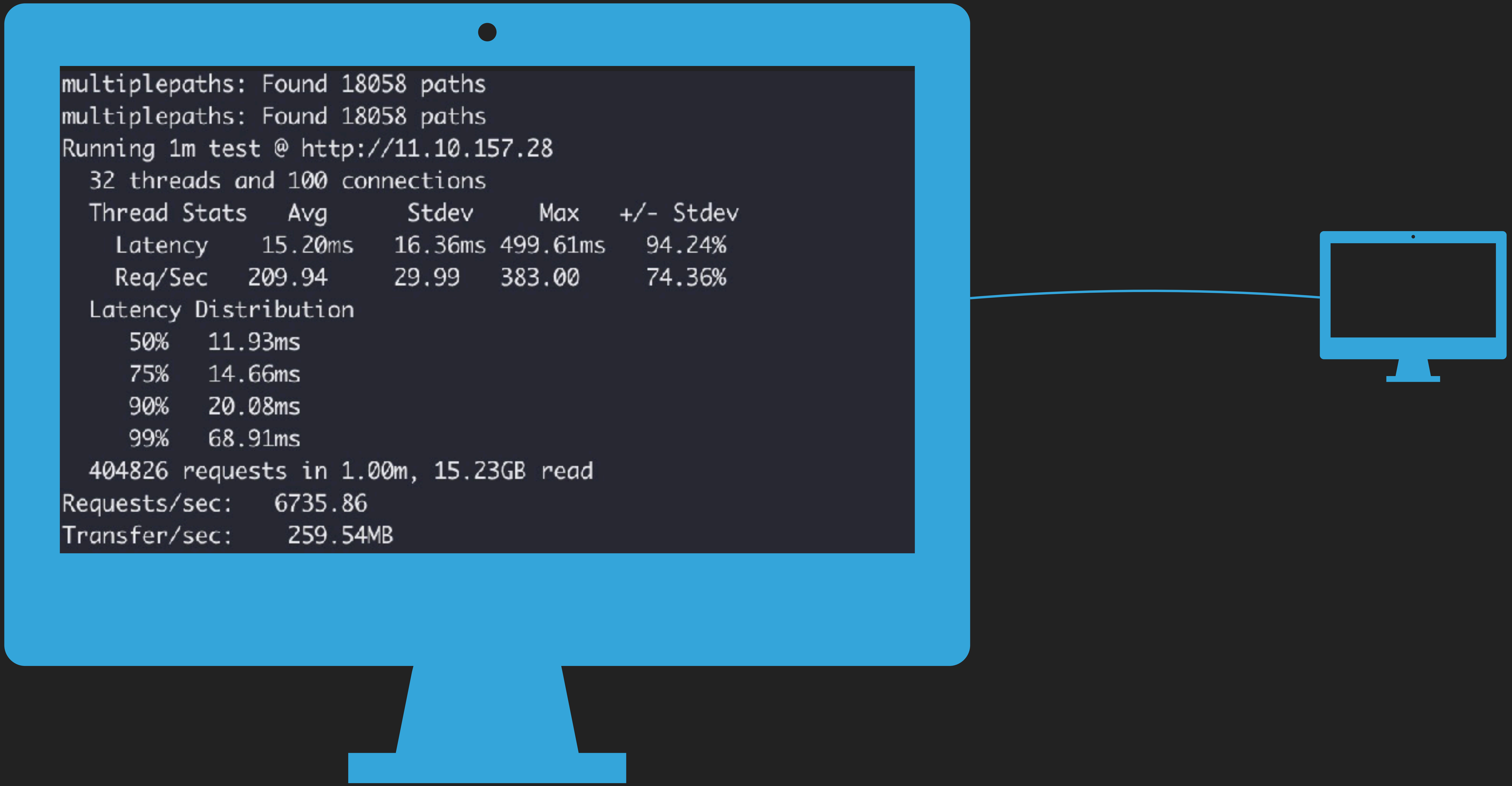




RUST

640x200	640x200	640x200
QPS: 60	QPS: 60	QPS: 3000
RT: 200ms	RT: 20ms	RT: 10ms

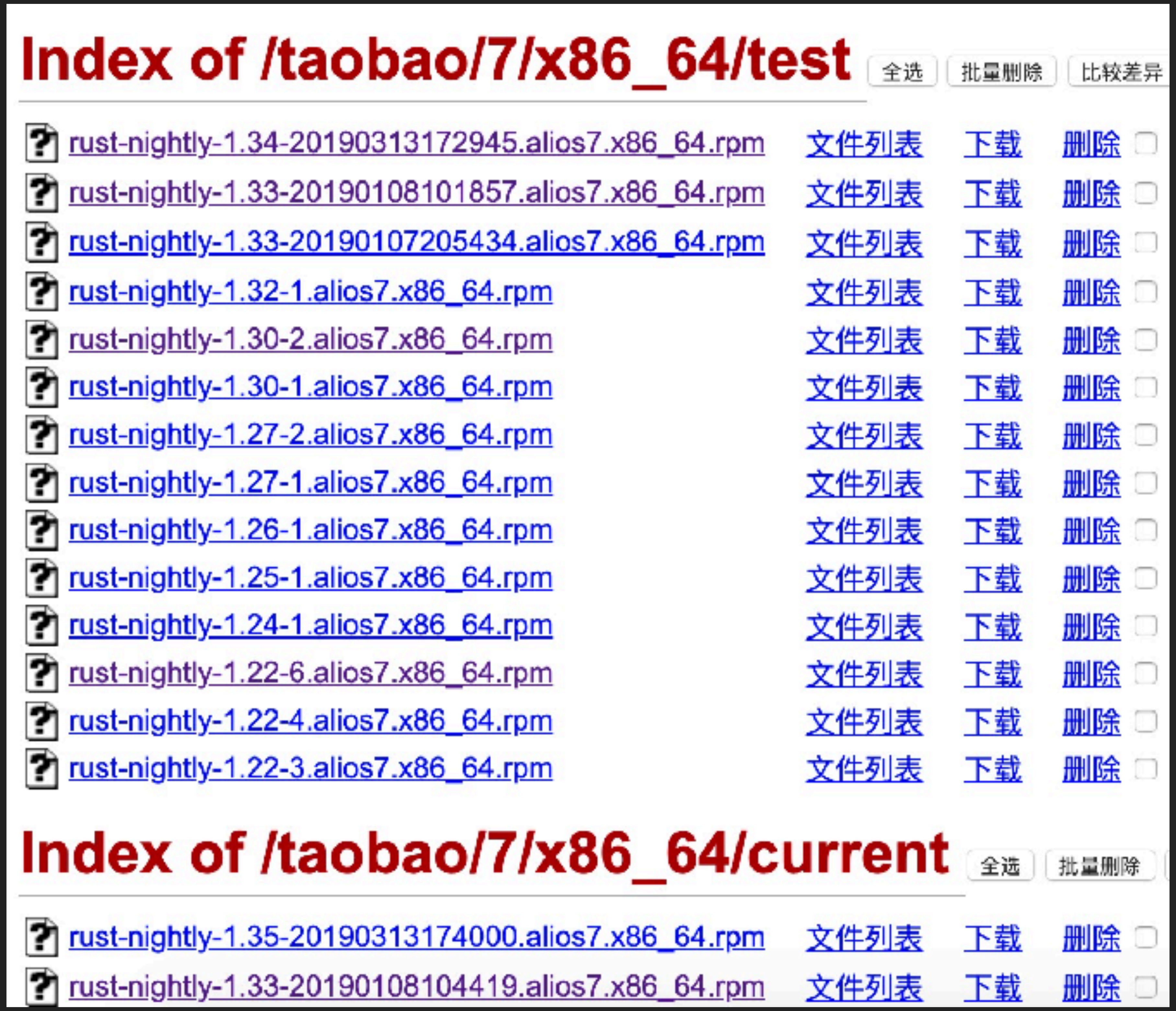
BENCHMARK & DIFF



```
multiplepaths: Found 18058 paths
multiplepaths: Found 18058 paths
Running 1m test @ http://11.10.157.28
  32 threads and 100 connections
  Thread Stats   Avg      Stdev     Max   +/- Stdev
    Latency    15.20ms   16.36ms 499.61ms   94.24%
    Req/Sec   209.94     29.99    383.00     74.36%
  Latency Distribution
     50%   11.93ms
     75%   14.66ms
     90%   20.08ms
     99%   68.91ms
  404826 requests in 1.00m, 15.23GB read
Requests/sec:   6735.86
Transfer/sec:    259.54MB
```

