The Art of SLOs

*混沌のなかにこそ ~~勝機~~ **信頼性**あり*

– 孫子の兵法 (The Art of War)

原文: https://cre.page.link/art-of-slos-slides

ようこそ！

恥ずかしがらないで・・・

となりの人に **やあ** ってあいさつしてみよう

参加者の約束

- 私たちは**学ぶ**ためにここにいる
- **質問**をする (手を挙げて)
- **ひとりずつ**しゃべる
- みんな**好意的**だと考える
- 「何についてしゃべってるんだっけ？」

もくじ

- 専門用語
- なぜサービスには SLO が必要なの？
- エラー バジェットを使う
- 良い SLI を選ぶ
- SLO と SLI を作る

Service Level Indicator

サービスの **信頼性** の **定量化できる** 尺度

Service **L**evel **O**bjectives

SLI に対する **信頼性の目標**

ユーザー？お客様？

お客様 というのは、サービスに **直接お金を払う** ユーザー

サービスには SLO が必要

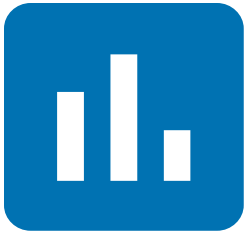
信じられないって？

「SLO を導入してから、運用と開発チームとの**関係**が、
わずかだが注目に値するほど改善した」

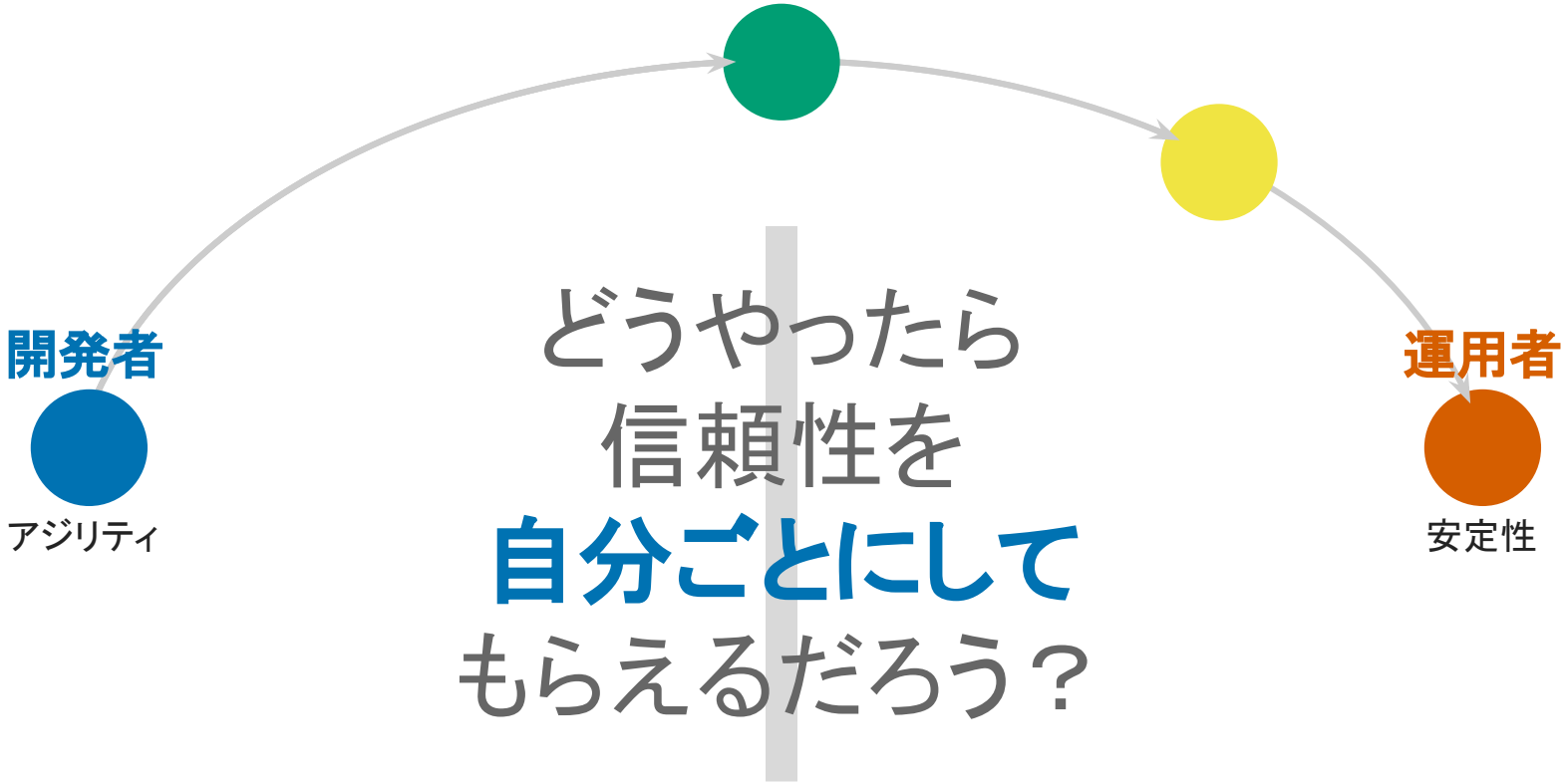
– Ben McCormack, Evernote; The Site Reliability Workbook, Chapter 3

「... 明確に定義しないことには、いい仕事をすることは難しい。
「いい仕事」を定義するために SLO は**ことばを与えてくれる**」

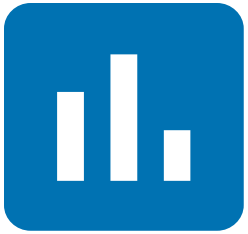
– Theo Schlossnagle, Circonus; Seeking SRE, Chapter 21



