

プログラミング学習サービスの
これまでを支えてきた技術と、
これからを創っていく技術



Index

- 自己紹介
- Progateについて
- 「これまで」を支えてきた技術
- 「これから」を創っていく技術



自己紹介



● 登壇者の紹介



Kazuki Maeda

前田 和樹

前職リクルートにてSREやPMなどに従事。現在、「Engineer Function」のマネージャーを担う。
AWS Community Builder Member



Makoto Shimazu

島津 真人

新規サービス企画・開発のチームのテックリード。
前職は Google のChromeチームで Software Engineer として勤務。
STEP教育コースの講師も行ってた。



Progateについて



● Progateとは
Progateの紹介



言語、フレームワークなど
全25以上のコースが

¥1078 /月 (税込)



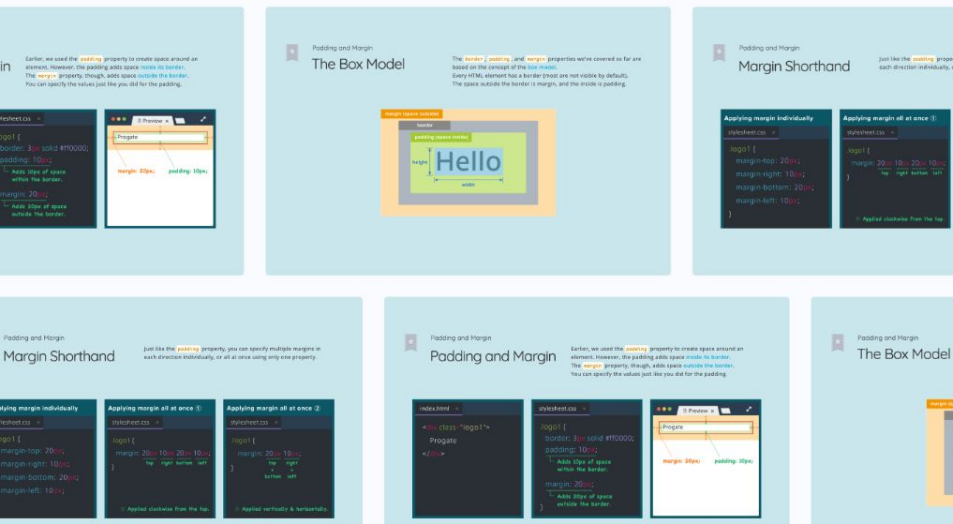
全コース、最初の
1レッスンは
無料

● Progateとは

Progateの学習方法

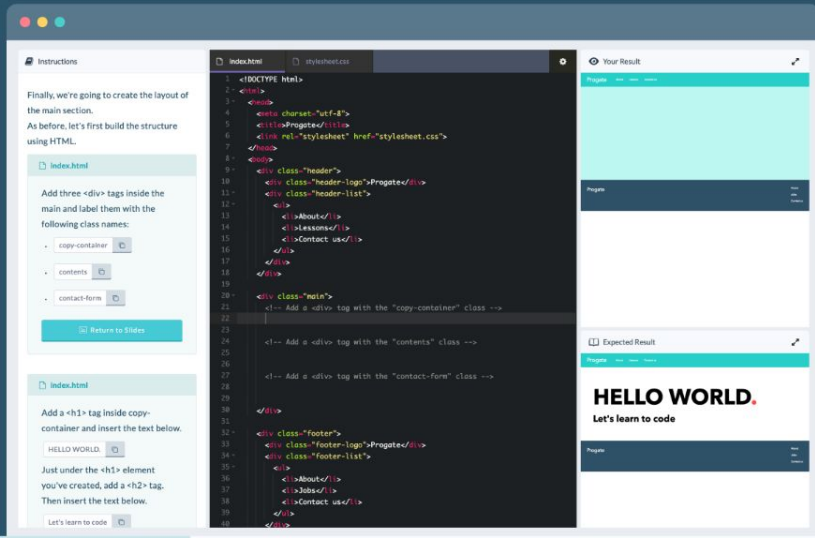
イラスト中心のスライドで学ぶ。

紙の本よりも直感的で、動画よりも学びやすい、「スライド学習」を採用しました。自分のペースで学習できること、復習しやすいことが強みです。



自分でコードを書いてすぐに実践。

ブラウザ上で、実際にコードを書いて結果を確認。初心者がつまずきやすい環境構築は不要です。

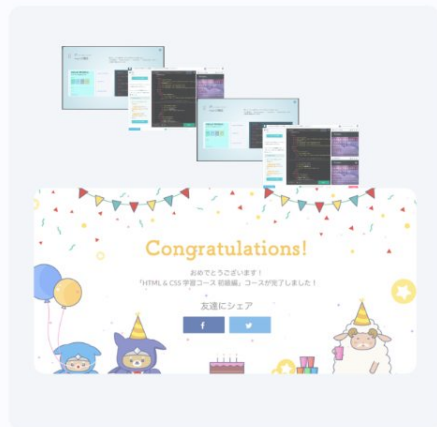


- Progateとは

Progateのこだわり

もっとやりたくなる・挫折しないUI/UX

スモールステップで
達成感を味い学ぶ



ゲーミフィケーションなどを
用いたハマるしかけ



オンラインでの
支え合い (Twitterなど)



- プロダクトについて

Progateアプリ版の紹介

progate アプリ版



ゲーム感覚で

サクサク学べるアプリ版

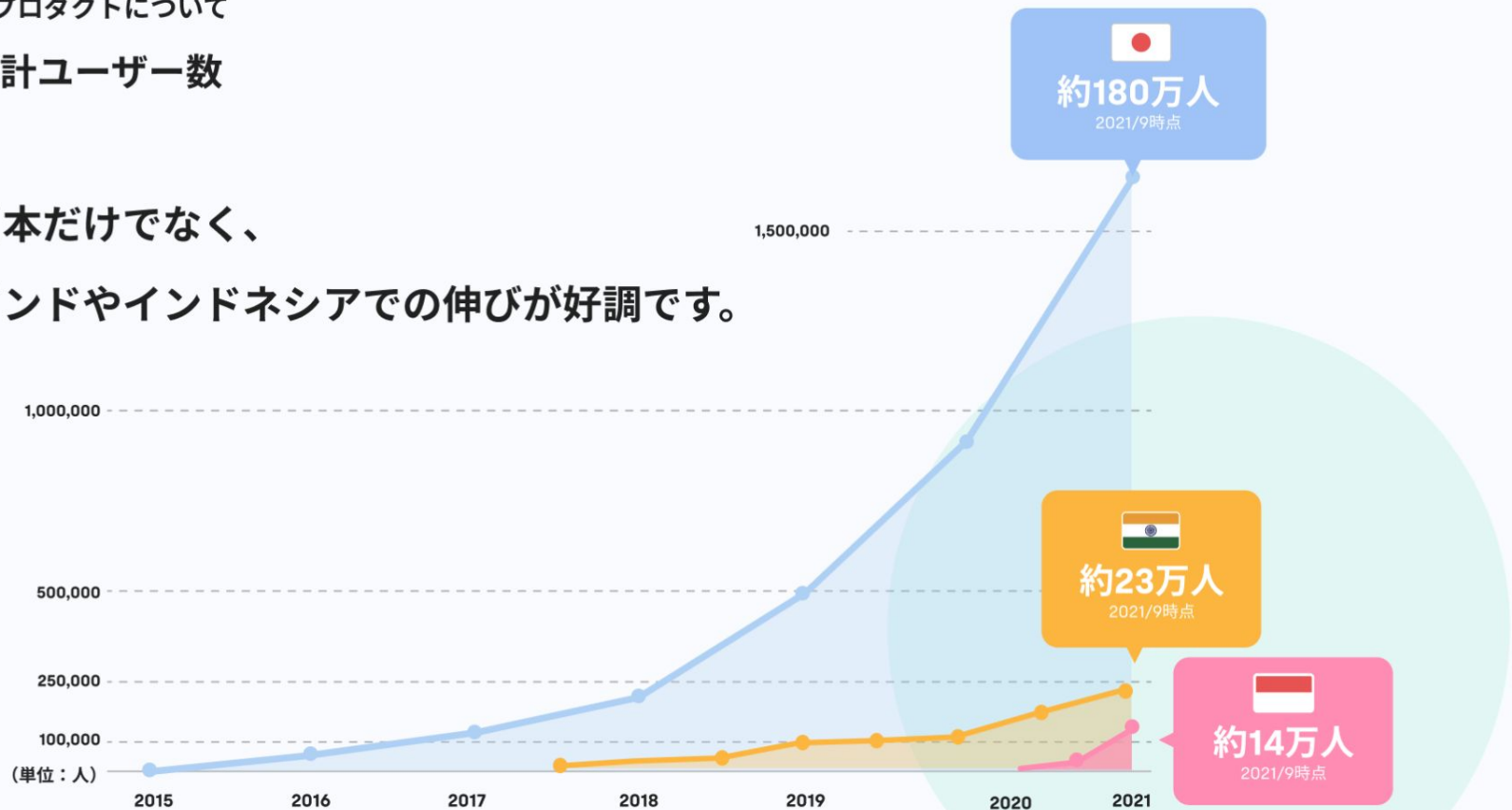
スマホやタブレットの画面に最適化したスライドとキーボードを用意しています。「まずは気軽に始めてみたい」という方にオススメです。