DEPLOY

- Rust-nightly RPM Package



Index of /taobao/7/x86_64/test

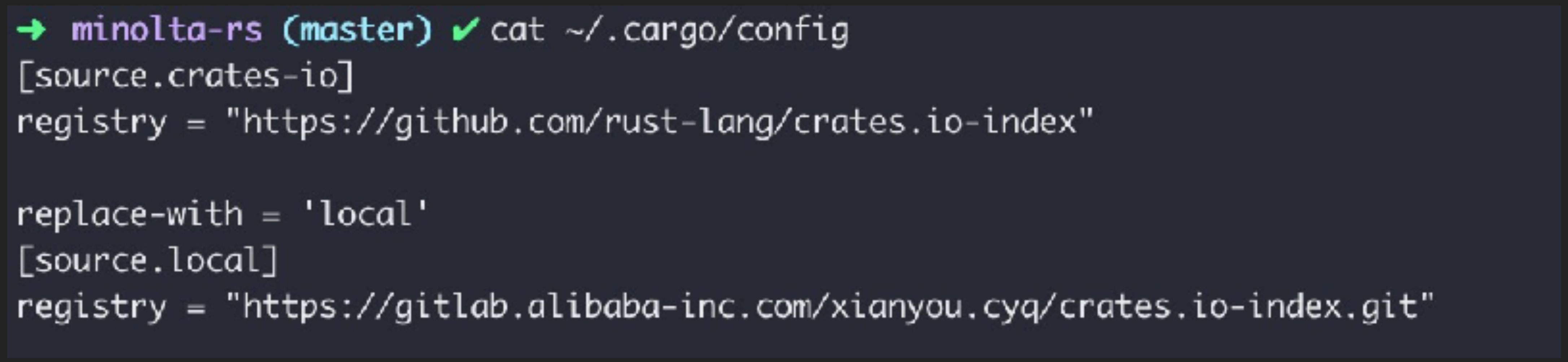
rust-nightly-1.34-20190313172945.alios7.x86_64.rpm	文件列表	下载	删除
rust-nightly-1.33-20190108101857.alios7.x86_64.rpm	文件列表	下载	删除
rust-nightly-1.33-20190107205434.alios7.x86_64.rpm	文件列表	下载	删除
rust-nightly-1.32-1.alios7.x86_64.rpm	文件列表	下载	删除
rust-nightly-1.30-2.alios7.x86_64.rpm	文件列表	下载	删除
rust-nightly-1.30-1.alios7.x86_64.rpm	文件列表	下载	删除
rust-nightly-1.27-2.alios7.x86_64.rpm	文件列表	下载	删除
rust-nightly-1.27-1.alios7.x86_64.rpm	文件列表	下载	删除
rust-nightly-1.26-1.alios7.x86_64.rpm	文件列表	下载	删除
rust-nightly-1.25-1.alios7.x86_64.rpm	文件列表	下载	删除
rust-nightly-1.24-1.alios7.x86_64.rpm	文件列表	下载	删除
rust-nightly-1.22-6.alios7.x86_64.rpm	文件列表	下载	删除
rust-nightly-1.22-4.alios7.x86_64.rpm	文件列表	下载	删除
rust-nightly-1.22-3.alios7.x86_64.rpm	文件列表	下载	删除

