Ruby と Go でゼロから作る証券システム：実際に開発してみて良かった点と辛かった点、そこから得た学び

2025.11.06(Thu.)

RubyWorld Conference 2025 Day1

小林悟史 (noel)

bloomo

ブルーモ証券株式会社

Partner with RubyStackNews

Independent Ruby & Rails publication for senior developers

Why RubyStackNews?

- Focused on Ruby and Ruby on Rails
- Long-form articles based on real conference talks
- Audience of senior developers and tech leads
- Readers from the US, Europe, and Asia

RubyStackNews turns conference talks and real-world experience into practical, production-focused technical articles.

Partnerships & Sponsorships

- Article sponsorships
- Inline placements inside articles
- Sidebar visibility

View partnership details

rubystacknews.com

Partnership info

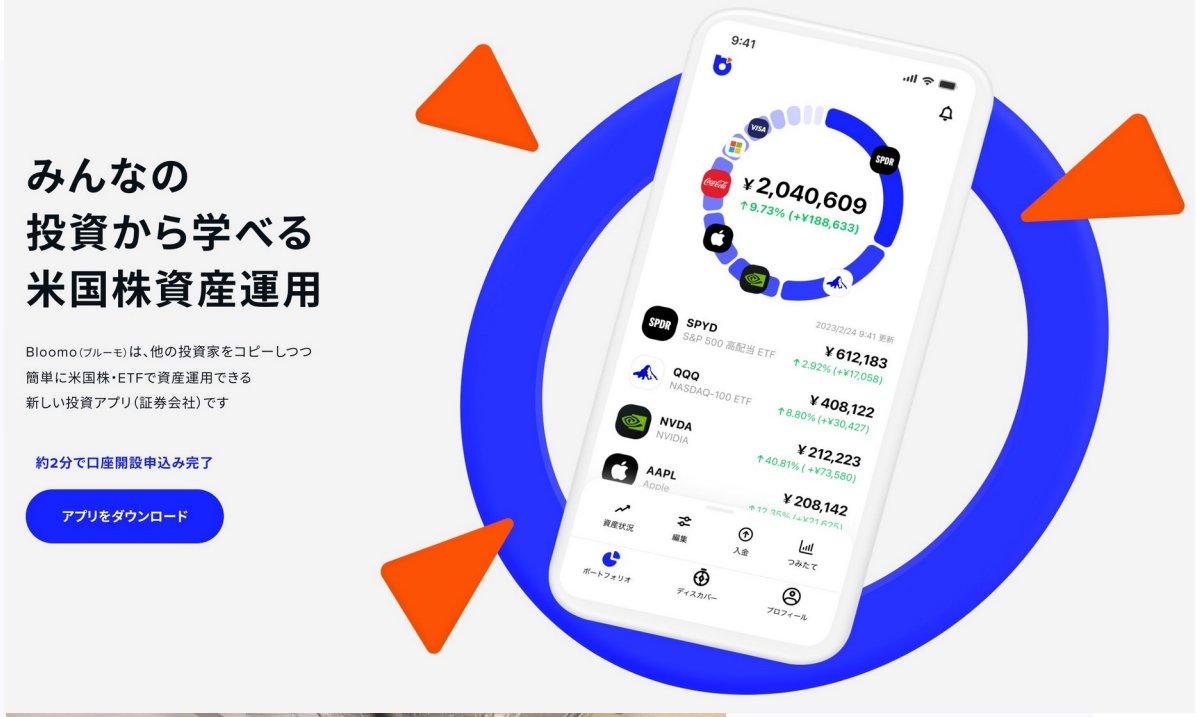
- 小林 悟史（小林 ノエル）
  @free_world21
- ブルーモ証券株式会社 取締役 CTO
- 2009 年度 未踏事業採択

- Omotesando.rb, Roppongi.rb, Shinjuku.rb とかによくいます

- 旅行・世界のコワーキングスペースめぐり（ワーケーション的な何か）が好き
- **趣味で**【政治資金データベース】を開発してます

好きなバンド

- L’Arc~en~Ciel



Index

- 対象者：高信頼領域のシステムをどのように作り上げるかに興味がある人
- 持ち帰れる知見
  - 複数の言語を採用したときのシステム境界を引くポイント
  - 発生する技術的な問題に対しての工夫ポイント
  - 証券システムもゼロから作れるということをイメージを持ってもらえる
- 発表資料は後ほど公開します

bloomo

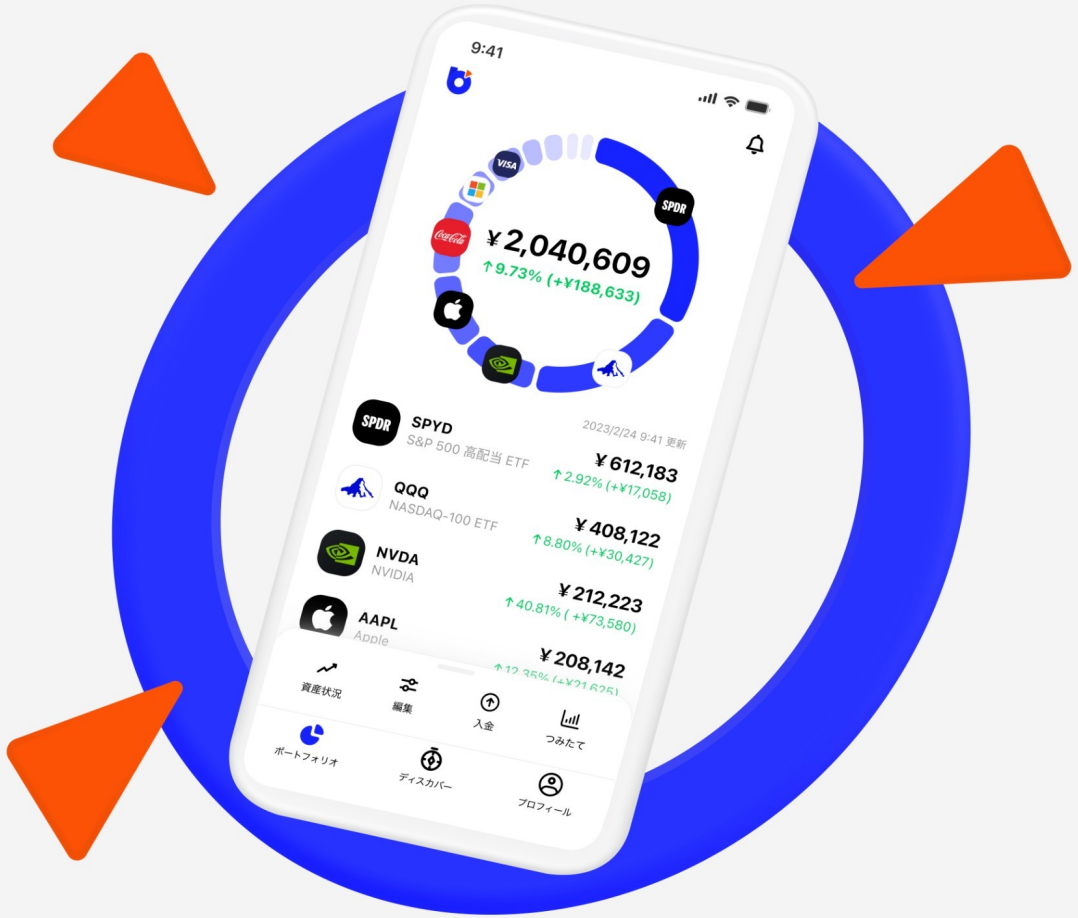
米国株資産運用アプリ Bloomo を提供中!