- プロダクトについて

累計ユーザー数

日本だけでなく、
インドやインドネシアでの伸びが好調です。

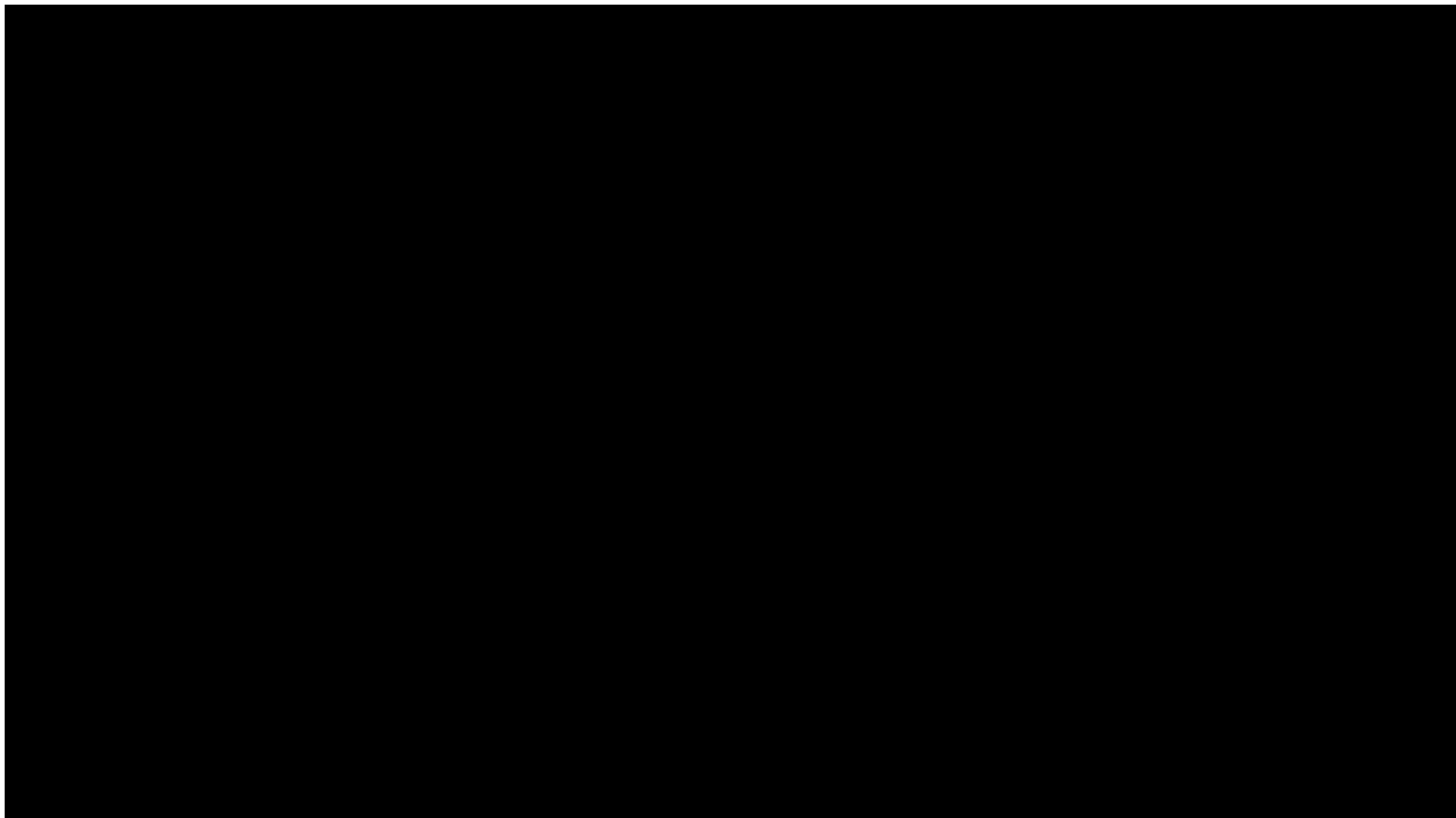


- Progateとは

日本だけではなく、
世界をターゲットにすることで面を広げ
提供価値の最大化を目指していきます。



サービスデモ(動画)



「これまで」を支えてきた技術

「これまで」のサービス提供の 難しいポイント



「これまで」のサービスを提供する上での難しいポイント

- ポイントに絞った学習
- ユーザーのコードを「実行」する
- ユーザーのコードを「判定」する
- グローバル展開対応



「これまで」のサービスを提供する上での難しいポイント

- ポイントに絞った学習
- ユーザーのコードを「実行」する
- ユーザーのコードを「判定」する
- グローバル展開対応



InputとOutputの学習設計

イラスト中心のスライドで学ぶ。

紙の本よりも直感的で、動画よりも学びやすい、「スライド学習」を採用しました。自分のペースで学習できること、復習しやすいことが強みです。



自分でコードを書いてすぐに実践。

ブラウザ上で、実際にコードを書いて結果を確認。初心者がつまずきやすい環境構築は不要です。

Earlier, we used the `padding` property to create space around an element. However, the padding adds space inside the border. The `margin` property, though, adds space outside the border. You can specify the values just like you did for the padding.

Padding and Margin
The Box Model

Padding and Margin
Margin Shorthand

Applying margin individually

Applying margin all at once (1)

Applying margin all at once (2)

```
index.html <div class="logo">
  <img alt="Logo" />
</div>
<div class="page">
  <h1>Hello World</h1>
  <h2>Let's learn to code</h2>
  <div class="contact-us">
    <button type="button">Contact us</button>
  </div>
</div>
```

```
stylesheet.css
body {
  font-family: sans-serif;
  padding: 10px;
}
h1 {
  margin-top: 20px;
  margin-right: 10px;
  margin-bottom: 20px;
  margin-left: 10px;
}
h2 {
  margin-top: 20px;
  margin-right: 10px;
  margin-bottom: 20px;
  margin-left: 10px;
}
div {
  padding: 10px;
}
```

Finally, we're going to create the layout of the main section. As before, let's first build the structure using HTML.

index.html

Add three `<div>` tags inside the main and label them with the following class names.

```
index.html
<div class="main">
  <div class="header">
    <div class="header-logs">
      <div class="header-list">
        <li>About</li>
      </div>
    </div>
  </div>
  <div class="page">
    <h1>Hello World</h1>
    <h2>Let's learn to code</h2>
    <div class="contact-us">
      <button type="button">Contact us</button>
    </div>
  </div>
</div>
```

```
stylesheet.css
body {
  font-family: sans-serif;
  padding: 10px;
}
h1 {
  margin-top: 20px;
  margin-right: 10px;
  margin-bottom: 20px;
  margin-left: 10px;
}
h2 {
  margin-top: 20px;
  margin-right: 10px;
  margin-bottom: 20px;
  margin-left: 10px;
}
div {
  padding: 10px;
}
```

Your Result

Expected Result

HELLO WORLD.
Let's learn to code

HELLO WORLD.
Let's learn to code

HELLO WORLD.
Let's learn to code

学んだ内容をすぐにアウトプットできる学習設計

ステップバイステップの演習

Ruby 学習コース I > Rubyを動かしてみよう > 1. Rubyを実行してみよう

手順

まずは、そのままRubyのコードを実行してみましょう。

「コンソール」に、あなたが書いたコードの実行結果が出力されます。

```
index.rb
```

「コンソール」の「▶」ボタンを押してください。

コンソールに「Hello World」と出力されましたか？
次は、「見本」の「▶」ボタンを押してみましょう。
見本と同様に「Hello Ruby」と出力されるように、コードを少し書き換えてみましょう！

```
index.rb
```

1行目の「Hello World」の部分

Hello Ruby に書き換えて

スライドで確認

「できた!」を押して、先に進みましょう。

```
index.rb
```

```
1 puts "Hello World"
```



Ruby on Rails5 学習コース VI > ユーザーの新規登録 > 7. ユーザー登録ページを作ろう

手順

ここでは、ユーザー登録ページの見た目を作っていきます。

```
../config/routes.rb
```

「localhost:3000/signup」にアクセスしたときに、Usersコントローラーのnewアクションを呼び出すルーティングを作成してください。

```
../controllers/users_controller.rb
```

newアクションを作成してください。

```
index.html.erb
```

```
1 <div class="main posts-index">
2   <div class="container">
3     <% @posts.each do |post| %>
4       <div class="posts-index-item">
5         <%= link_to(post.content, "/posts/#{post.id}") %>
6       </div>
7     <% end %>
8   </div>
9 </div>
```

```
<div class="main users-new">
<div class="container">
<div class="form-heading">新規ユ
```

レッスンを進めるごとにできることが積み上がっていく学習設計

「これまで」のサービスを提供する上での難しいポイント

- ポイントに絞った学習
- ユーザーのコードを「実行」する
- ユーザーのコードを「判定」する
- グローバル展開対応



ブラウザ上で動くエディタ・ターミナル・ブラウザ

Quest 道場コース ツイートアプリ > ツイートアプリのバグ修正 > 1. いいね...

エディタ ターミナル ブラウザ データベース

報告されているエラーを修正しましょう。
「答えを見る」の中にヒントが用意されています。困ったら見てみましょう。

※この問題に答えは用意されていません

まずはバグを再現してみましょう

バグ再現手順

1. にんじゃわんこのアカウントにログインする
2. /users/1/likes にアクセスして、いいね一覧が表示されることを確認
3. ログアウトする
4. /users/1/likes にアクセスして、エラーが

```
application_controller.rb
1- class ApplicationController < ActionController::Base
2-   before_action :set_current_user
3-
4-   def set_current_user
5-     @current_user = User.find_by(id: session[:user_id])
6-   end
7-
8-   def authenticate_user
9-     if @current_user == nil
10-      flash[:notice] = "ログインが必要です"
11-      redirect_to("/login")
12-    end
13-  end
14-
15-  def forbid_login_user
16-    if @current_user
17-      flash[:notice] = "すでにロ
18-      redirect_to("/posts/index"
19-    end
20-  end
21-
22-  end
23-
```

ターミナル1 ターミナル2

```
tweet_app $ ls
Gemfile      README.md   app          config       db
Gemfile.lock Rakefile   bin          config.ru    lib
tweet_app $ pwd
/home/progate/tweet_app
tweet_app $ echo 'Hello world!'
Hello world!
tweet_app $
```

プレビュー

localhost:3000/posts/index

TweetApp

にんじゃわんこ 投稿一覧 新規投稿 ユーザー一覧 ログアウト

ログインしました

- にんじゃわんこ**
もう少しでTweetApp完成するぞー！
- ベイビーわんこ**
にんじゃわんこ兄ちゃん、最近プログラミング頑張ってるなあ。創りたいサービスがあるらしい。僕も頑張るぞー！
- しょう**
【プログラミング学習のProgate】Ruby on Rails学習コースIIIを修了しました！レベルがどんどん上がっていくから楽しい！
- みちこ**
第1回Progate Sunday Schoolに参加してきた！にんじゃわんこ

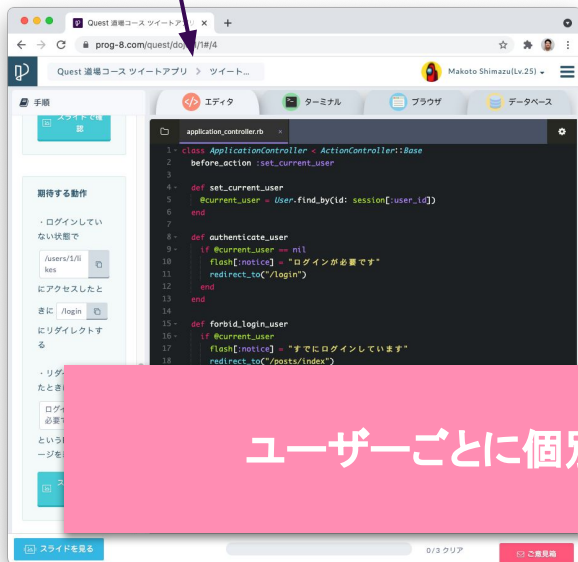
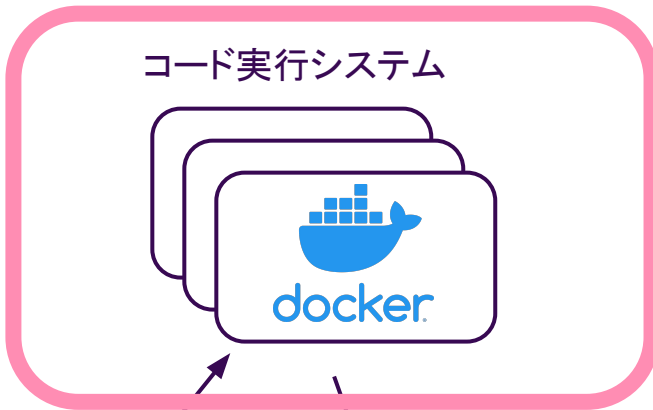