Index of /taobao/7/x86_64/current

rust-nightly-1.35-20190313174000.alios7.x86_64.rpm	文件列表	下载	删除
rust-nightly-1.33-20190108104419.alios7.x86_64.rpm	文件列表	下载	删除

DEPLOY

- Crates mirror
  - Sync crates.index and crates
  - Create crates file server
  - Configuration



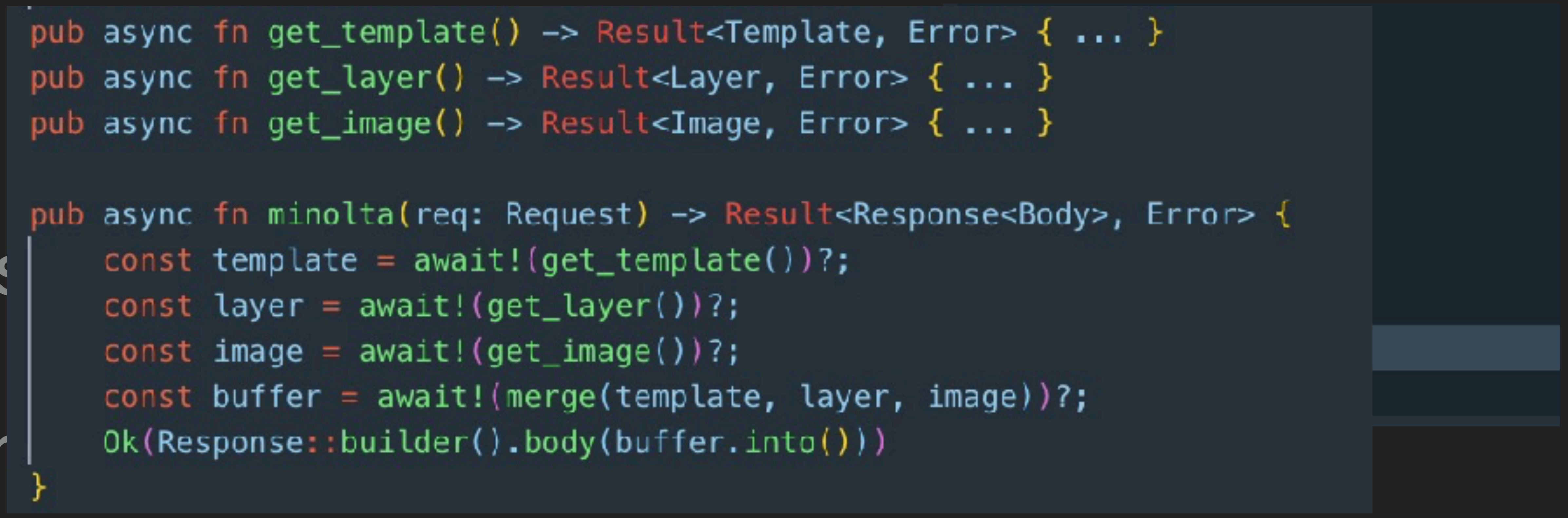
```
➜ minolta-rs (master) ✔ cat ~/.cargo/config
[source.crates-io]
registry = "https://github.com/rust-lang/crates.io-index"

replace-with = 'local'
[source.local]
registry = "https://gitlab.alibaba-inc.com/xianyou.cyq/crates.io-index.git"
```

MONITORS

- Warning: QPS
- Warning: RT
- Critical: CPU usage
- Critical: MEM usage
- Critical: Disk usage
- Urgent: Process

- Future is difficult, but async/await is coming soon
- Deploy
- The ecos
  - concur
  - oss-rust-sdk