みんなの
投資から学べる
米国株資産運用

Bloomo（ブルーモ）は、他の投資家をコピーしつつ
簡単に米国株・ETFで資産運用できる
新しい投資アプリ（証券会社）です

約2分で口座開設申込み完了

アプリをダウンロード



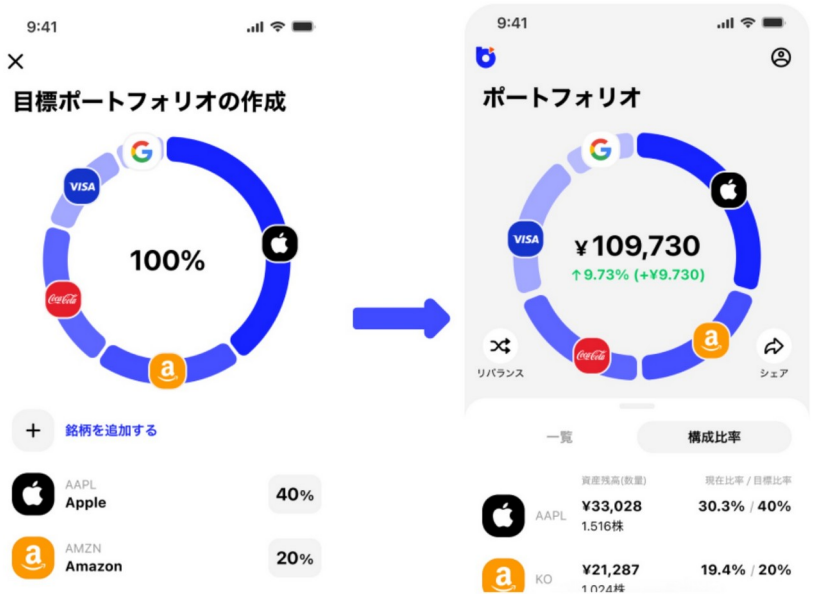
コピペで「バフェット投資」 スマホ完結で若者も気軽に
YOUTH FINANCE①

bloomo

ポートフォリオ機能で、高度な資産運用のハードルを下げている

ポートフォリオ投資機能

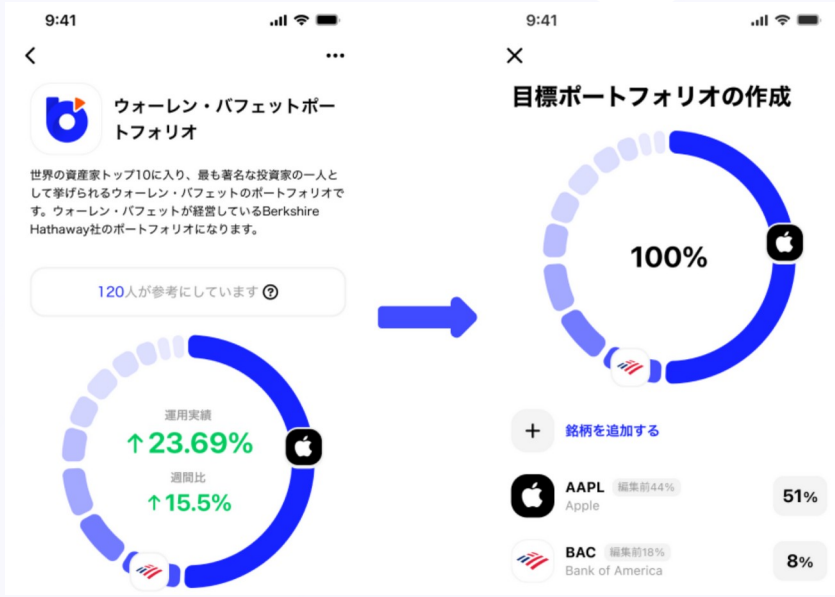
米国株・ ETF で理想のポートフォリオを作成したら、
両替や買付はブルーモが自動執行してくれる。