コードの実行

Web Server



コード実行システム



ユーザーごとに個別のコード実行環境を提供している



「これまで」のサービスを提供する上での難しいポイント

- ポイントに絞った学習
- ユーザーのコードを「実行」する
- ユーザーのコードを「判定」する
- グローバル展開対応

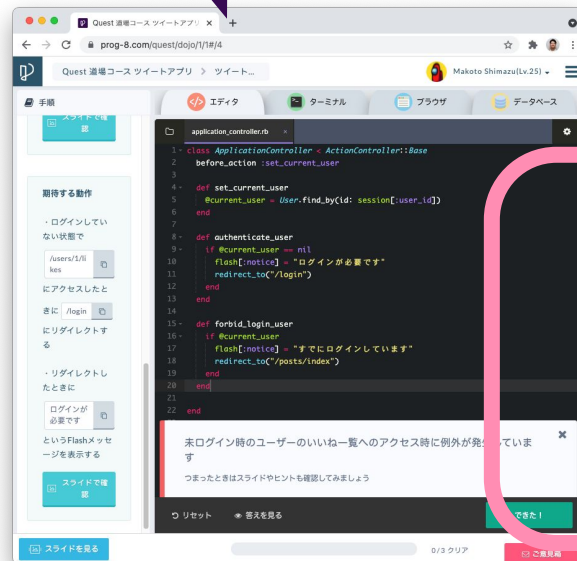
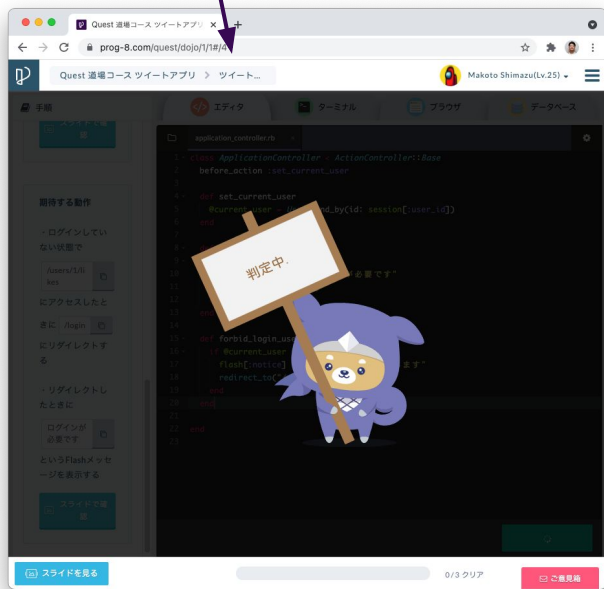
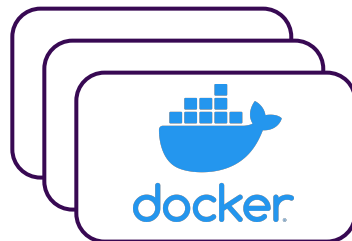


コードの判定

Web Server



コード実行システム



独自の判定ロジック

書かれているコードは
想定通り？



独自の判定ロジック

書かれているコードは
想定通り？

コードの実行結果は
正しい？



独自の判定ロジック

書かれているコードは
想定通り？

テストは通る？



コードの実行結果は
正しい？

独自の判定ロジック

書かれているコードは
想定通り？

テストは通る？



コードの実行結果は
正しい？

生成されたHTMLと
見本との差異は？

独自の判定ロジック

書かれているコードは
想定通り？

テストは通る？



コードの実行結果は
正しい？

生成されたHTMLと
見本との差異は？

レッスンに応じて結果を多角的に判定する独自の判定ロジック



「これまで」のサービスを提供する上での難しいポイント

- ポイントに絞った学習
- ユーザーのコードを「実行」する
- ユーザーのコードを「判定」する
- グローバル展開対応



コンテンツの多言語対応

Getting Started with HTML

How HTML Works

Let's get started with HTML!
The first rule of writing HTML code is to surround text with **tags**.
Tags give a meaning, such as a [heading](#) or a [link](#), to text.

index.html x

Surround text with tags.

```
<h1> Hello World </h1>
```

```
<a> Progate </a>
```

Preview x

Hello World
Progate

Without HTML

Hello World
The text becomes a "heading".
Progate
The text becomes a "link".

With HTML

Next Slide (or the → key)

Memulai dengan HTML

Mari kita mulai pelajaran kita dengan HTML!
Aturan pertama penulisan code HTML adalah mengapit teks dengan tag. Tag dapat memberikan arti seperti **judul** atau **tautan** pada teks.

Preview x

Hello World
Progate

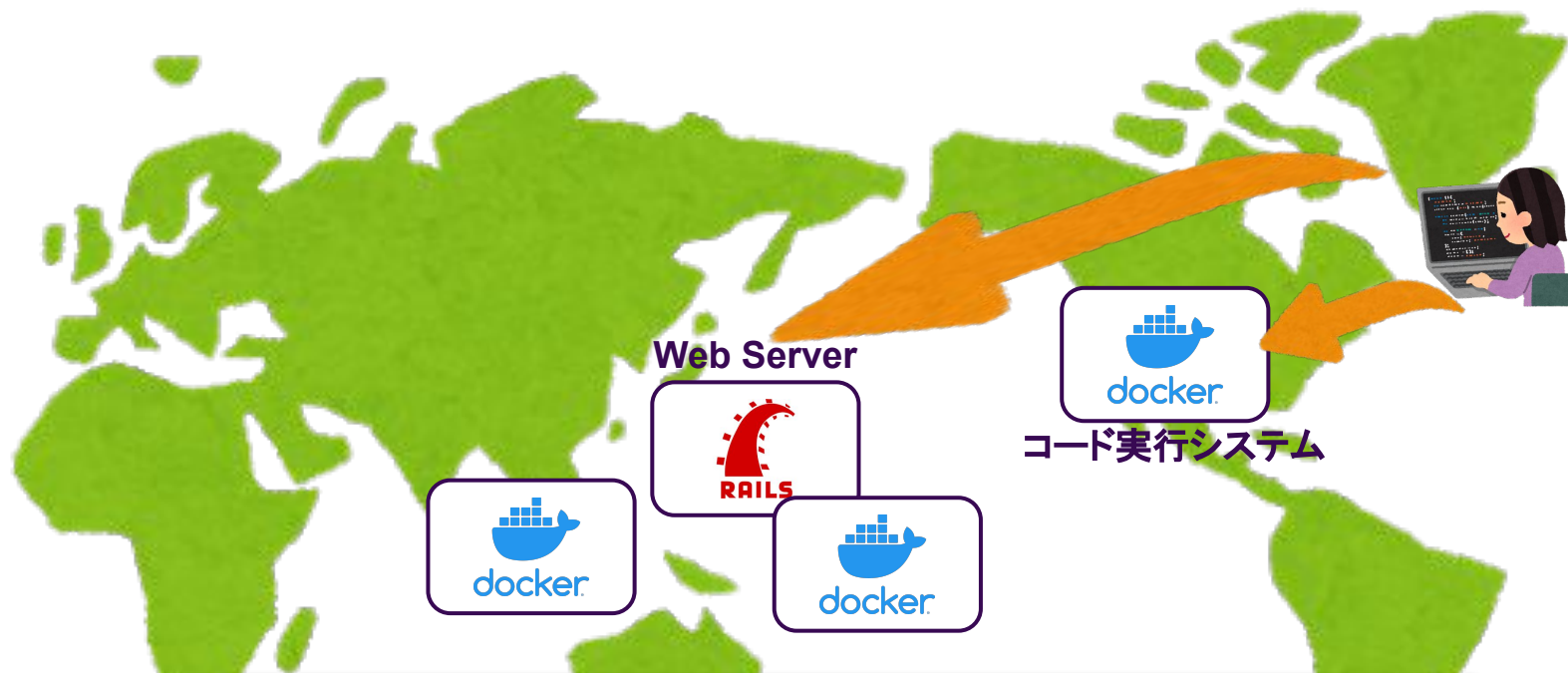
Tanpa HTML

Hello World
Teks ini menjadi "heading" (judul).
Progate
Teks ini menjadi "link" (tautan).

Dengan HTML

Slide Berikutnya (atau tombol →)