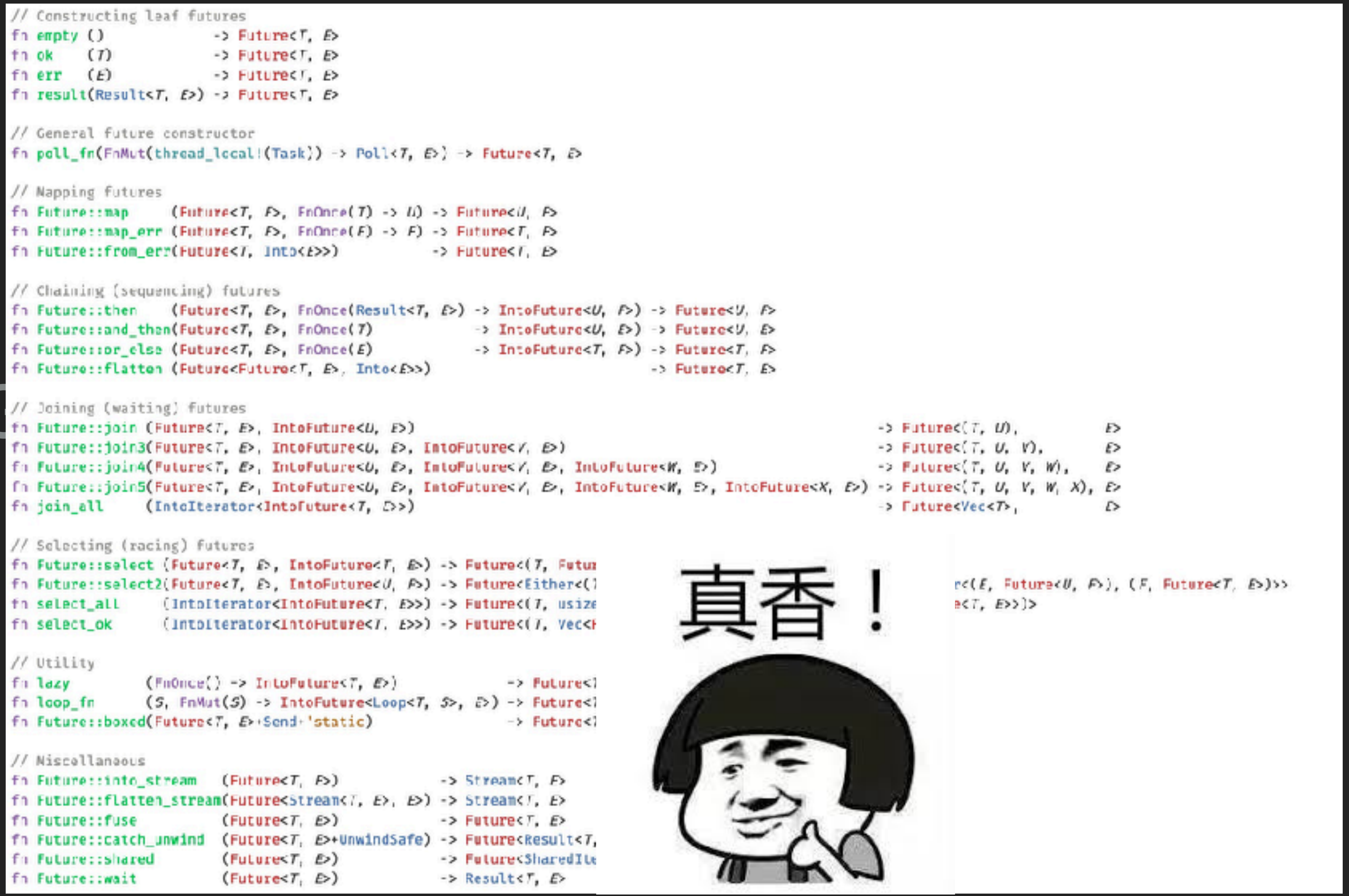


```rust
pub async fn get_template() -> Result<Template, Error> { ... }
pub async fn get_layer() -> Result<Layer, Error> { ... }
pub async fn get_image() -> Result<Image, Error> { ... }

pub async fn minolta(req: Request) -> Result<Response<Body>, Error> {
    const template = await!(get_template())?;
    const layer = await!(get_layer())?;
    const image = await!(get_image())?;
    const buffer = await!(merge(template, layer, image))?;
    Ok(Response::builder().body(buffer.into()))
}
```