どんなシステムでも
最も重要な機能は
信頼性



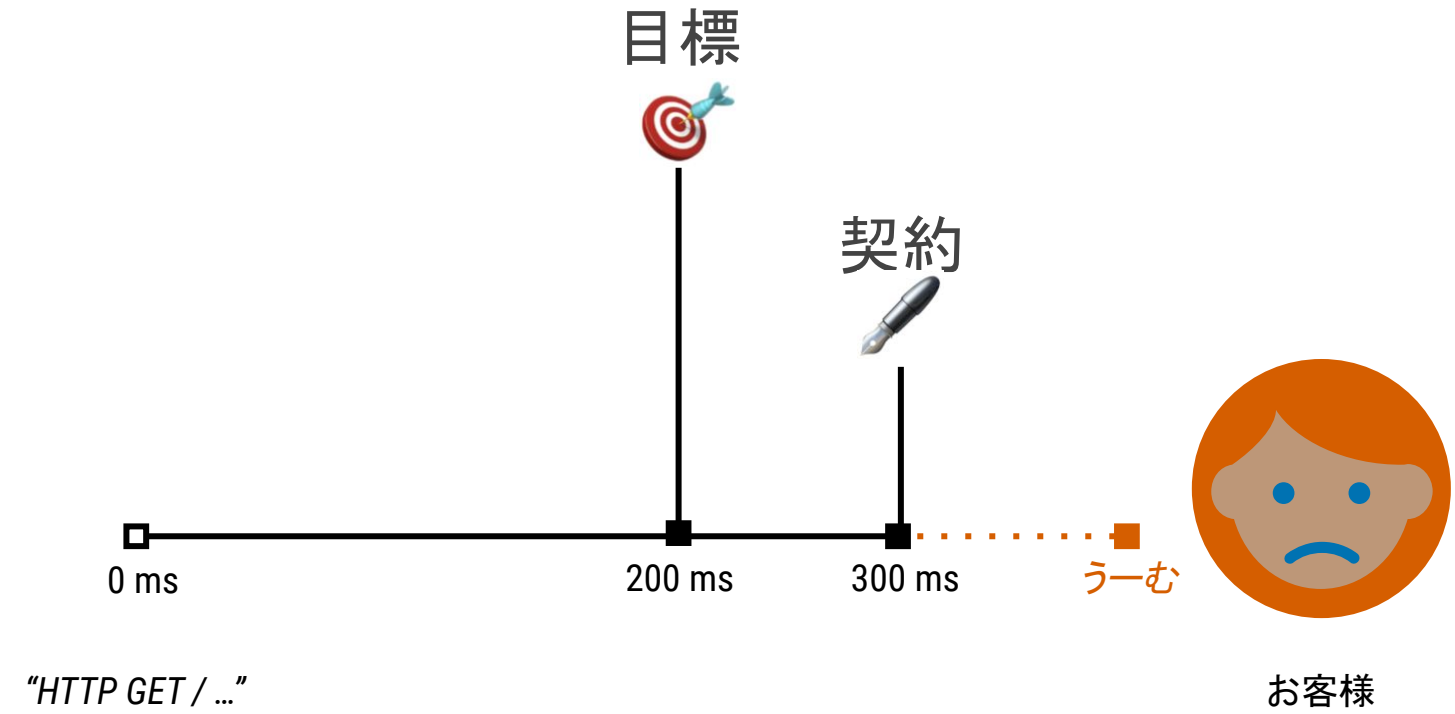
どうやったら
信頼性を
自分ごとにして
もらえるだろう？



みんながサービスの
望ましい信頼性に
ついて納得できる
原理に基づいた方法

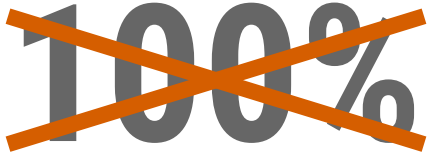
「信頼性が高い」って？

Netflix, Google 検索, Gmail, Twitter を思い浮かべてみて…
サービスがちゃんと「動いてる」って、どうやって見分ける？


目標
契約
0 ms
200 ms
300 ms
うーむ
“HTTP GET / …”
お客様

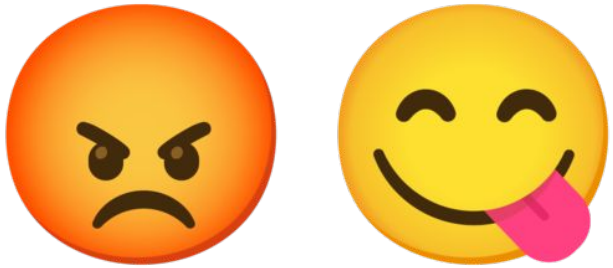
ついてきてる？

どういうときに**もっと高い信頼性**が
必要になるんだろう？



基本的に**どんなときでも** 100% は**誤った**信頼性の目標です

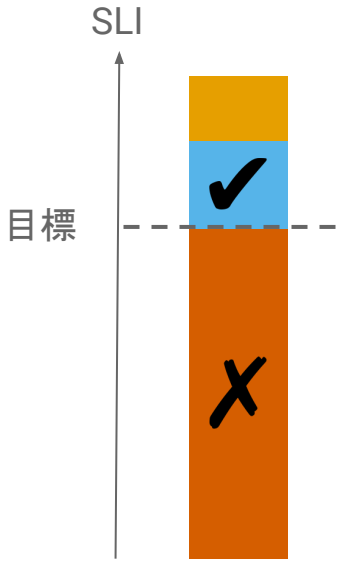
— ***Benjamin Treynor Sloss***, *VP 24x7, Google;* Site Reliability Engineering, Introduction



SLO は、**ギリギリ達成** できればサービスの**典型的なお客様**が
満足するような、性能や可用性のレベルにすべき

「SLO を満たしている」⇒「お客様は満足している」
「お客様が不満になる」⇒「SLO を満たしていない」

SLO を測って
目標を**わずか**
に超えるよう
にがんばる...