コード実行システムをグローバルに配置



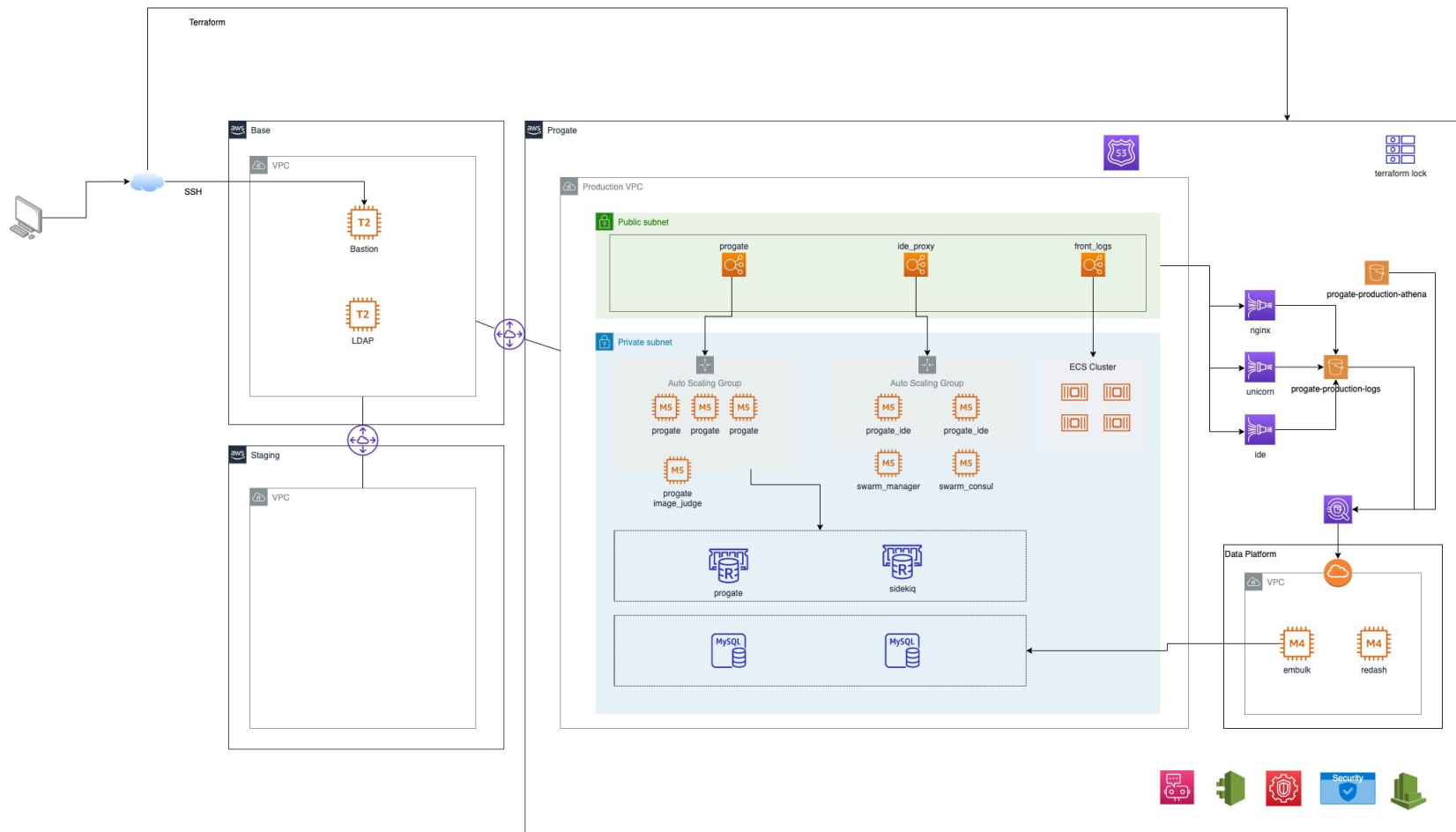
よりユーザーに近いロケーションでコードの実行が可能



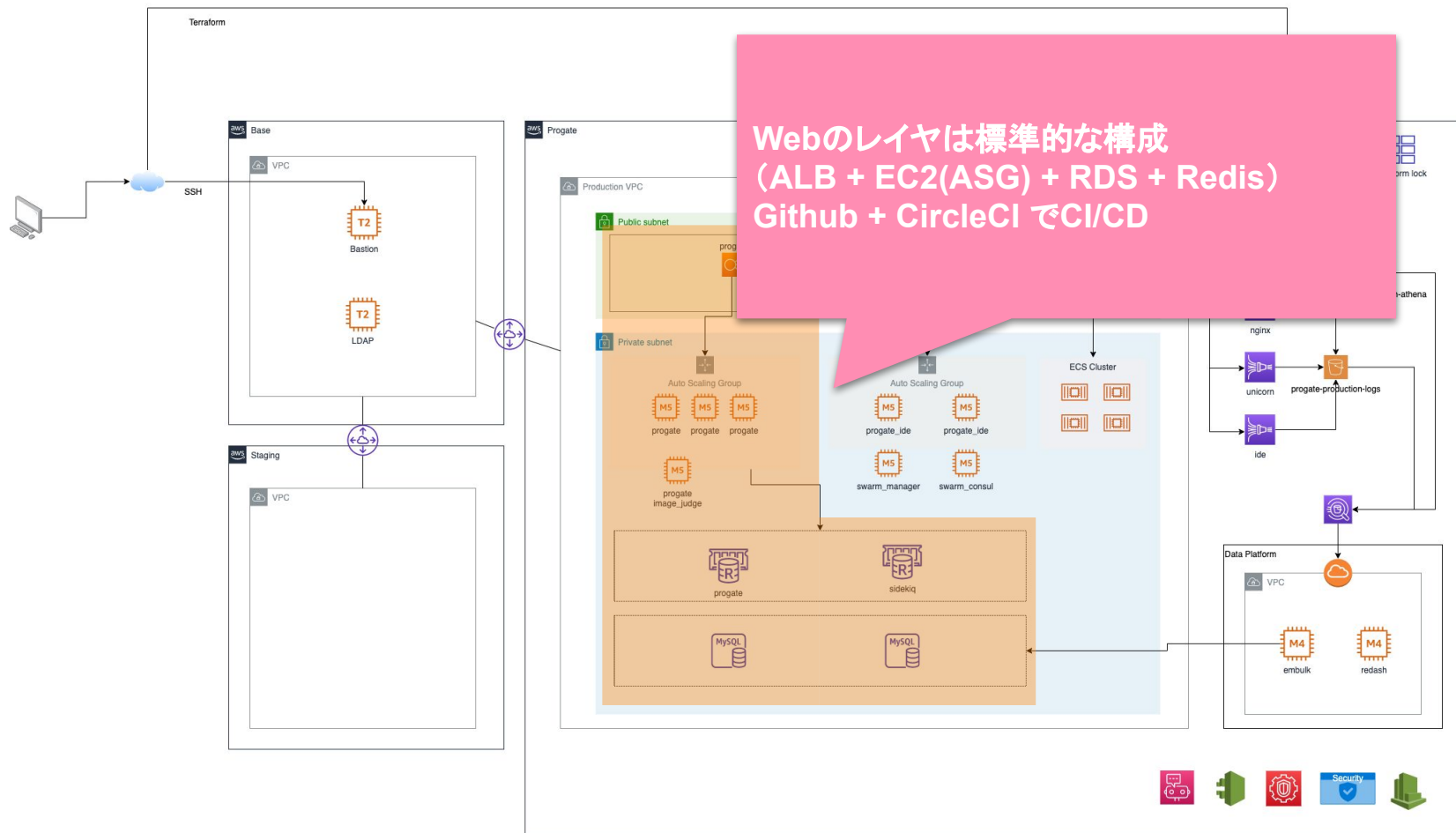
「これまで」のサービスを
構成しているアーキテクチャ



「これまで」のサービスを提供しているアーキテクチャ



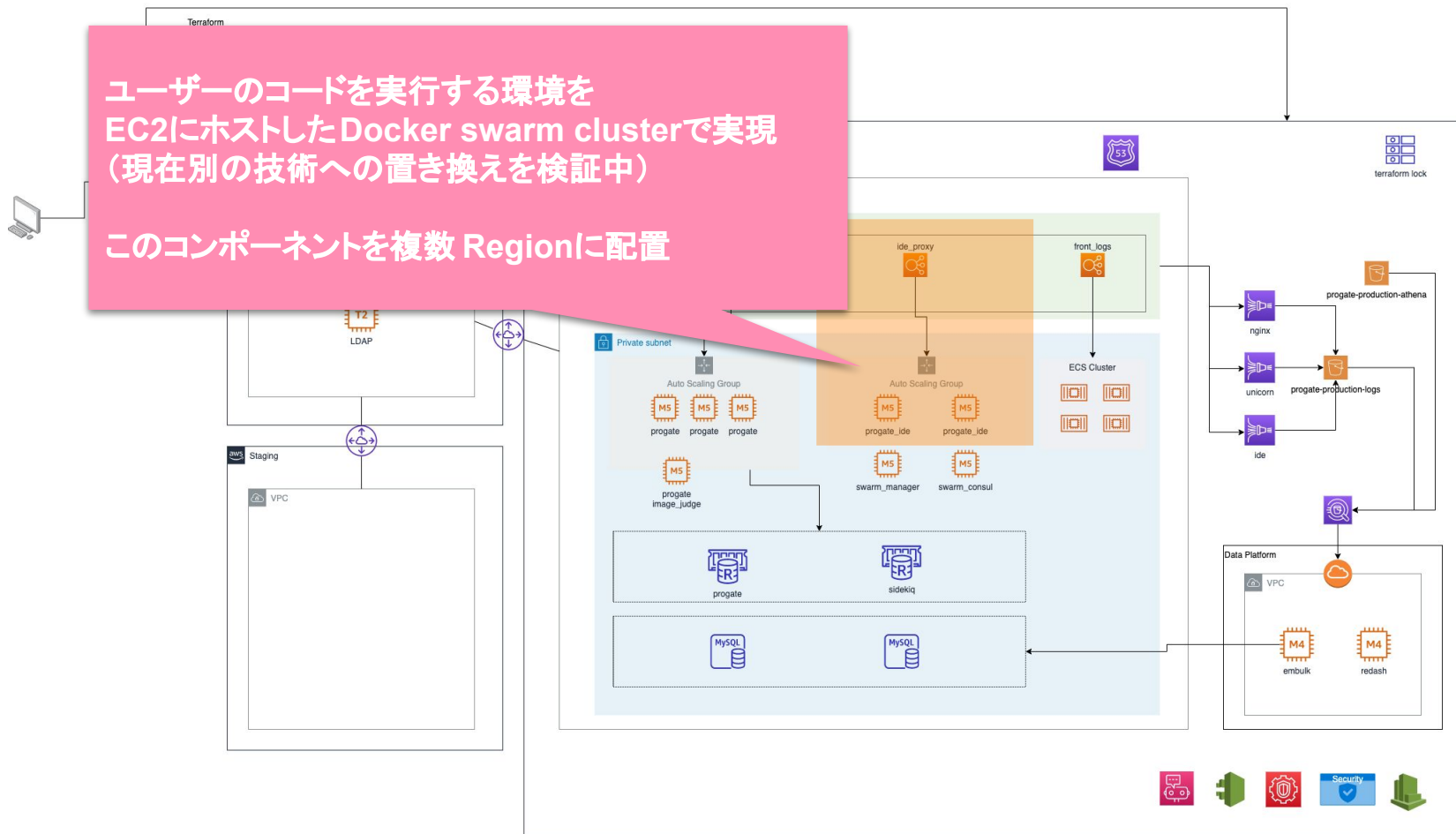
基本的なWeb構成



ユーザーのコード実行システム

ユーザーのコードを実行する環境を
EC2にホストしたDocker swarm clusterで実現
(現在別の技術への置き換えを検証中)