複数銘柄への分散投資が手間なく実現できる
（ユーザーの保有銘柄数は 10 以上（日本平均の 3 倍程度））

共有・コピー機能

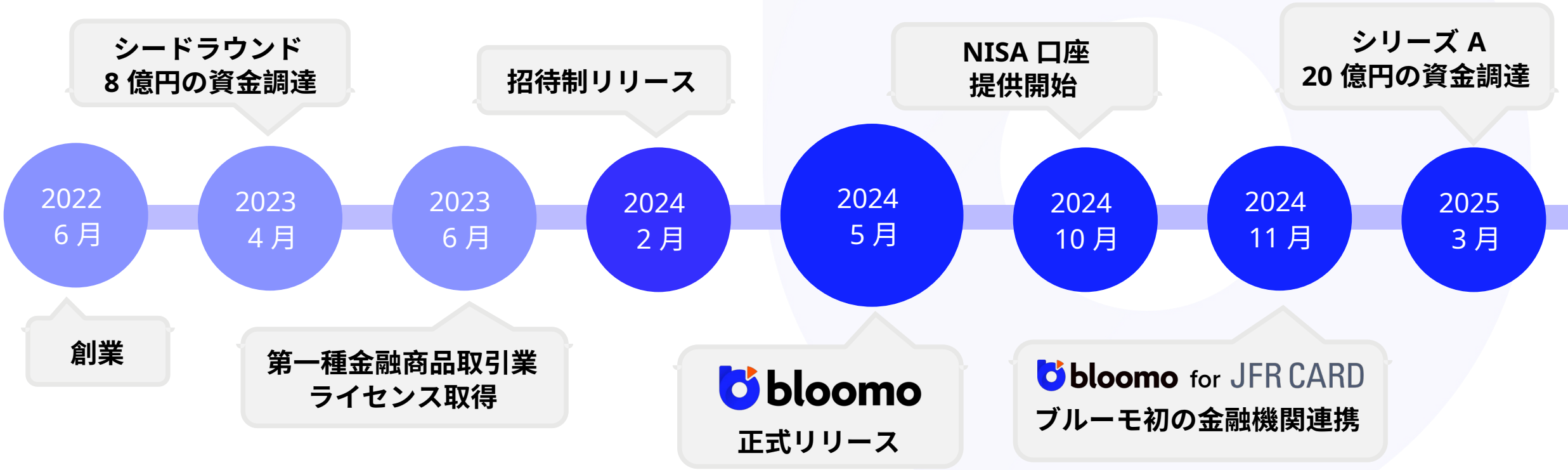
専門家や他のユーザーのポートフォリオを見て、
ワンタップでコピーできる。



初心者でもポートフォリオ作成が可能に
（ユーザーの 9 割以上がコピーから開始）

6 年ぶりの証券会社スタートアップとして立ち上がった

個別株を取扱う証券会社スタートアップとしては、 Finatext ・ FOLIO 以来の存在。史上最速ペースで金商 1 種（証券会社）ライセンス取得・プロダクトリリースを続けてきた。



実は
Ruby コミュニティーの勉強会がきっかけで
生まれた会社です

bloomo

Attended 52 | Organize 2 | Presenter 3 | Bookmark 0

2019 05/29 (Wed) 19:30〜 — Attending 出席済み — Ended
Ebisu.rb
Ebisu.rb #23
iTume 他
東京都渋谷区恵比寿南3-1-1
19/30

2019 05/24 (Fri) 19:00〜 — Attending — Ended
Roppongi.rb
Roppongi.rb #10 "夜のLT会"
totzyuta 他
東京都港区六本木3-2-1 (住友不動産六
26/36

2019 05/09 (Thu) 19:30〜 — Attending — Ended
Omotesando.rb
表参道.rb #46 〜After RubyKaigi 〜
sinsoku 他
東京都渋谷区恵比寿南3-1-1
28/36

2018 11/08 (Thu) 19:30〜 — Cancel — Ended
Omotesando.rb
表参道.rb #40 〜Railsアップグレード〜
sinsoku 他
東京都渋谷区恵比寿南3-1-1
34/33

2018 10/31 (Wed) 19:30〜 — Attending 出席済み — Ended
Ebisu.rb
Ebisu.rb #19
iTume 他
東京都渋谷区恵比寿4-20-3 (恵比寿ガー
18/28

2018 10/29 (Mon) 19:30〜 — Cancel — Ended
Meguro.rb
Meguro.rb#20 2018/10/29(Mon.) at Viibar
south37 他
東京都品川区上大崎2-13-17(目黒東急
19/30

2018 10/24 (Wed) 19:30〜 — Attending 出席済み — Ended
Shinjuku.rb
Shinjuku.rb #66 プロダクトの品質担保
treby 他
東京都渋谷区代々木1-36-4 (全理連ビル
17/20

2019 11/15 (Fri) 19:00〜 — Attending — Ended
銀座Rails
銀座Rails#15 @リンクアンドモチベーション
ginkouno 他
オンライン
126/140

2019 10/24 (Thu) 19:00〜 — Cancel — Ended
銀座Rails
銀座Rails#14 @リンクアンドモチベーション
ginkouno 他
オンライン
75/130

2019 09/12 (Thu) 19:00〜 — Cancel — Ended
銀座Rails
銀座Rails#13 @リンクアンドモチベーション
ginkouno 他
オンライン
136/150

2019 08/29 (Thu) 19:00〜 — Waitlist — Ended
銀座Rails
銀座Rails#12@DeNA
ginkouno 他
オンライン
170/115

2019 06/20 (Thu) 19:00〜 — Attending — Ended
Roppongi.rb
Roppongi.rb #11 "夜のLT会"
totzyuta 他
東京都港区六本木4-1-4(黒崎ビル4階)
17/35

2019 06/06 (Thu) 19:30〜 — Attending — Ended
Omotesando.rb
表参道.rb #47 〜API〜
sinsoku 他
東京都渋谷区神宮前5-52-2
37/43

bloomo

金融機関の（システムの）

2019.12.13 銀座Rails#16
@free_world21

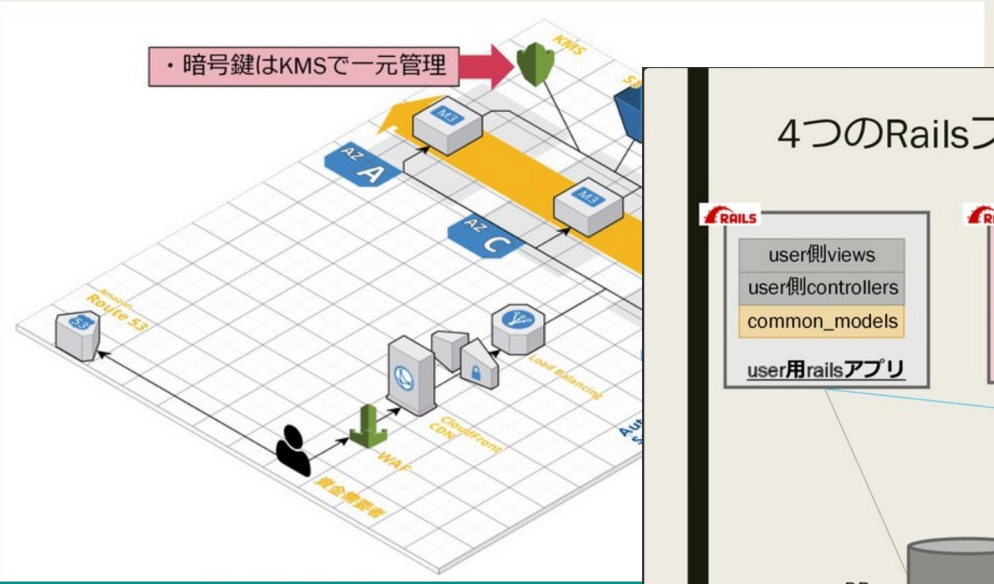
金融業は規制産業

- 免許制
  - 法律等に書かれている要件を満たし
  - 銀行業（銀行法）
  - 保険業（保険業法）

ものすごく雑にまとめます

- 機能要件的な話
  - 特定の操作や情報を見れる人を絞りましょう（権限管理）
  - 特に重要な情報や操作（例：個人情報の閲覧）をする時
    - 申請・承認のようなプロセスを構築しましょう
    - その履歴もちゃんと残しましょう
  - 重要な情報は暗号化をなどをして、鍵の管理を徹底しましょう
  - ょう（OTPなど）