最新版: 10.17

バージョン 10.17 での変更点:
スペースバーを押したままにしても、CPUが過熱しなくなりました。

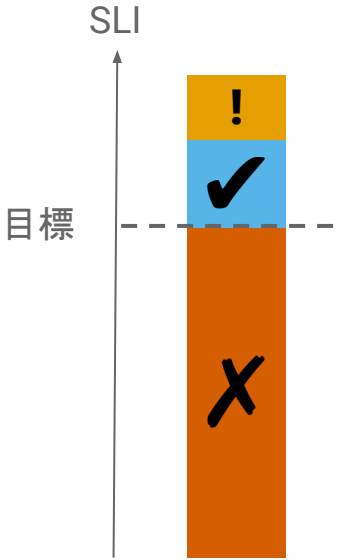
コメント:

昔からのユーザー4 は書きました:
このアップデートは俺のワークフローをぶっ壊した！俺の Ctrl キーは手が届きにくいから、代わりにスペースバーを押したままにして、急激な温度上昇を Ctrl として解釈するように emacs を設定してるんだよ！

管理者 は書きました:
それはやめたほうがいい。

昔からのユーザー4 は書きました:
聞けよ、俺の設定はちゃんと動いてたんだ！スペースバーで加熱できるオプションを追加しろ！

"Workflow", *Randall Munroe, XKCD*
Source: https://xkcd.com/1172/



…でも高くしすぎない。ユーザーが**それに頼ってしまう**

エラー バジェット

SLO は**許容可能なレベル**の信頼性の欠如を意味する

*これは**割り当てることができる予算**になる*

実装のしくみ

決められた**期間**(例: 28 日)における SLO の**パフォーマンス**を評価
残った予算がエンジニアリング作業の**優先順位を決める**

ITIL での近似

SLO を満たしている → ほとんどの運用作業は**標準的な変更**

SLO を満たさなく**なりそう** → **通常の変更**に戻す

(ごめん、「標準的」と「通常」の違いすらわからない …)

エラー バジェットは
何に**使えば**いいの？

エラー バジェットは

- 新しい機能の**リリース**
- 予定されたシステムの**変更**
- 避けられないハードやネットワークの**障害**
- 計画**停止**
- リスクの高い**実験**

などに使える

エラー バジェットの利点

- **開発と SRE の共有のインセンティブになる**
  イノベーションと信頼性のいいバランスを見つけられる
- **開発チームが自分でリスクを管理できる**
  エラーバジェットをどう使うかを開発チームが決める
- **非現実的な信頼性の目標は魅力的でなくなる**
  そういう目標はイノベーションの速度を落とす
- **開発チームが自己管理するようになる**
  エラーバジェットは開発チームにとっても貴重な資源になる
- **システム稼働時間に対する責任を共有する**
  インフラ障害もエラーバジェットを消費する

まだついてきてる？

演習

信頼性の原則

社員のみなさんへ

先日のシステム障害に対する否定的な報道をうけて、私たち全員が、サービスの信頼性についてもっと真剣に受け止める必要があると確信しました。
この公開書簡では、将来のみなさんの正しい意思決定を促すために、信頼性に関する 3 つの原則を示したいと思います。

最初の原則は、ユーザーに関することです。私たちは今回ユーザーを失望させてしまいましたが、ユーザーはもっと　優遇されるべきです。ユーザーは、私たちのサービスを利用したら、幸せになるべきなのです！

私たちのビジネスは...

1. 信頼性に対して財政的なコミットをすることで、ユーザーの信頼を取り戻す必要があります。
2. ユーザーが将来のシステム障害を許容、あるいは楽しめるような方法を見つける必要があります。
3. 機能を追加する以前に、信頼性に対するユーザーの期待に応える必要があります。
4. ユーザーを幸せにする機能をもっと早く作る必要があります。
5. 今後二度と障害を起こすようなことがあってはなりません！

2番目の原則は、サービスの構築方法に関するものです。信頼性を組み込むために、開発プロセスを変更する必要があります。

私たちのビジネスは...

1. 短い反復で、早く失敗し、エラーを早期に発見する必要があります。
2. リスク軽減のため運用チームが新機能の設計を担う必要があります。
3. 信頼性が高いことが確認できた場合にのみ、新機能を公開する必要があります。
4. 制御された小さなステップでソフトウェアを構築してリリースする必要があります。
5. システムの信頼性が低い場合に、機能リリースの頻度を下げる必要があります。

3番目の原則は、私たちの運用の実践に関するものです。今日私たちがしていることはうまくいきません。運用チームが燃え尽きており、インシデント率が高すぎます。改善するには、別のやりかたで行う必要があります。

私たちのビジネスは...

1. 運用チームと開発チームが信頼性の責任を共有する必要があります。
2. 運用作業とチームの優先順位を信頼性の目標に結び付ける必要があります。
3. 運用負荷を削減するためにシステムをより強固にする必要があります。
4. 障害が発生するのを防ぐために、全リリースで運用チームに拒否権を与える必要があります。
5. Twitterでの否定的な苦情を運用チームに直接通知する必要があります。

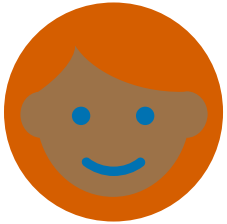
これらの 原　則 を 実　践 するために、Google からいくつかのアイデアを借ります。次のステップは、サービスの SLO をいくつか 定　義 し、それらに 対 するパフォーマンスの 測 定 を 開 始 することです。