このコンポーネントを複数 Region に配置



アーキテクチャ進化の一例

クライアントサイドのログを収集する基盤は
マイクロサービス化して ECS Cluster (Fargate) で構築

→ TechBlog で詳細解説しています！

<https://tech.prog-8.com/entry/2021/03/17/080000>



progate | Tech Blog

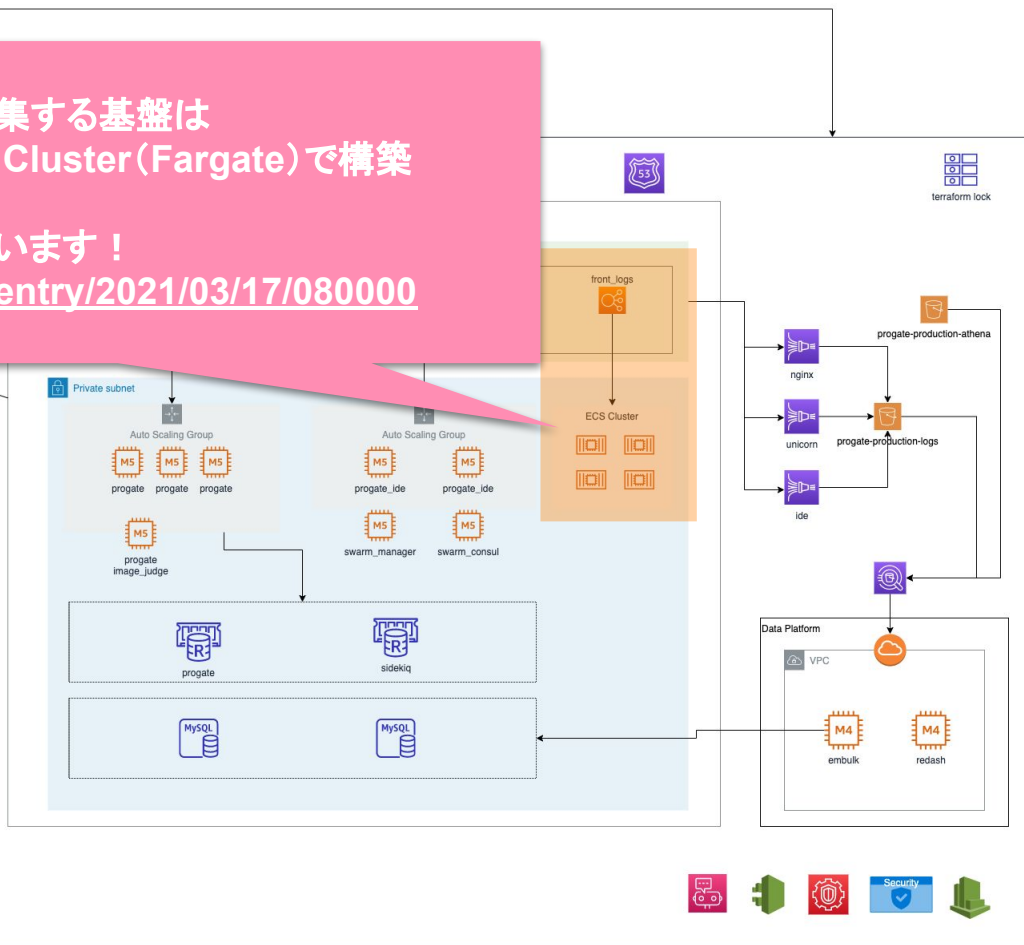


2021-03-17

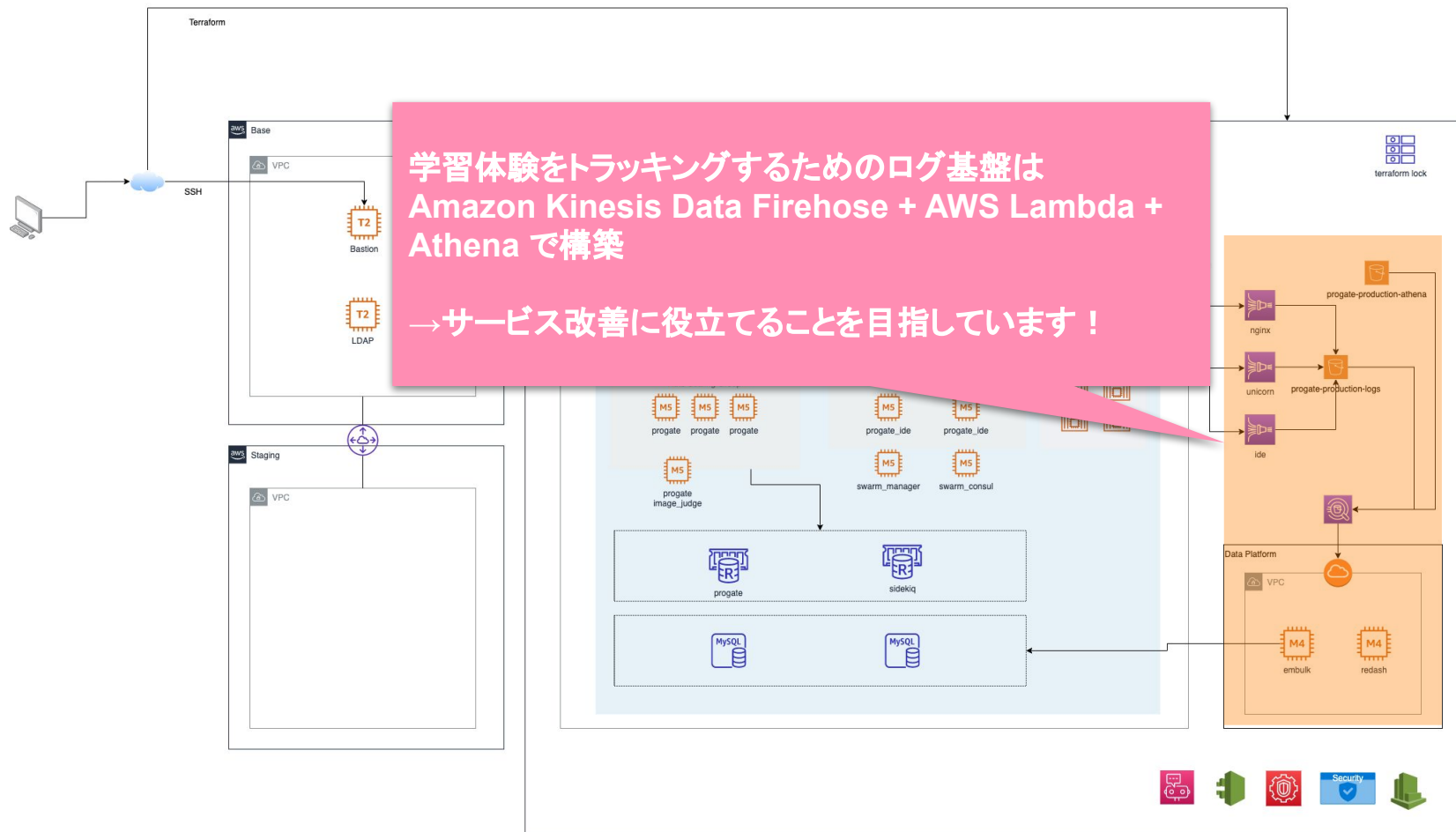
モノリシックサービスから高負荷なエンドポイントを切り出して段階的に運用改善した話

Progateの小笠原です。普段はSREチームで開発効率化やサービスの安定化に取り組んでいます。

本稿では弊社SREチームで取り組んだ事例の一つである「モノリシックなサービスから高負荷なエンドポイントを切り出して段階的に運用改善した話」について紹介させていただきます。

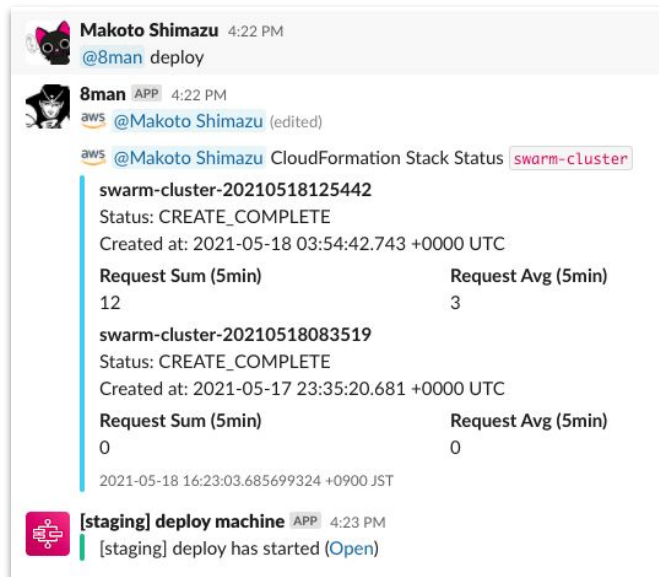


マネージドサービスの活用



AWSを活用していたからできたこと

- 枯れた技術を用いた安定したサービス提供と、アップデートの早い豊富なマネージドサービスを用いたスピーディな新規開発
 - メインのWebシステムはEC2ベースの枯れた技術を選定していますが、非常に安定的にサービス提供を行うことができます
 - データストリームやChatOpsなど新規に構築する機能はマネージドサービスを活用し、スピーディに開発できています



The screenshot shows a Slack conversation in a channel named "#[staging] deploy machine".

- Makoto Shimazu** (@8man) posted a message: "deploy" at 4:22 PM.
- 8man** (APP) (@Makoto Shimazu) posted a message: "CloudFormation Stack Status `swarm-cluster`" at 4:22 PM.
- The message content shows the status of two swarm clusters:
 - swarm-cluster-20210518125442**
 - Status: CREATE_COMPLETE
 - Created at: 2021-05-18 03:54:42.743 +0000 UTC
 - Request Sum (5min): 12
 - Request Avg (5min): 3
 - swarm-cluster-20210518083519**
 - Status: CREATE_COMPLETE
 - Created at: 2021-05-17 23:35:20.681 +0000 UTC
 - Request Sum (5min): 0
 - Request Avg (5min): 0
- A timestamp "2021-05-18 16:23:03.685699324 +0900 JST" is visible.
- [staging] deploy machine** (APP) posted a message: "[staging] deploy has started (Open)" at 4:23 PM.