インフラ



4つのRailsプロジェクト

- user用railsアプリ

暗号化について（AWS KMS）

- Customer Master Key（CMK）を指定して、data key（新しい暗号鍵）を要求する
  - `CMK has_many :data_keys`
- 以下のものがKMSから返ってくる
  - *A:* 平文の暗号鍵
  - *B:* Aが暗号化されたもの
- Aで暗号化して、それは消去。BをDBなどに保存しておく。
- BをKMSに投げつけると復号化して返してくれる（Aを得られる）ので、データ本体をAで復号化する

https://speakerdeck.com/free_world21/jin-rong-ji-guan-false-sisutemufalse-zuo-rifang

bloomo

- LT 資料は Speaker Deck にあげて LinkedIn にもリンク載せてた
- 2020 年に当時勤めていた会社を退職
- フリーランス 1 本生活に戻りつつ、自分の趣味のサービス開発をしていた
- そんな中ある日、 1 本の DM が Twitter （現 X ）に届く
  - それが、後に一緒に会社を立ち上げ、現在のブルーモ証券代表でもある中村だった
- 会って話して意気投合
- そのまま起業へ・・・

そうしてできたのが【ブルーモ証券】

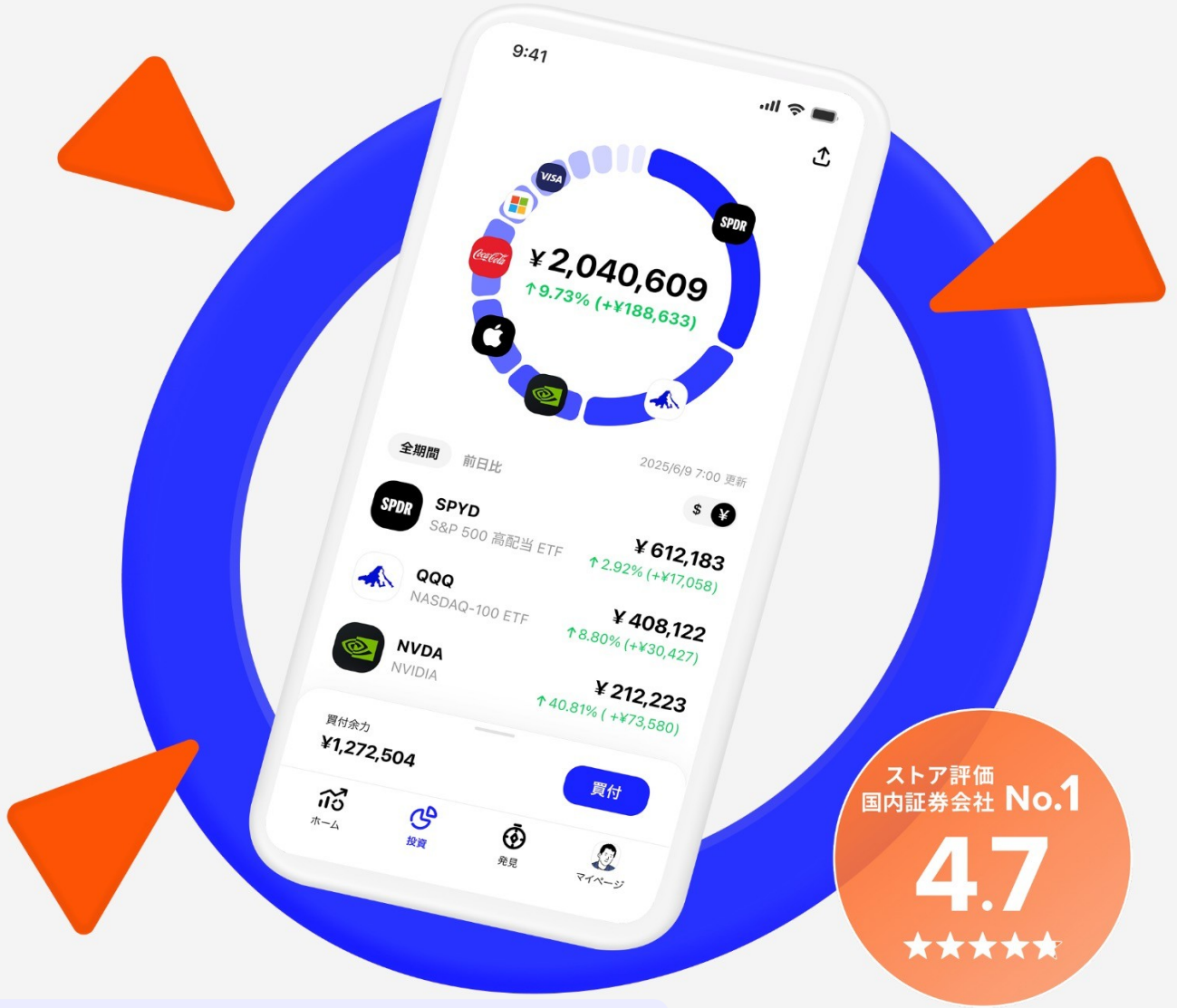
米国株投資で
次の運用を始めよう

国内No.1評価の最先端投資アプリ

約2分で口座開設申込み完了

ブルーモをはじめる

※ 国内No.1評価：2025/3/17現在 当社調べ
（比較対象：App Storeにて提供されている米国株投資アプリ（SBI証券、楽天証券、マネックス証券、松井証券、moomoo証券、ウィブル証券）について、ストア平均評価を比較）



https://bloomo.co.jp/

bloomo

02
初期的な設計とシステム境界: Ruby と Go の役割分担

創業当時からこだわっているポイント

UI/UX に徹底的にこだわりたい

- 金融系のサービスは供給者側の原理（古くからあるシステムの仕様）に引きずられがち
- UI/UX の観点から初期から**ネイティブアプリ**でやりたい

最初から第一種金融商品取引業（証券会社）のライセンス取得を目指す

- 米国株（米国 ETF 含む）専業
- つまりガチの証券会社
  - 小林が前職で経験したのは第一種少額電子募集取扱業務（株式投資型クラウドファンディング）
- **個人情報やお金の取り扱いなど、多くのセキュリティー要件を満たさないといけない**