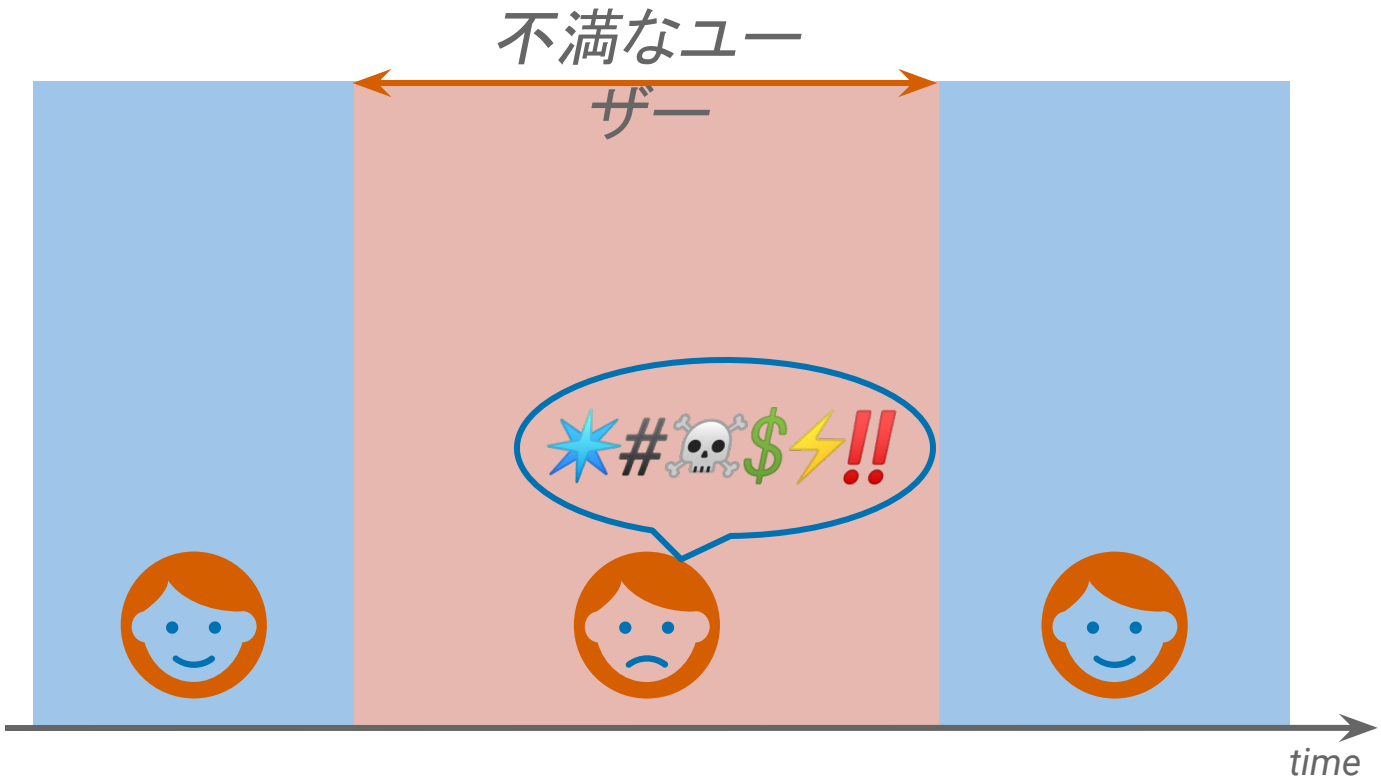
読んでくれてありがとう！

Eleanor Exec, CEO

休憩！

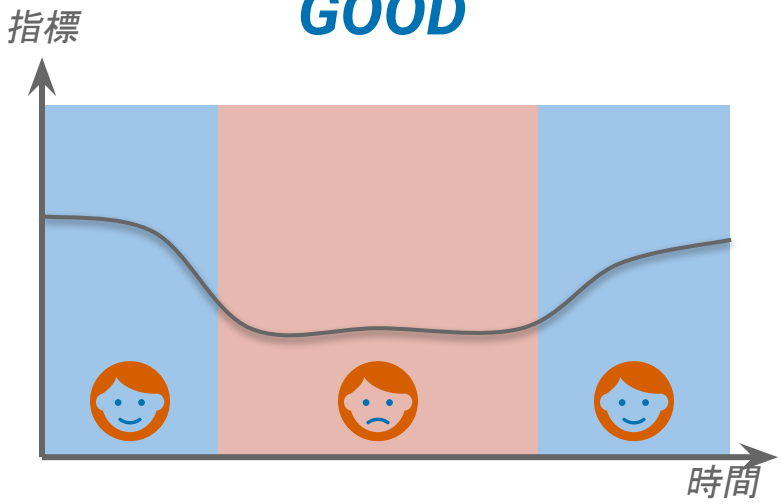
いい SLI を選ぶ



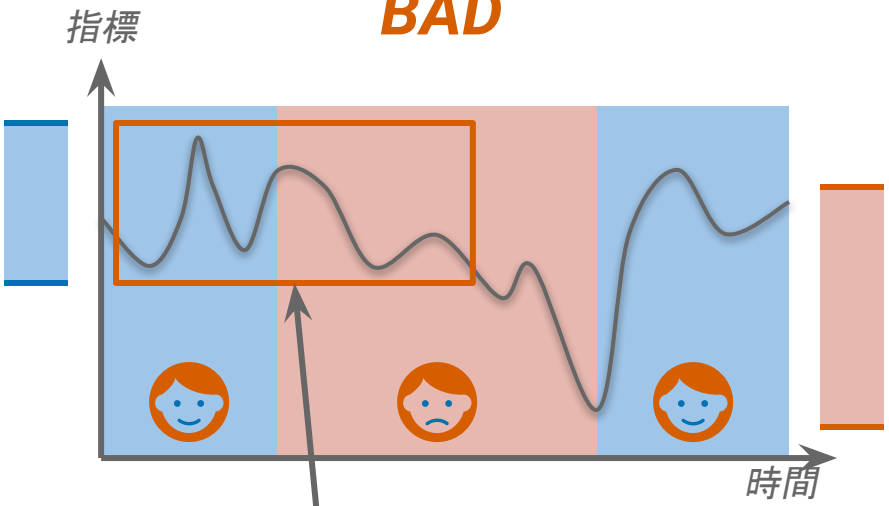

不満なユーザー
✴️#☠️$⚡‼️
time

BAD

GOOD

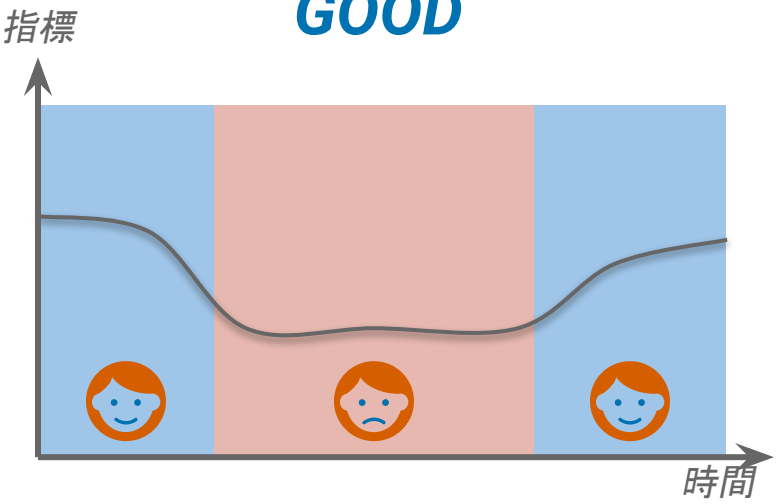


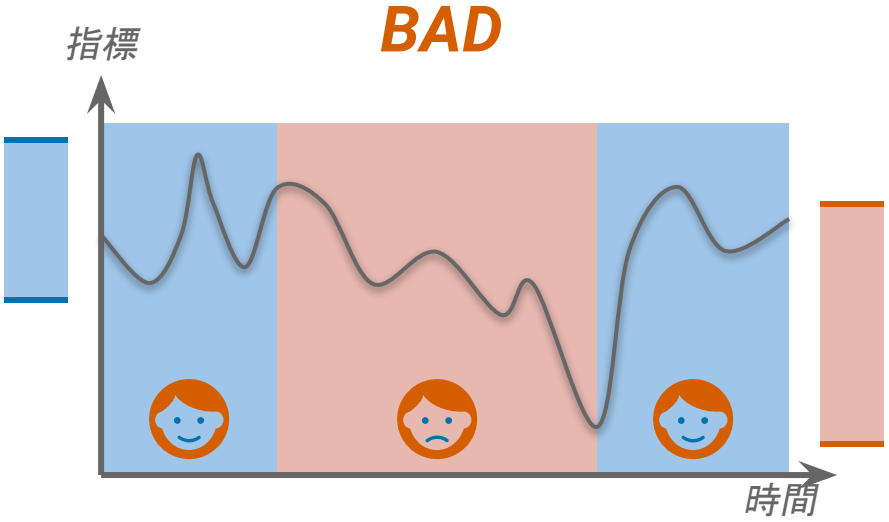
BAD

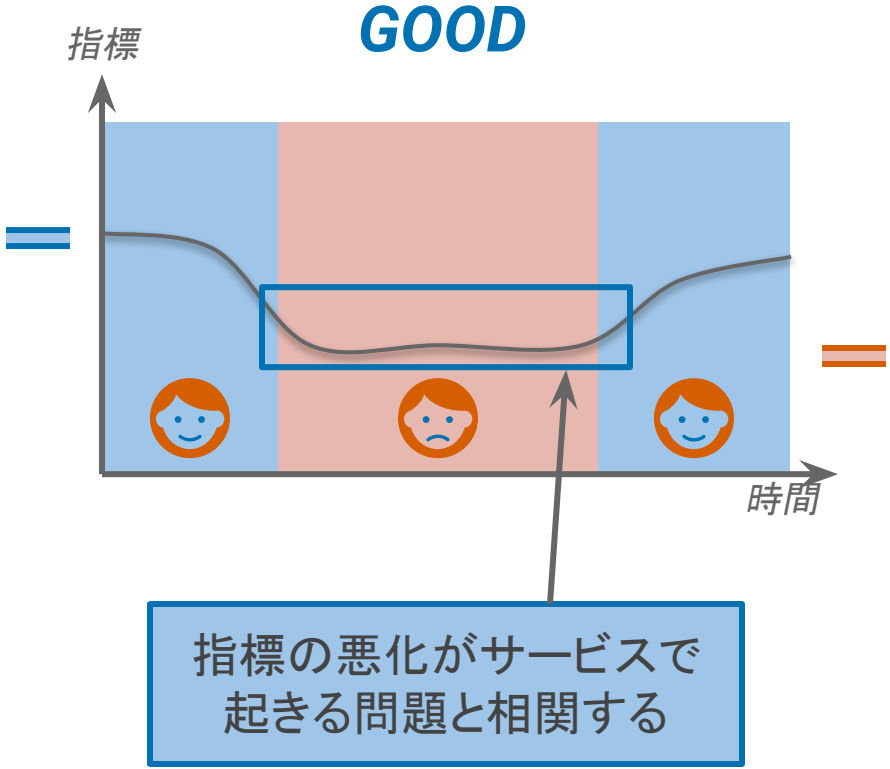


上下していて指標の悪化がわかりにくい