Gate サービスのこれから



● Vision/Mission

Vision/Missionのイメージ

Mission

Vision



Be the gate

プログラミングの楽しさを
知る入口になる

- 楽しさが伝わる
- 自分にもできると思える
- 誰もが始められる

Be the path

自走できるところまで
導ける道になる

- 何を学ばばいいかわかる
- 必要な知識を得られる
- その知識を使って試行錯誤する経験ができる

An independent coder

壁にぶつかっても自分で
動き、考えて乗り越えられる

- 世の中にある問題を発見できる
- 問題に対する解を生み出すことができる
- 誰かの役に立つことができる

new doors

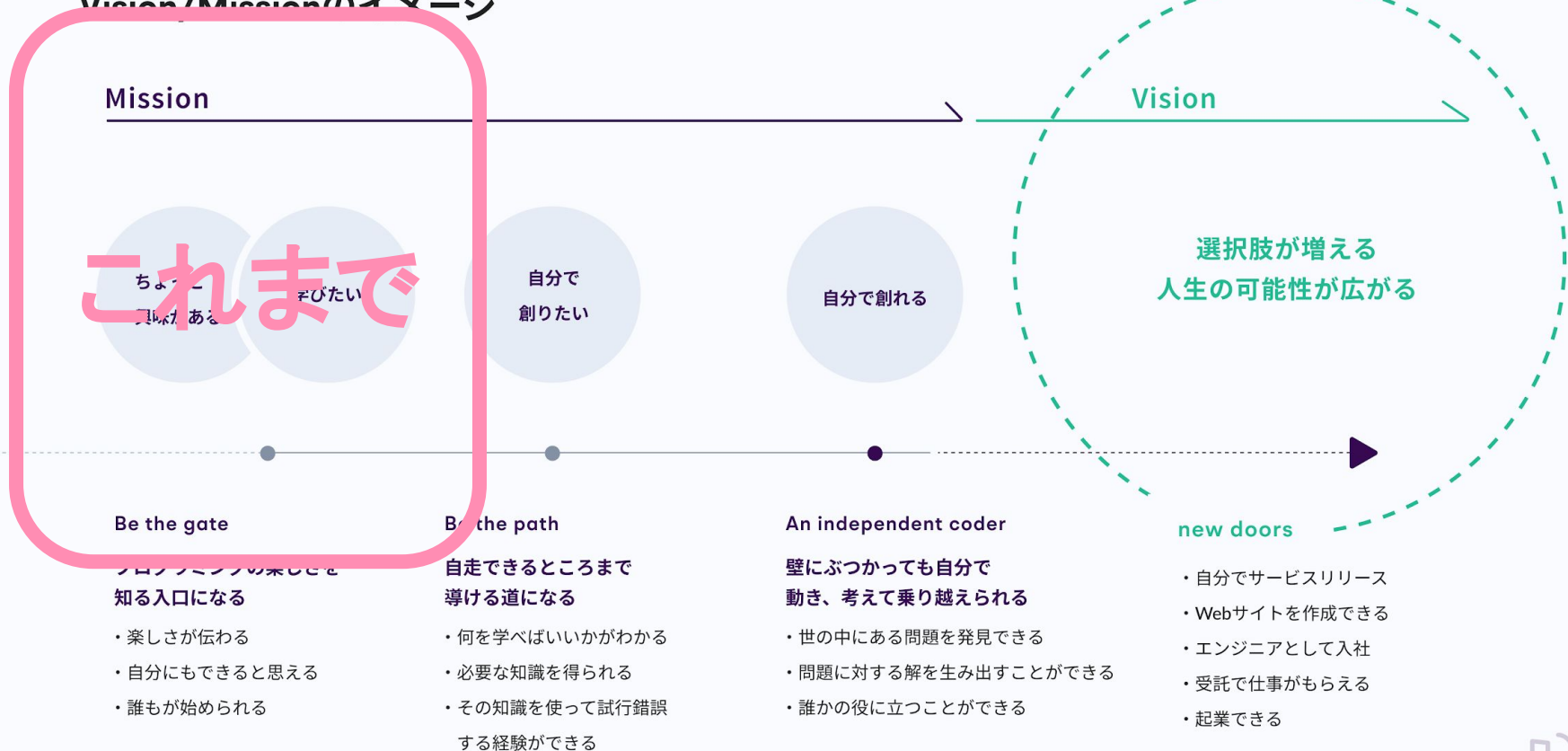
選択肢が増える
人生の可能性が広がる

- 自分でサービスリリース
- Webサイトを作成できる
- エンジニアとして入社
- 受託で仕事がもらえる
- 起業できる



● Vision/Mission

Vision/Missionのイメージ

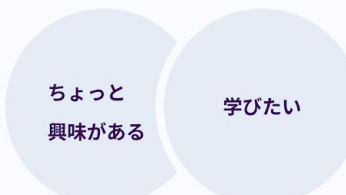


● Vision/Mission

Vision/Missionのイメージ

Mission

Vision



これから
選択肢が増える
人生の可能性が広がる

Be the gate

プログラミングの楽しさを
知る入口になる

- ・楽しさが伝わる
- ・自分にもできると思える
- ・誰もが始められる

Be the path

自走できることにより
導ける道になる

- ・何を学ばいいかわかる
- ・必要な知識を得られる
- ・その知識を使って試行錯誤する経験ができる

An independent coder

誰にもつかつかでも自分で
動き、考えて乗り越えられる

- ・世の中にある問題を発見できる
- ・問題に対する解を生み出すことができる
- ・誰かの役に立つことができる

new doors

- ・自分でサービスリリース
- ・Webサイトを作成できる
- ・エンジニアとして入社
- ・受託で仕事がもらえる
- ・起業できる



「これから」を創っていく技術



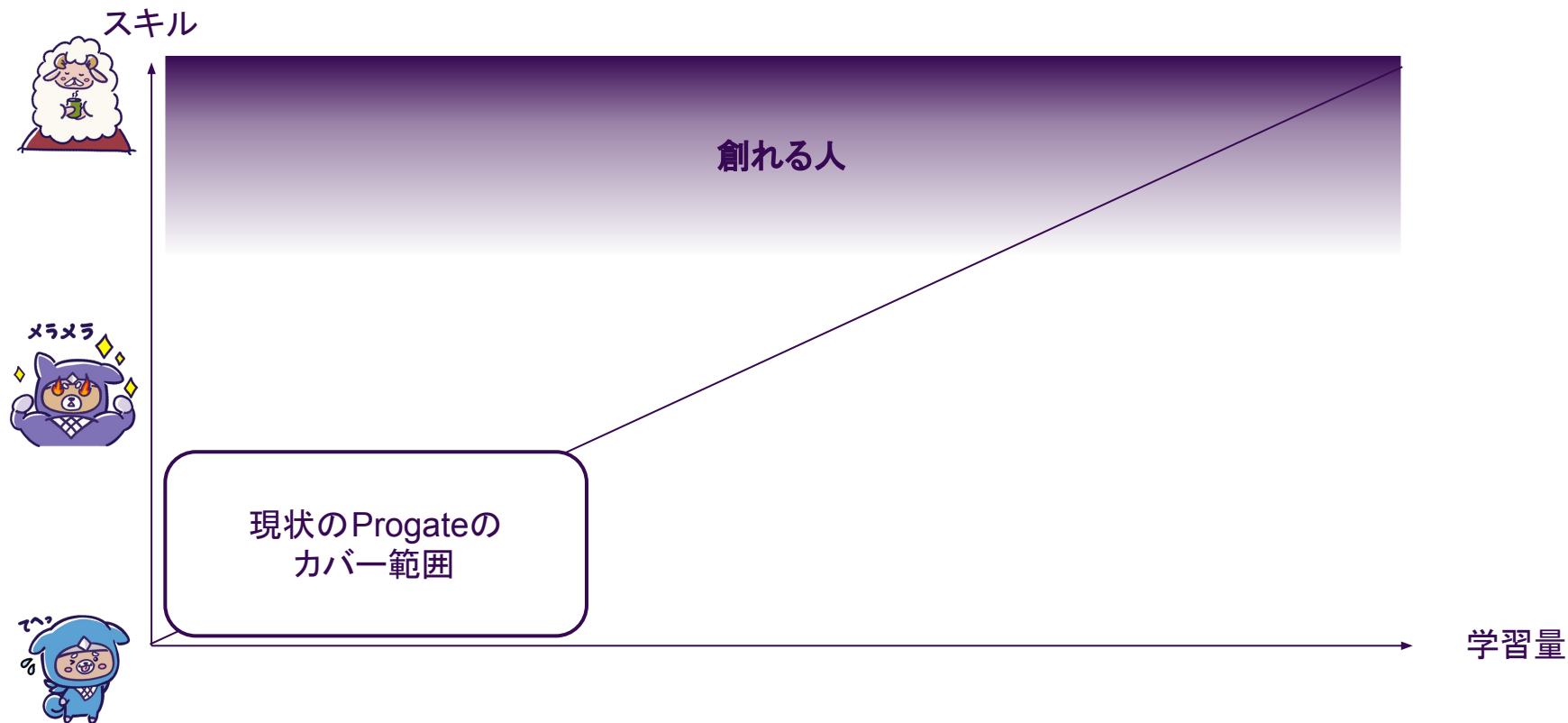
これからやっていきたいこと

最短で「創れる人」になってもらうための、
プログラミング学習における「知の高速道路」を整備していきたい。

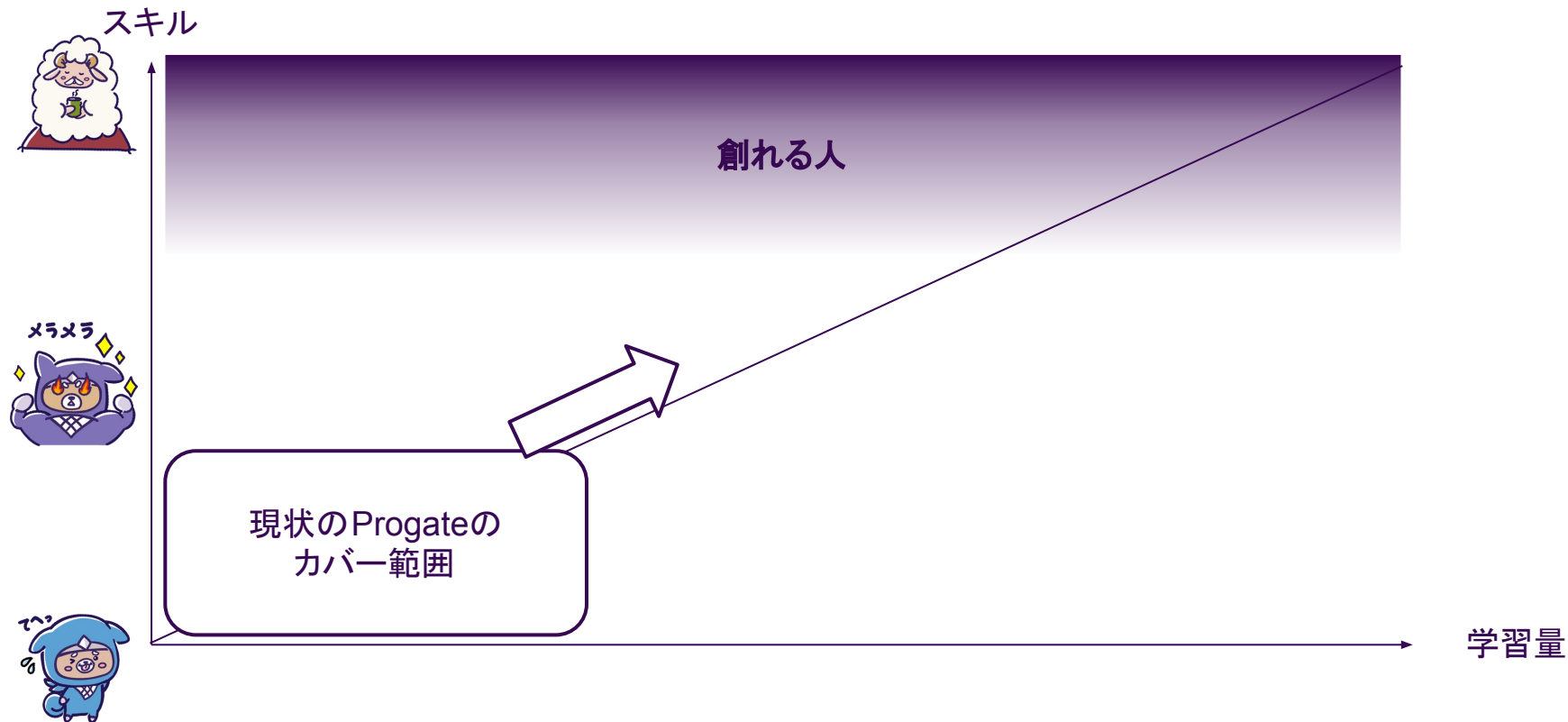
- ・基礎的なプログラミングを躓くことなくきちんと体験できる環境
- ・いろいろなコードを書いてみたり、作業をする試行錯誤の中で必要な知識や経験を身につけてもらえるような演習体験



Progate が提供できているもの・したいもの



Progate が提供できているもの・したいもの



Quest

Quest

期間限定のチャレンジコース



0%

道場コース

Quest ツイートアプリ

🕒 目安時間 1h30m

このレッスンではRailsを使ってWebアプリケーションのバグ修正にチャレンジします。バグ修正を通してコードリーディングやデバッグのスキルを身につけましょう。※Railsの知識は...