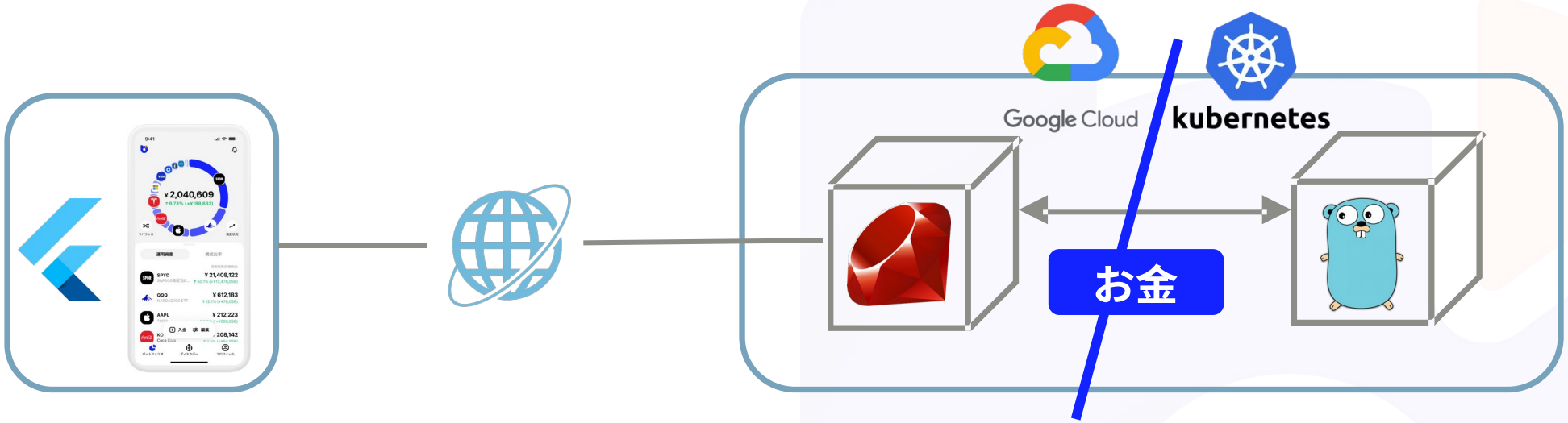
まずは技術選定

- モバイル（ネイティブ）アプリ： Flutter
- クラウドインフラ： Kubernetes (Google Cloud Platform)
- バックエンド：下記のように選択肢が多く、かなり悩んだ

	Ruby	Python		TypeScript	Golang
Framework	Fullstack (Rails)	Fullstack (Django)	microframework (Flask, FastAPI 等)	無限 (Express, Nest 等)	microframework (gin, echo 等)
ORM	ActiveRecord	Django ORM	SQLAlchemy	TypeORM, Prisma 等	自作 , ent
migration	ActiveRecord::Migration	Django migration	alembic	db-migrate 等	golang-migrate, goose
テスト	minitest, rspec	Django test, pytest	pytest	Jest, Jasmine 等	built-in
型サポート	RBS	Type Hints(built-in)		built-in （ただしコンパイルオプション多数）	built-in
勢い	下火?	web 単独では? （基本 AI とセット）		ある	ある
その他	ライブラリ多数、 コミュニティが成熟	同左だが、あくまで AI という context で語られることが多い		TypeScript の native runtime は まだまだ黎明期	なかったらとりあえず自分で作る的な文化が強め

bloomo

バックエンドの結論： Ruby on Rails と Go を併用



<u>お金以外の部分（アプリの API サーバ、管理画面）＝ Ruby on Rails</u>

- ORM や migration 、テストのフレームワークとか個別に選ぶ必要がない：考慮・調査工数を大幅に削減できる
- 認証認可やセキュリティーとかはいったんフレームワークやデファクトのライブラリで対応
- 要件がころころ変わるフロントや管理画面に柔軟に対応できる
- Rails 人材：スタートアップ界隈、プログラミング初学者など人数はそれなりにいる

<u>お金を取り扱う部分＝ Golang</u>

- 硬く作るべき部分を硬い言語で作ってるというイメージ感
- インターネットから直アクセスなし：こまかい認証認可とかセキュリティーとかの考慮・調査工数を減らす
- Golang 人材：メガベンチャーでの採用事例が多いので、メガベンチャー転職組とかの受け皿になれたらいいな

03
Ruby on Rails が担う領域：顧客データ等

Ruby on Rails が担う領域

- スムーズな口座開設機能
  - 本人確認・同意フロー
  - 暗号化と個人情報管理
- 前段ゲートウェイ
  - 認証・認可
  - 入力内容の正規化や validation
- 証券オペレーション向け管理画面
  - 口座開設審査
  - 顧客対応、取引状況の確認
  - 重要書面管理
- システム運用者向け機能（管理画面の一部）
  - 強制アップデート、メンテナンスモード
  - FeatureFlag