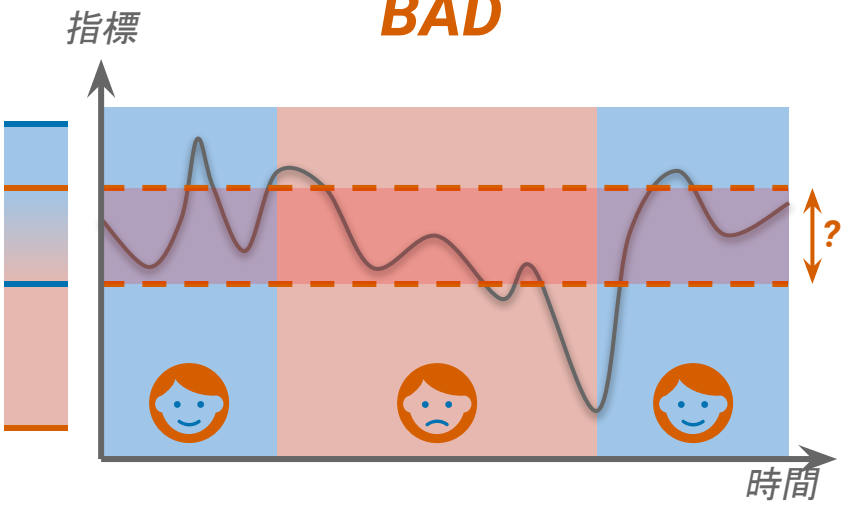
GOOD




BAD
指標
時間

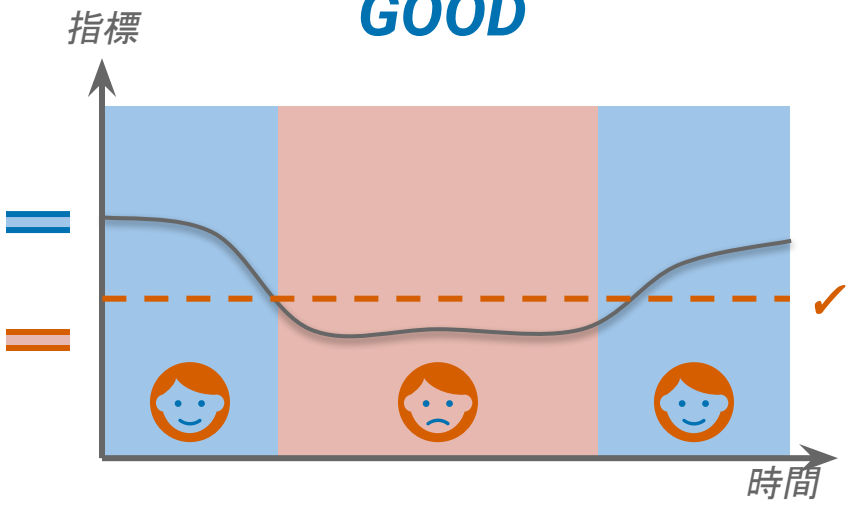

GOOD
指標
時間
指標の悪化がサービスで
起きる問題と相関する

BAD

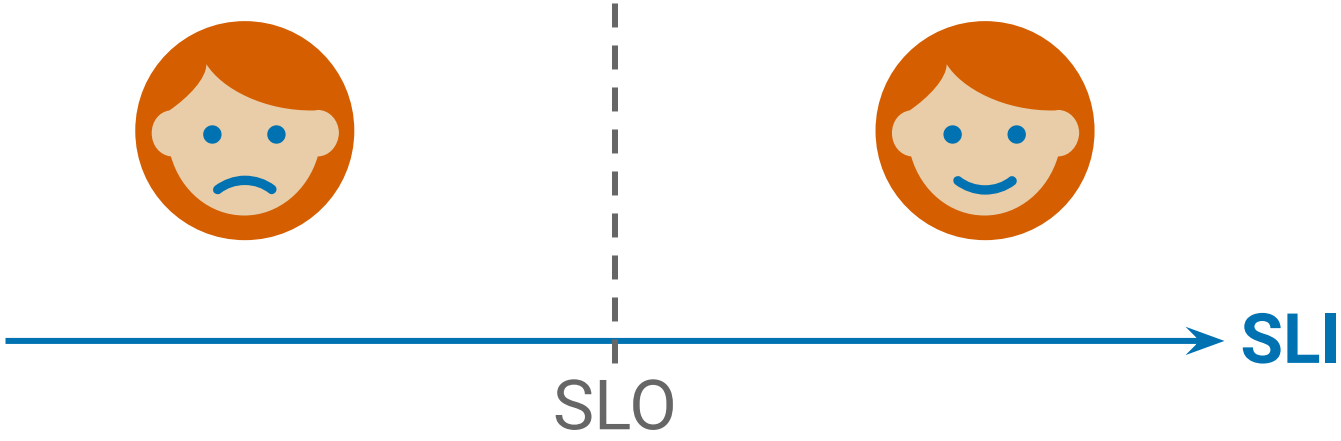


指標の S/N 比が悪い

GOOD



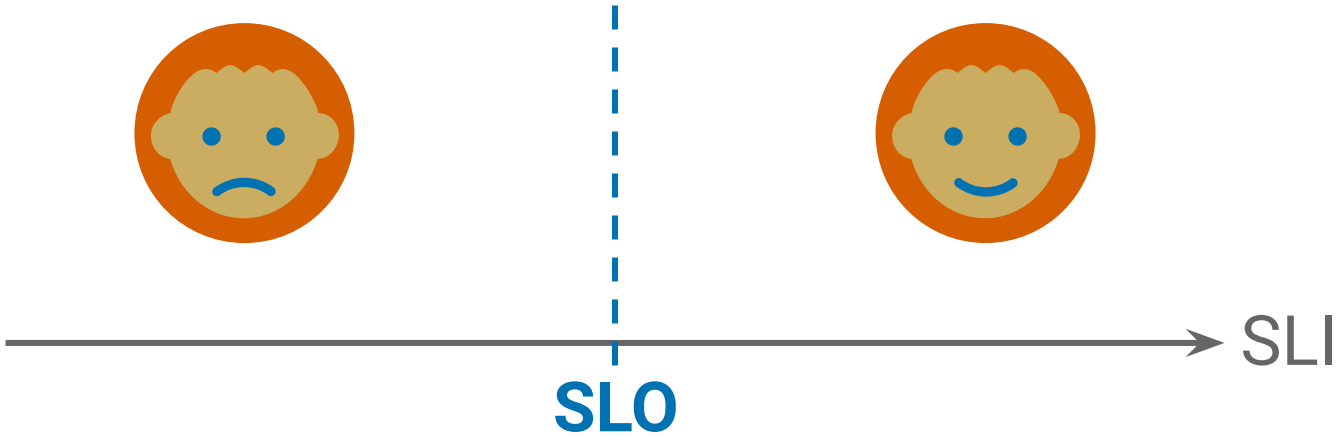
指標は良好な
S/N 比を示す

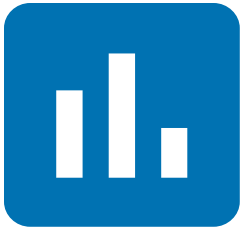

SLI
SLO