TIPS

- Error handling
- Futures cheatsheet
- Use Node.js/Python/C
- Be patient, rust is diffi



```
// Constructing leaf futures
fn empty ()               -> Future<T, E>
fn ok    (T)              -> Future<T, E>
fn err   (E)              -> Future<T, E>
fn result(Result<T, E>)   -> Future<T, E>

// General future constructor
fn poll_fn(FnMut(thread_local!(Task)) -> Poll<T, E>) -> Future<T, E>

// Mapping futures
fn Future::map      (Future<T, E>, FnOnce(T) -> U) -> Future<U, E>
fn Future::map_err  (Future<T, E>, FnOnce(E) -> F) -> Future<T, F>
fn Future::from_err(Future<T, Into<E>>)            -> Future<T, E>

// Chaining (sequencing) futures
fn Future::then     (Future<T, E>, FnOnce(Result<T, E>) -> IntoFuture<U, F>) -> Future<U, F>
fn Future::and_then(Future<T, E>, FnOnce(T)            -> IntoFuture<U, E>) -> Future<U, E>
fn Future::or_else  (Future<T, E>, FnOnce(E)            -> IntoFuture<T, F>) -> Future<T, F>
fn Future::flatten  (Future<Future<T, E>, Into<E>>)                          -> Future<T, E>

// Joining (waiting) futures
fn Future::join (Future<T, E>, IntoFuture<U, E>)                                                  -> Future<(T, U),          E>
fn Future::join3(Future<T, E>, IntoFuture<U, E>, IntoFuture<V, E>)                                -> Future<(T, U, V),       E>
fn Future::join4(Future<T, E>, IntoFuture<U, E>, IntoFuture<V, E>, IntoFuture<W, E>)              -> Future<(T, U, V, W),    E>
fn Future::join5(Future<T, E>, IntoFuture<U, E>, IntoFuture<V, E>, IntoFuture<W, E>, IntoFuture<X, E>) -> Future<(T, U, V, W, X), E>
fn join_all     (IntoIterator<IntoFuture<T, E>>)                                                  -> Future<Vec<T>,          E>

// Selecting (racing) futures
fn Future::select (Future<T, E>, IntoFuture<T, E>) -> Future<(T, Futur
fn Future::select2(Future<T, E>, IntoFuture<U, F>) -> Future<Either<(T                r<(E, Future<U, F>), (F, Future<T, E>)>>
fn select_all     (IntoIterator<IntoFuture<T, E>>) -> Future<(T, usize                e<T, E>>)>
fn select_ok      (IntoIterator<IntoFuture<T, E>>) -> Future<(T, Vec<F

// Utility
fn lazy        (FnOnce() -> IntoFuture<T, E>)              -> Future<T
fn loop_fn     (S, FnMut(S) -> IntoFuture<Loop<T, S>, E>) -> Future<T
fn Future::boxed(Future<T, E>+Send+'static)                -> Future<T

// Miscellaneous
fn Future::into_stream    (Future<T, E>)                 -> Stream<T, E>
fn Future::flatten_stream(Future<Stream<T, E>, E>)       -> Stream<T, E>
fn Future::fuse           (Future<T, E>)                 -> Future<T, E>
fn Future::catch_unwind   (Future<T, E>+UnwindSafe)      -> Future<Result<T,
fn Future::shared         (Future<T, E>)                 -> Future<SharedIte
fn Future::wait           (Future<T, E>)                 -> Result<T, E>
```

Thank you! 阿里妈妈 · 创意中心