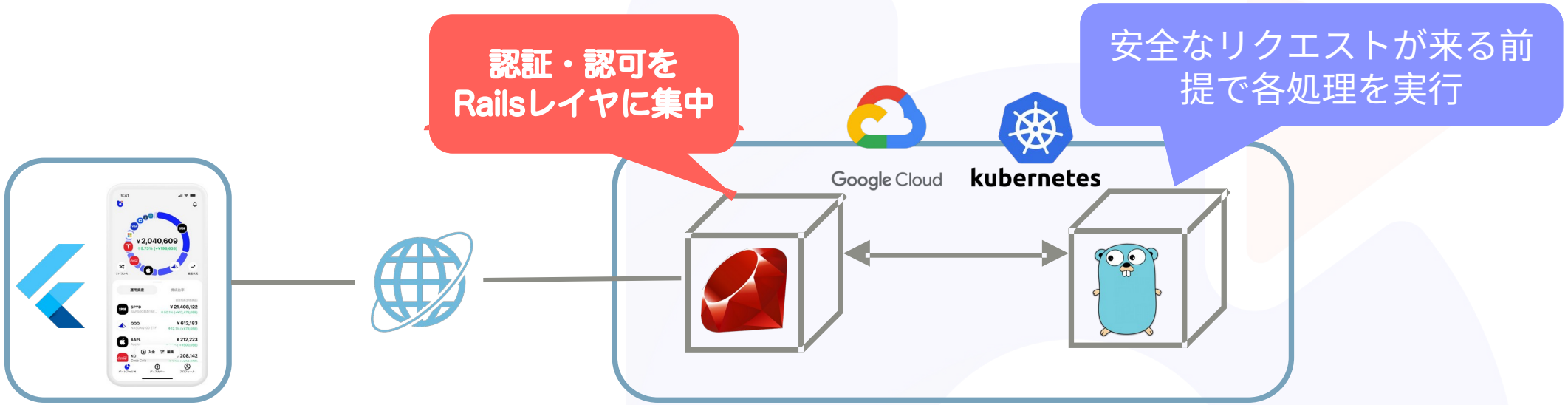
bloomo

口座開設機能

- 顧客に氏名や住所をアプリで入力してもらう
- 本人確認フローは外部の eKYC サービスを活用
- 個人情報管理と暗号化は Rails レイヤーで自前実装
  - Ruby on Rails には ActiveRecord が提供する暗号化機構があるが、より高度な要件を満たすために attr_enctypted gem を活用して実装
  - 個人情報のレコードごとに暗号鍵が異なる、安全性の高い構造

https://kaigionrails.org/2024/talks/f-world21/

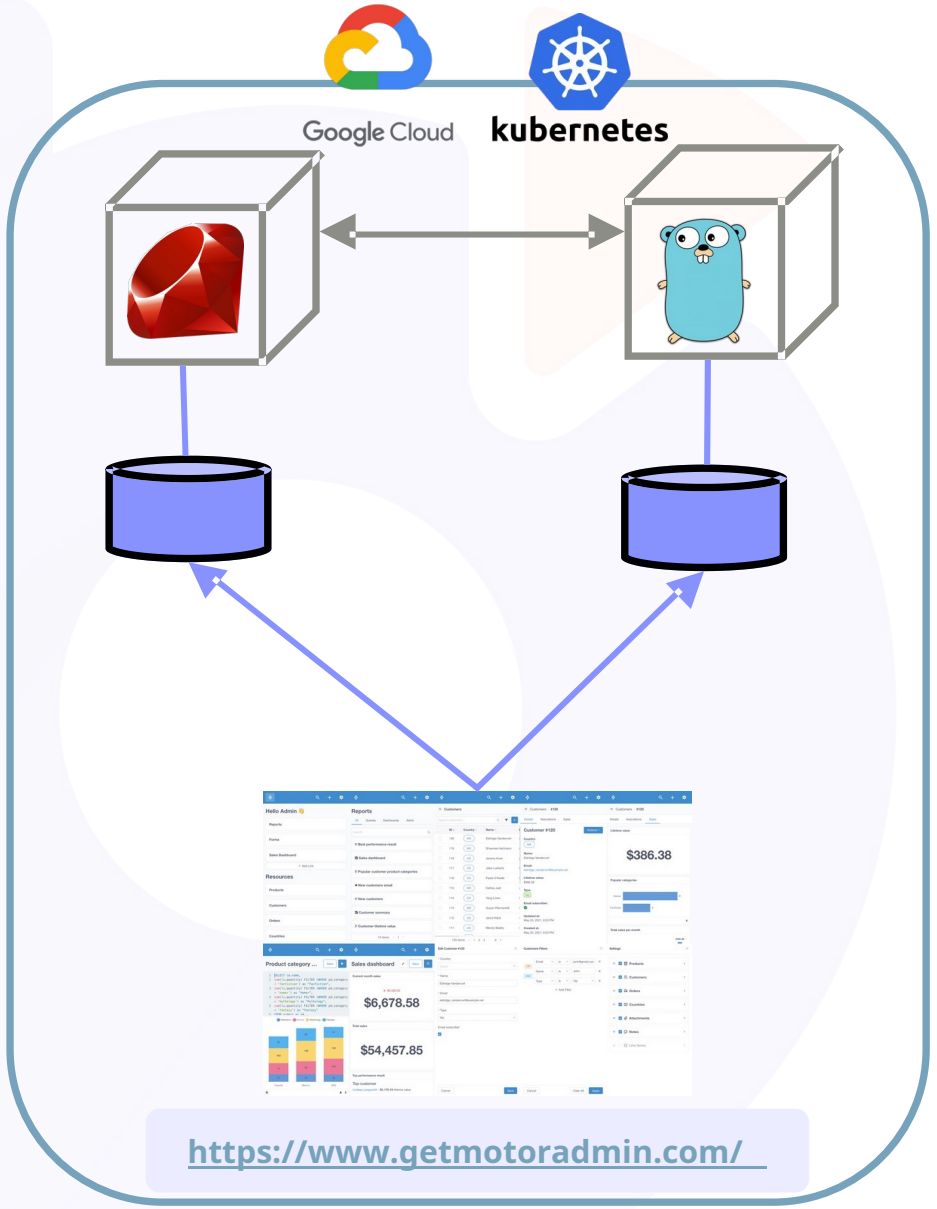
前段ゲートウェイ



- 初期実装時は Ruby on Rails でよく使われる devise gem を活用して実装
- Rails の後ろにいる Go のモジュールは認証・認可が取れてる前提で、メインの取引機能の実装に集中できるようになった
- ネット証券を中心とした不正アクセスや乗っ取り騒動時には速やかに二要素認証を導入
  - devise -> firebaseauthentication へ移行