$$\text{SLI} : \left(\frac{\text{良いイベント}}{\text{有効なイベント}} \right) \times 100\%$$

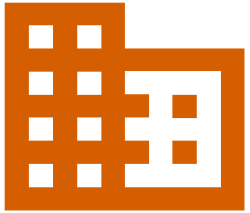
3–5 SLIs*

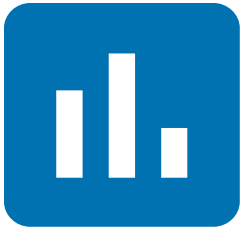
* ユーザージャーニーあたり


SLI
SLO

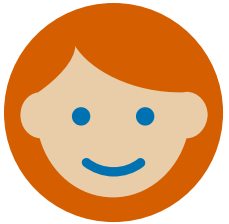


ビジネスの視点からみて
どれくらいの
パフォーマンスが
必要だろう？

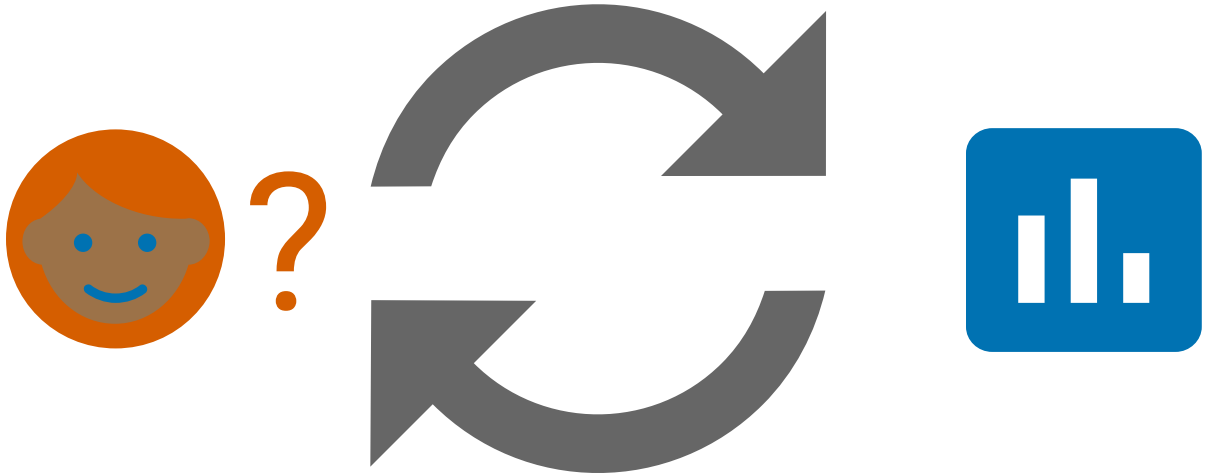




ユーザーの期待は
過去のパフォーマンスに
ものすごくひっぱられる



継続的な

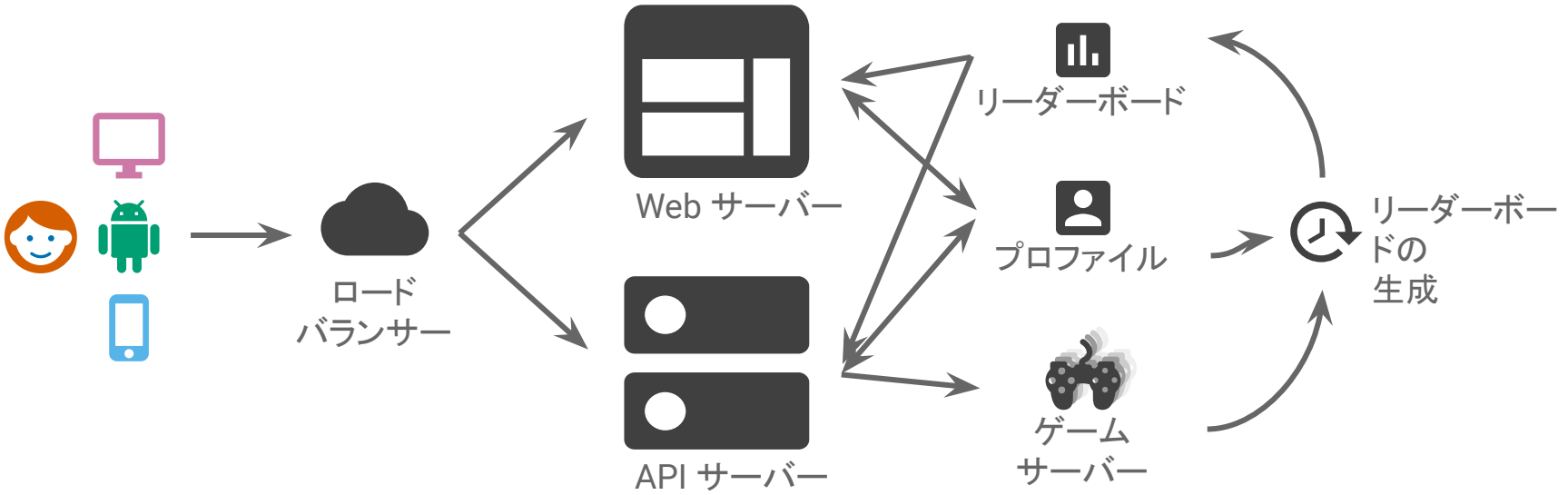


改善

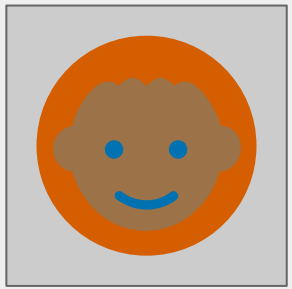
おなかいっぱい？

SLO と SLI を作る

みんなのゲーム: Fang Faction



https://fangfactiongame.com/profile/someuser



SomeUser
かえる族
Faction Score: **31337**
Midwest Canyon

1. Tri-Bool **65535**
2. Tri Repetae **61995**
3. Triassic Five **52391**
4. Tricksy Hobbits **37164**
5. かえる族 **31337**
6. Trite Examples **29243**