レッスンを始める

レッスン詳細へ



100%

道場コース

Quest ドローンメール

🕒 目安時間 3h

このレッスンではJavaScriptを使って経路探索にチャレンジします。なるべく短い経路を見つけるプログラムの作成を通じて、経路探索のアルゴリズムについて学びましょう。

復習する

レッスン詳細へ

デバッグ問題
「バグがあるので直してください」

アルゴリズム問題(TSP)
「すべての地点を一筆書きできる
短い経路を見つけてみましょう」



これまでの違いは？

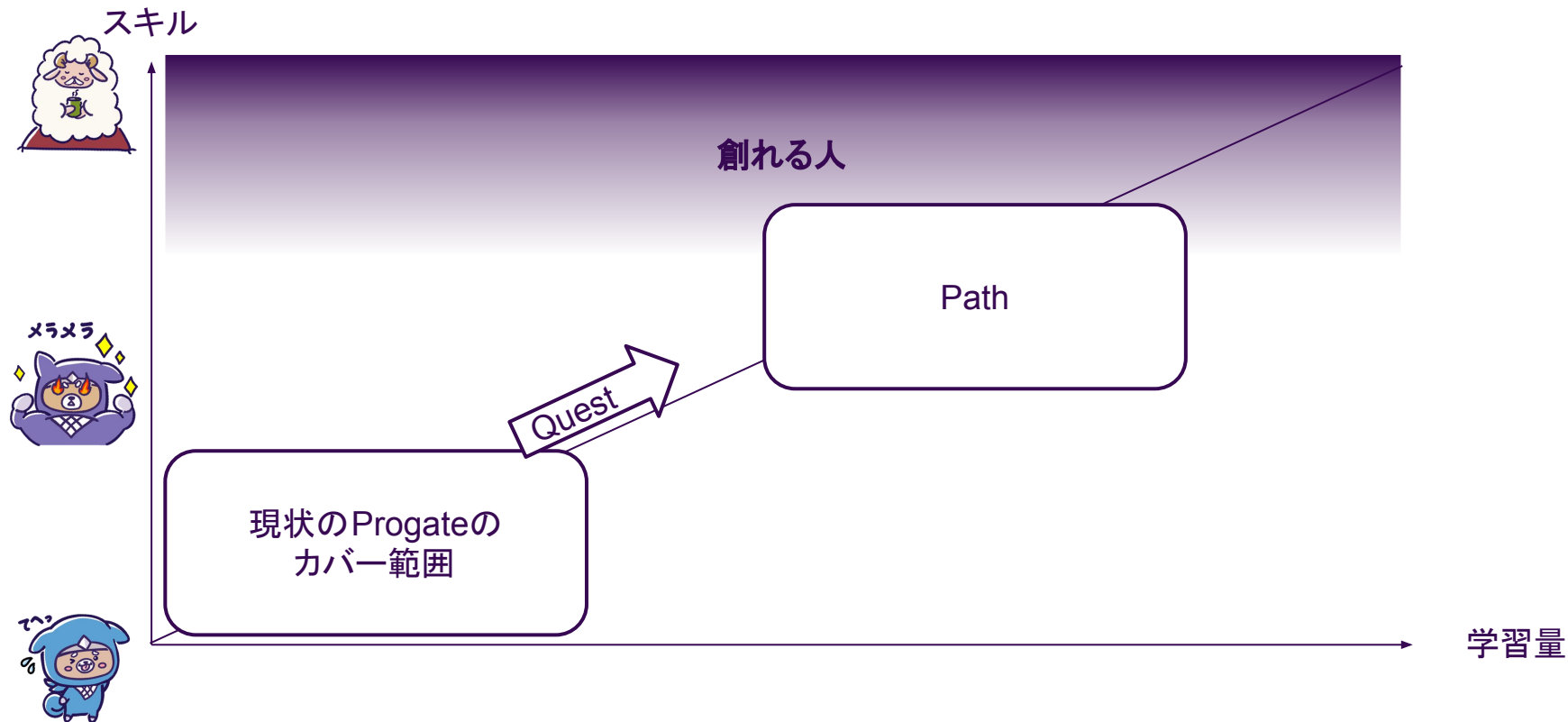
The screenshot shows a web browser window with a breadcrumb trail: "Quest 道場コース ツイートアプリ > ツイートアプリのバグ修正 > 1. いいね...". The page is divided into two main sections. On the left, there is a "手順" (Procedure) sidebar. The main content area contains a message: "報告されているエラーを修正しましょう。 「答えを見る」の中にヒントが用意されています。困ったら見てみましょう。 ※この問題に答えは用意されていません" (Please fix the error being reported. Hints are prepared in "View Answer". If you get stuck, please look at them. *There is no answer prepared for this problem). Below this is a light blue box with instructions: "まずはバグを再現してみよう" (First, let's try to reproduce the bug). Underneath is a section titled "バグ再現手順" (Bug reproduction procedure) with four steps: 1. ログイン (Login), 2. Accessing "/users/1/likes", 3. ログアウト (Logout), and 4. Accessing "/users/1/likes" again. On the right, a code editor shows the "application_controller.rb" file with the following Ruby code:

```
1 class ApplicationController < ActionController::Base
2   before_action :set_current_user
3
4   def set_current_user
5     @current_user = User.find_by(id: session[:current_user_id])
6   end
7
8   def authenticate_user
9     if @current_user == nil
10      flash[:notice] = "ログインが必要です"
11      redirect_to("/login")
12    end
13  end
14
15  def forbid_login_user
16    if @current_user
17      flash[:notice] = "すでにログインしています"
18      redirect_to("/posts/index")
19    end
20  end
21 end
```

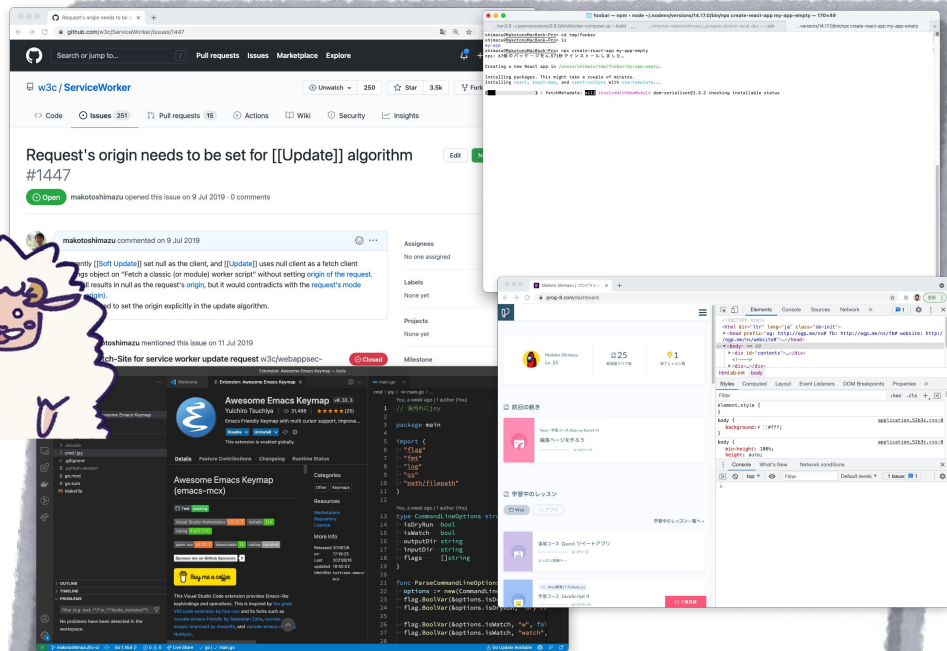
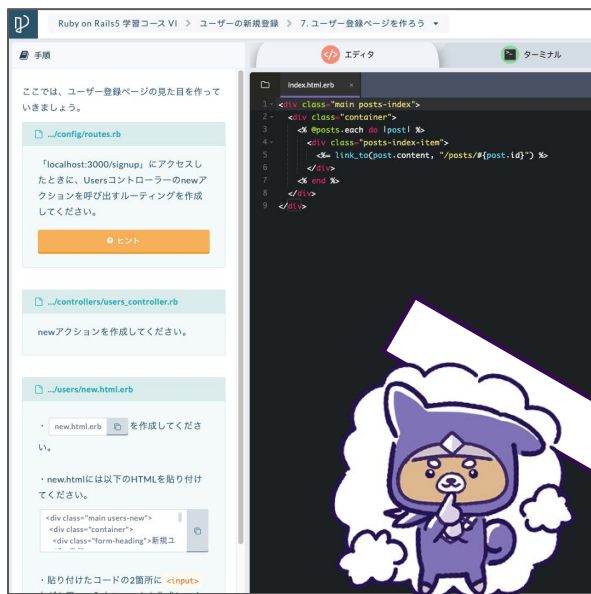
- ・課題(バグを直す)をどう対処するか、手順を自分で考える必要がある
- ・ヒントは与えられ、自分で検索しながら学習していくスタイル