証券オペレーション・システム運用者向け管理画面

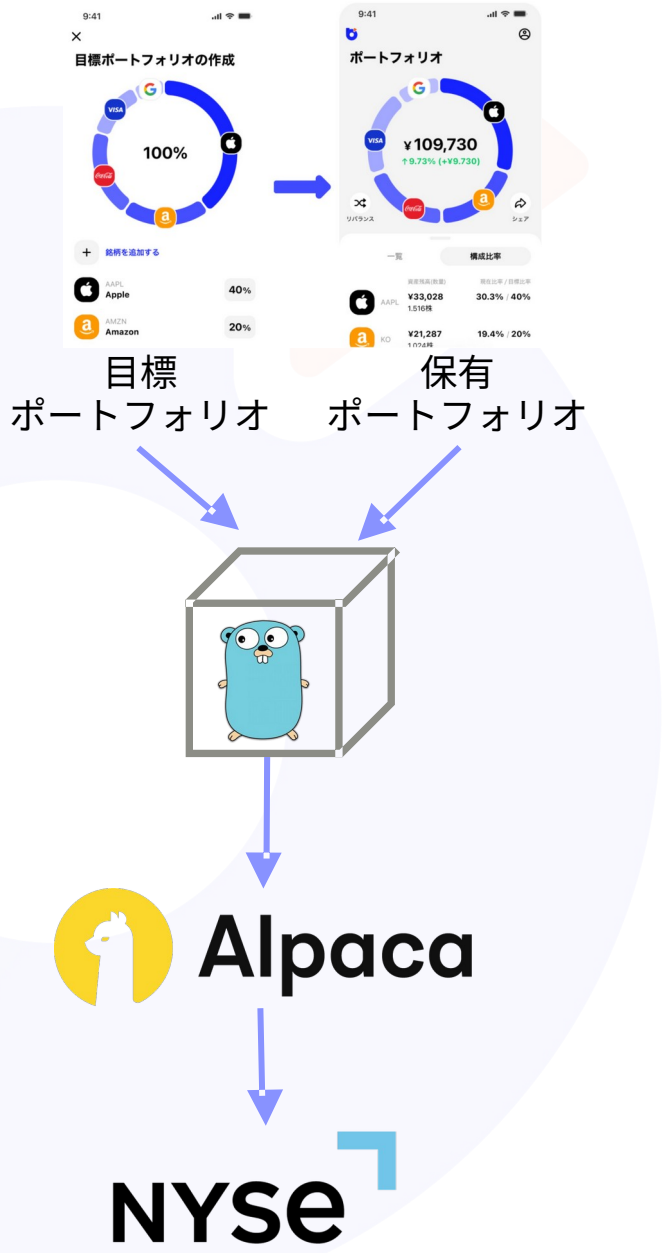
- ActiveAdmin, RailsAdmin といった Admin Dashboard 系を一通り調査
- motor-admin-rails gem を使って初期的な管理画面を実装
  - 選定理由は『一番 UI がキレイだったから』
  - カスタマイズもかなりできて、自由度高い
- Rails の**複数 DB 接続機能**を活用し、 Go 側モジュールが管理してる DB の内容も見れるようにした
  - Go 側モジュールの DB については読み込みは DB 直アクセス、書き込みや API 経由
  - 最終的にはすべて R/W どちらも API 経由に移行予定
- 強制アップデート、メンテナンスモード、FeatureFlag 管理といったシステム運用者向け機能も管理画面内に実装
- 非同期処理やバッチ処理は sidekiq で管理



04
Go が担う領域:
お金を扱う領域

Go が担う領域

- 投資をするうえでの目標値となる目標ポートフォリオと実際に保有している資産のポートフォリオ管理
- 注文計算
  - 買付、売却、リバランス
  - 自動での配当金再投資
- 米国の証券会社（ Alpaca Securities ）に取り次いでもらって米国市場に発注
  - Alpaca Securities が公開している API を使用
  - 日本の証券会社で米国株を買うときは他社も現地証券会社に取り次いでもらってる
- 約定取込→顧客への資産配分→金銭照合

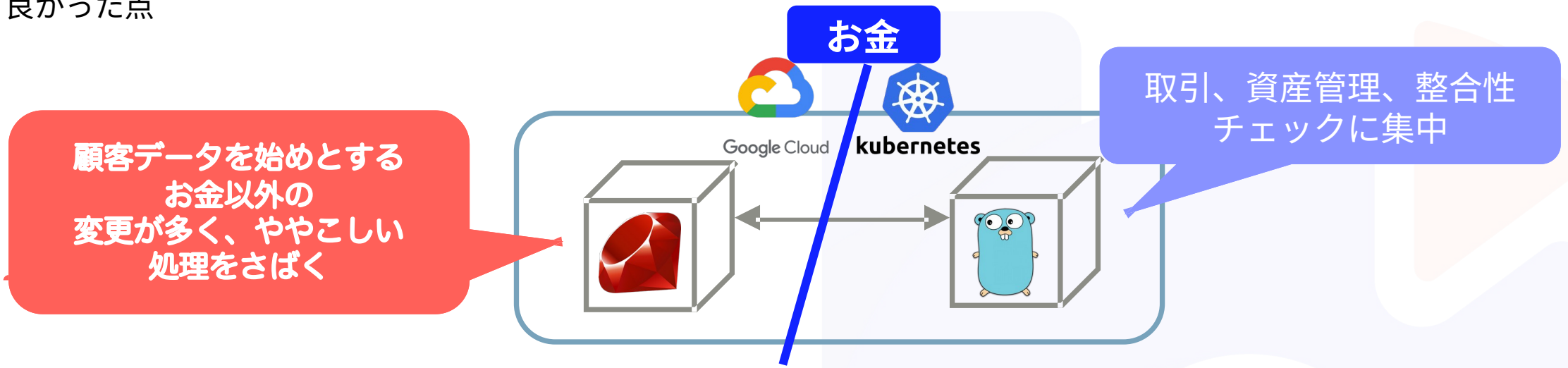


Go 側モジュールを実装するうえで気をつけていること

- （今は）パフォーマンスより整合性重視
- 冪等性を担保できるように API 設計
  - 例：同一顧客から注文リクエストが複数同時にきても、整合性が保たれるように実装
- 変更が入るリクエストがきた場合、処理の開始時点で、起点となるレコードの行 Lock を取得してから排他処理を実現
- データが不整合な状態になっていないかを検証するバッチを複数回実行している
- バッチ処理は Kubernetes レイヤーで管理

05
何が良く、どこが辛かったか

良かった点



- **境界を「お金」で引くことで各コンポーネントの責務が明瞭化し、認知負荷が下がった**
- Rails と Go で独立してデプロイが可能になり、変更の局所化できている点
- チーム内に技術的な多様性が生まれた
  - Ruby： 動的型付け、オブジェクト指向
    - 設計レイヤーの話は『とりあえず Ruby on Rails のお作法にできるかぎり乗る』
  - Go： 静的型付け、（独自の？）オブジェクト指向
    - 設計レイヤー含め自分たちで決めて作る文化が強い
  - エンジニアチーム内で各言語やパラダイムの良い点、悪い点を共有しつつ

辛かった点

- 統合・結合テスト
- 監視・モニタリング
- 二言語運用の教育 / 採用コストと増加
- Rails 側と Go 側でどうしても重複・ 2 重管理せざるを得ないデータが出てきてしまう点（技術的負債）

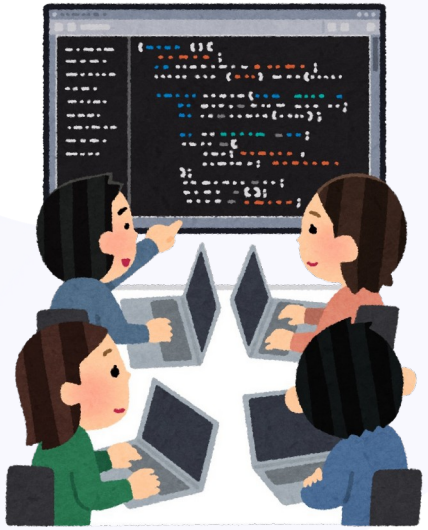
学び（＝辛かった点を覆すための工夫や課題）

統合・結合テスト

- E2E のテストは顧客が口座開設から入金・発注までを行うパスを中心に見ることによりデグレや後方互換性を検証するようにした
- 自動化できてるのはごく一部で**まだまだ人間が手で検証してる**

監視・モニタリング

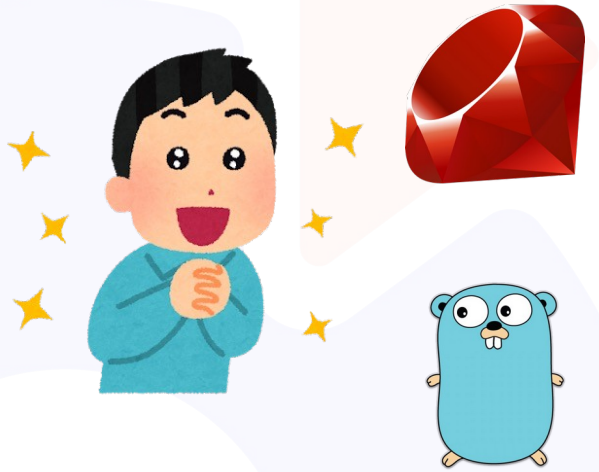
- Opentelemetry や Sentry を導入
- SLI/SLO ダッシュボードを構築し、エンジニア全員で確認＆改善していってる
- エラーが起きたときのアラート通知からのインシデント対応プロセスの整備
- 監視モニタリングプロセスの効率化も今後の課題



学び（＝辛かった点を覆すための工夫や課題）

二重言語対策

- 技術カットでチームを区切るのではなく、顧客の関心領域ごとにチームを区切るようにした
- Rails と Go を書くエンジニアがワンチームにまとまり、顧客が抱える課題に対して同じ目線を持てる
- **『幅広い技術に触れたい』というモチベーションを持つメンバーを迎え入れる**



2 重データなどの技術的負債に対する対策

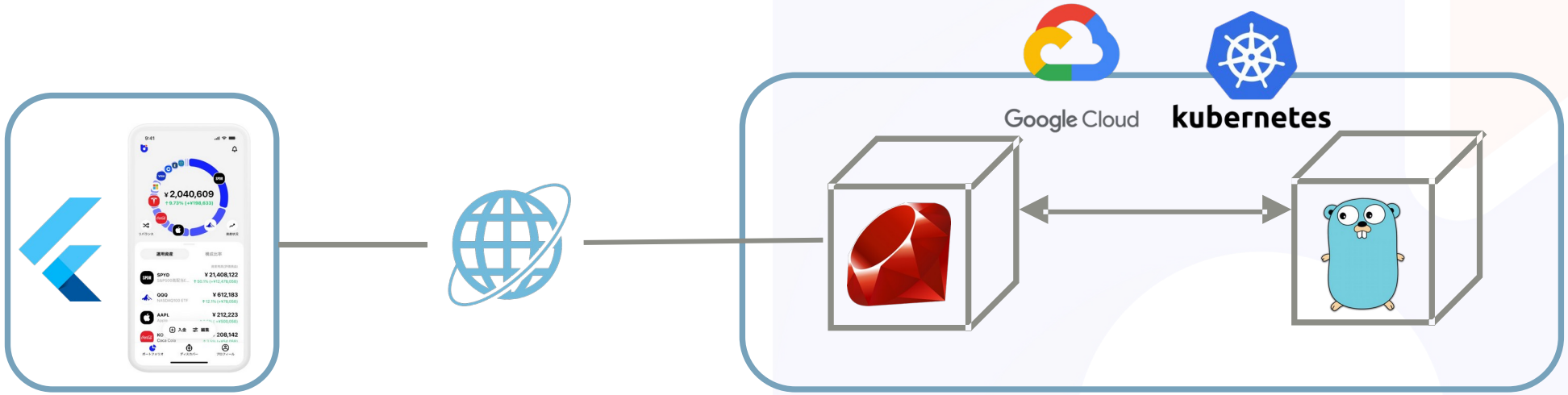
- スピード優先でデータが重複してしまう領域がどうしても発生する
- ビジネスサイドを含め、継続的なリファクタリングを行うことに同意
- 常により良く整合性の取れたアーキテクチャに変更していくための**投資をする文化**を醸成

06
まとめ

覚えて帰ってほしいこと

- **ブルーモ証券＝ Ruby コミュニティー発の証券会社**
- 『顧客のお金を扱うかどうか』でシステムとしての実装と採用技術を分けている
- 何か特別なことをしているわけではなく、一つ一つはよくある技術課題
  - リリース時点で実装しておかなければいけない機能や運用ルールなどが幅広い
  - 証券会社のシステムもゼロから作れる
- エンジニアチーム内の適度な**技術的多様性**が生まれた
- テストや監視・モニタリングは仕組みを整えつつも今はまだ人力で頑張ってる点も多い
- ビジネスサイド含め、**会社全体で継続的な改善に投資する**ことを合意

We are Hiring!



一緒に Bloomo のサービス開発をしてくれる仲間を募集中！

bloomo

https://careers.bloomo.co.jp/