Progate が提供できているもの・したいもの



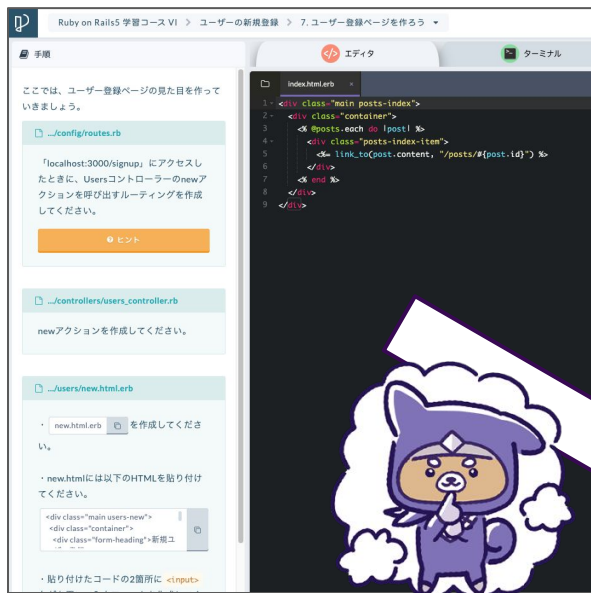
実践的な、試行錯誤しながら学べる演習とはなにか？



普段僕たちが行っているような開発が出来るまでに何を体験する必要があるだろうか？



実践的な、試行錯誤しながら学べる演習とはなにか？



問題の分割

環境構築

エラーの分析

きれいなコーディング

ツールの操作

コードの理解

デバッグ

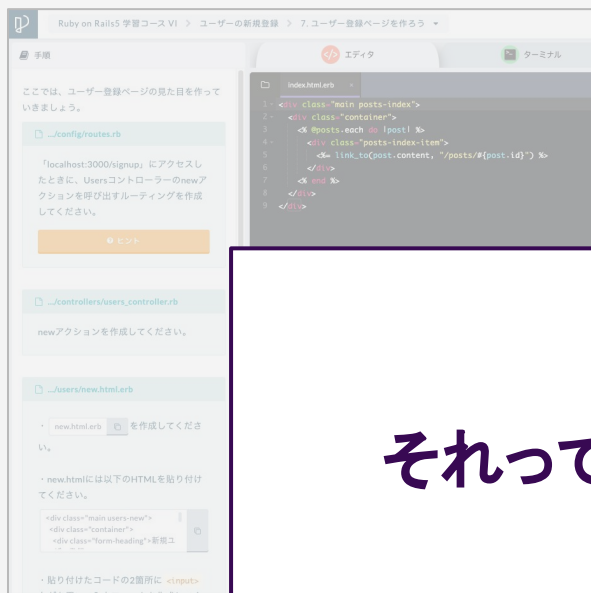


普段僕たちが行っているような開発ができるまでに何を体験する必要があるだろうか？

プログラミングももちろんだけど、それ以外にもたくさんの要素がある！



実践的な、試行錯誤しながら学べる演習



問題の分割

環境構築

それってどんな演習なら体験できる？



の分析

普段僕たちが行っているような開発が出来るまでに何を体験する必要があるだろうか？

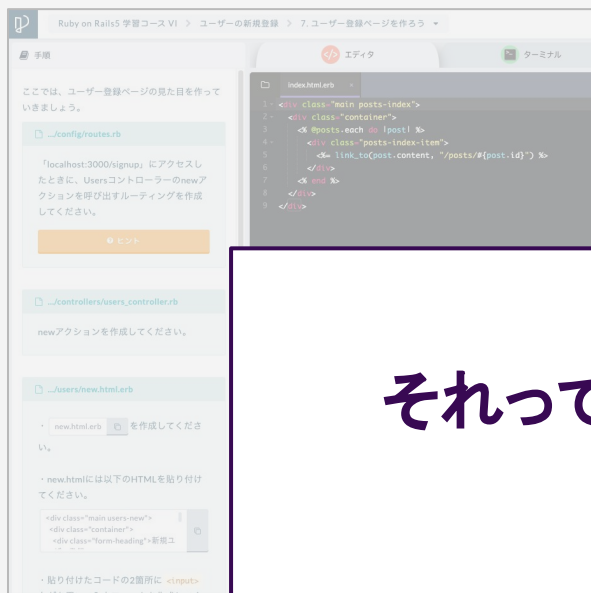
プログラミングももちろんだけど、それ以外にもたくさんの要素がある！

ツールの操作

コードの理解

デバッグ





問題の分割

環境構築

の分析

それってどんな演習なら体験できる？



→ 鋭意作成中！

デバッグ

ツールの操作

コードの理解

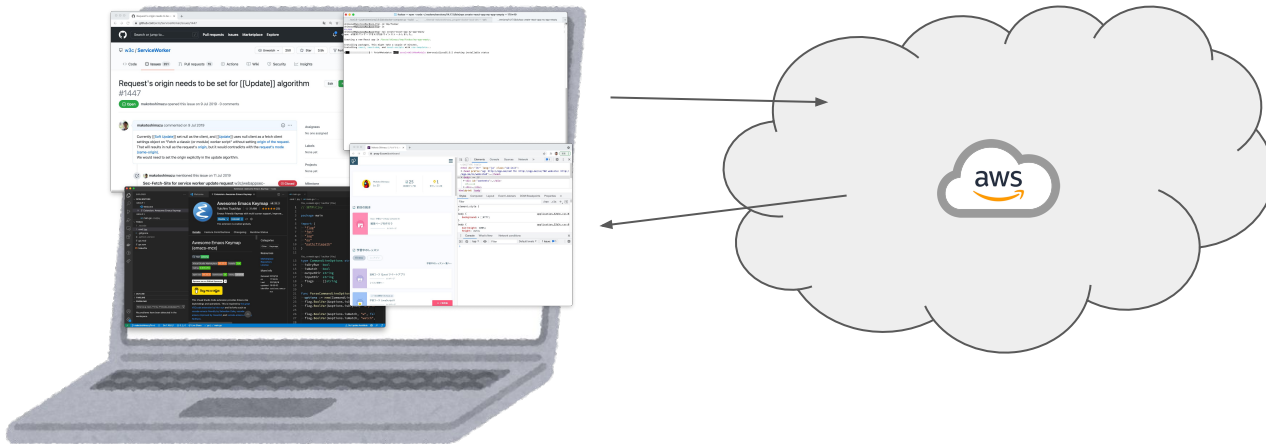
普段僕たちが行っているような開発が出来るまでに何を経験する必要があるだろうか？

プログラミングももちろんだけど、それ以外にもたくさんの要素がある！



作っているものの一例: ユーザー体験のサポートのためのコマンドラインツール

- 補助輪としてユーザー体験を支えるツール
 - ユーザーさんが簡単に環境構築できるようサポート
 - テストを走らせる
- Goを採用
 - クロスコンパイルが簡単
 - バックエンドとの通信のための gRPCのサポートが充実している
 - Cobraのようなコマンドラインツールをつくる使い勝手のよいライブラリもある



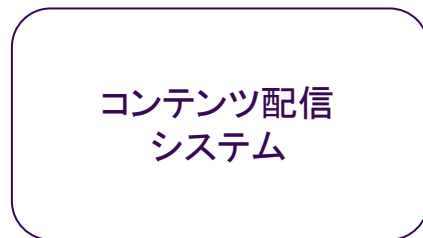
作っているものの一例:コンテンツ管理システム

- エンジニアとしては、カスタマーとしてユーザーさんの他に社内のコンテンツプランナーやi18nチームもいる

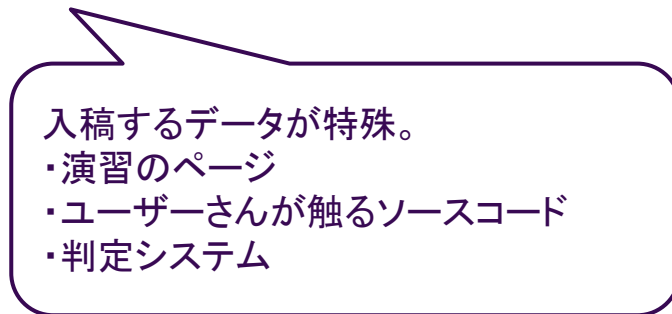
コンテンツプランナー



i18nチーム



ユーザーさん



Progate では、2つの軸を据えてプロダクト開発を行っています。

- ・躓くことなくプログラミングの世界に触れることができるプロダクト
- ・「創れる人」になるための力をつけることができるプロダクト

Progate エンジニアチームとしては、

よりよいユーザー体験を提供するために、必要な技術的な課題を日々解いています。



採用特設サイトを公開しています！

- エンジニアのインタビュー記事やイベントレポートなど掲載
- 是非一度ご覧ください！



<https://bit.ly/progate-aws-devday-2021>

＼積極採用中！／



We are hiring!

**Progateで働くことに興味のある方
ぜひ一度ご覧ください！**

Progate エンジニア 採用情報



カジュアルに話を聞いてみたい方はrecruit@prog-8.comまで！

Progateのエンジニアについてもっと知りたい方はこちら！

Progate Tech Blog



会社について詳しく知りたい方はこちら！

Progate Speaker Deck



Gateサービスを提供する上での難しいポイント

- ポイントに絞った学習
 - step by step の学習
 - 余計なことを考えなくてもいい UX
 - Slide/Instruction構成
 - 演習ごとにカスタマイズされた演習画面、ブラックアウト



Gateサービスを提供する上での難しいポイント

- ユーザーのコードを「実行」する
 - IDE周りの構成
 - Swarm on EC2
 - WebSocket
 - ファイル同期



Gateサービスを提供する上での難しいポイント

- ユーザーのコードを「判定」する
 - Judge/ImageJudge周りの構成
 - Judge
 - クライアントサイドJS
 - 独自ルールに基づいた静的判定、出力判定、rspec判定など
 - ImageJudge
 - OpenCVを用いたimage diff



Gateサービスを提供する上での難しいポイント

- グローバル展開対応
 - 多言語対応
 - コンテンツ作成コスト
 - 認証・決済基盤
 - 各国に適した決済手法の展開
 - それらを抽象化して認可する認証基盤
 - 東京regionからの配信最適化
 - edgeロケーションの活用、js最適化



Gateサービスを構成しているアーキテクチャ

- 全体俯瞰図
- 最近の進化
 - ログサービスの切り出し
 - アプリケーションイベントを分析データとして活用する基盤
 - 配信高速化
 - DevOps (ChatOps) の進化



Pathの話

- 今考えている案、みたいな枕詞？
- 技術選定のトピック(なぜgoを採用したのか)
- コンテンツデータ管理
 - ローカル開発体験を提供するための手法？
 - 既存サービスとの認証連携などの課題？
 - 学習体験としての工夫？



Pathを作る上でAWSでよかったこと

- アップデートが早いのが嬉しい
 - gRPCを技術選定で採用しようとしたときに、AWSのいろんなサービスがgRPCに対応し始めているのでうれしい
 - ALBとか