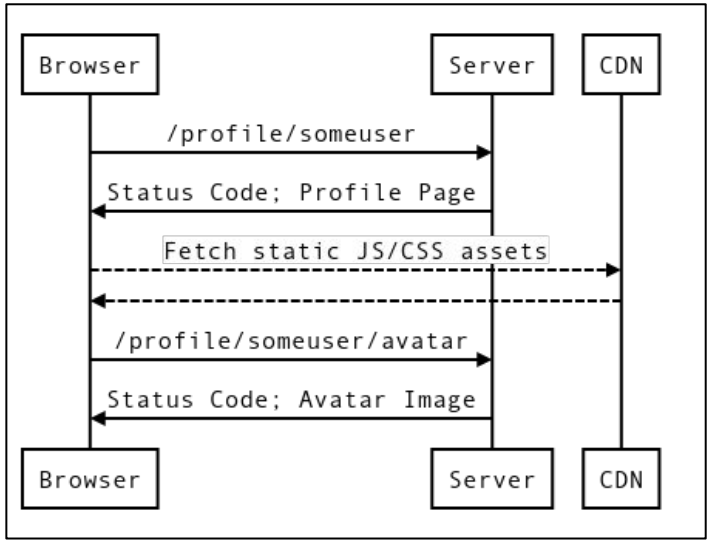
SomeUser のプロファイル

チーム: かえる族

リーダー: SomeUser

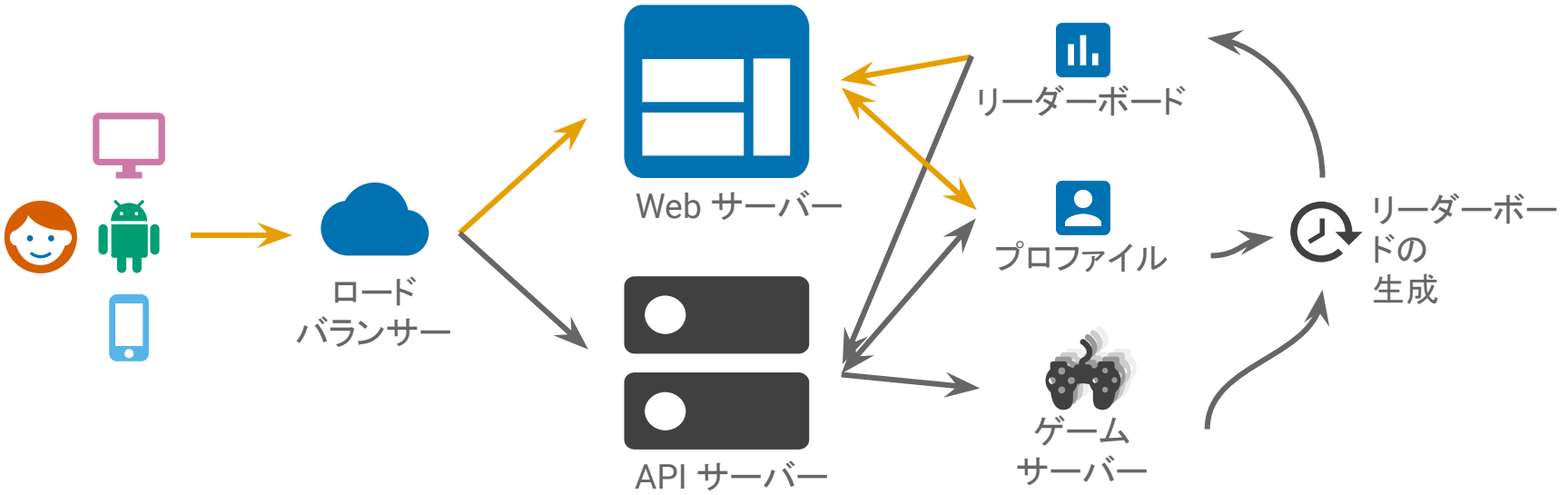
メールアドレス: user@example.com

更新



Google

プロファイル ページの読み込み



SLI Menu

リクエスト / レスポンス	可用性 レイテンシー 品質
データ処理	カバレッジ 正確性 鮮度 スループット
ストレージ	スループット レイテンシー

可用性

プロファイルページの読み込みに**成功する**

レイテンシー

プロファイルページの読み込みが**速い**

可用性

プロファイルページの読み込みに**成功する**

- **成功**をどう定義するか
- 成功/失敗はどこに**記録**されるか

レイテンシー

プロファイルページの読み込みが**速い**

- **速さ**をどう定義するか
- タイマーはいつ**開始/停止**するか

可用性

プロファイルページの読み込みに**成功する**

- **成功**をどう定義するか
- 成功/失敗はどこに**記録**されるか

有効なリクエストのうち、**正常に**処理された割合

レイテンシー

プロファイルページの読み込みが**速い**

- **速さ**をどう定義するか
- タイマーはいつ**開始/停止**するか

有効なリクエストのうち、しきい値よりも**速く**処理された割合

可用性

プロファイルページの読み込みに**成功する**

- **成功**をどう定義するか
- 成功/失敗はどこに**記録**されるか

有効なリクエストのうち、**正常に**処理された割合

レイテンシー

プロファイルページの読み込みが**速い**

- **速さ**をどう定義するか
- タイマーはいつ**開始/停止**するか

有効なリクエストのうち、しきい値よりも**速く**処理された割合

可用性

プロファイルページの読み込みに**成功する**

- **成功**をどう定義するか
- 成功/失敗はどこに**記録**されるか

/profile/{user} または **/profile/{user}/avatar** に対する **HTTP GET リクエスト**のうち、**正常に**処理された割合

レイテンシー

プロファイルページの読み込みが**速い**

- **速さ**をどう定義するか
- タイマーはいつ**開始/停止**するか

/profile/{user} に対する **HTTP GET リクエスト**のうち、しきい値よりも**速く**処理された割合

可用性

プロファイルページの読み込みに**成功する**

- **成功**をどう定義するか
- 成功/失敗はどこに**記録**されるか

/profile/{user} または **/profile/{user}/avatar** に対する **HTTP GET** リクエストのうち、**正常に**処理された割合

レイテンシー

プロファイルページの読み込みが**速い**

- **速さ**をどう定義するか
- タイマーはいつ**開始/停止**するか

/profile/{user} に対する **HTTP GET** リクエストのうち、しきい値よりも**速く**処理された割合

可用性

プロファイルページの読み込みに**成功する**

- **成功**をどう定義するか
- 成功/失敗はどこに**記録**されるか

/profile/{user} または **/profile/{user}/avatar** に対する **HTTP GET** リクエストのうち、ステータスが ***200番台***、***300番台*** または ***400 番台*** **(excl. 429)** を返す割合

レイテンシー

プロファイルページの読み込みが**速い**

- **速さ**をどう定義するか
- タイマーはいつ**開始/停止**するか

/profile/{user} に対する **HTTP GET** リクエストのうち、***X*** **ms** の**範囲内で**処理されたリクエストの割合

SLI Menu

測定の戦略

アプリケーションレベルのメトリックス

ログ 処理

フロント エンド インフラのメトリックス

人工的なクライアント/データ

クライアント サイド への埋め込み

可用性

プロファイルページの読み込みに**成功する**

- **成功**をどう定義するか
- 成功/失敗はどこに**記録**されるか

ロードバランサーで測定したステータスが **/profile/{user}** または **/profile/{user}/avatar** に対するすべての **HTTP GET** リクエストのうち、***200 番台***、***300 番台***または ***400 番台*** **(excl. 429)** を返す割合

レイテンシー

プロファイルページの読み込みが**速い**

- **速さ**をどう定義するか
- タイマーはいつ**開始/停止**するか

ロードバランサーで測定した、**/profile/{user}** に対する **HTTP GET** リクエストのうち、***X* ms** の**範囲内にレスポンス全体**を送信したものの割合

演習

ポストモーテム

可用性

ロードバランサーで測定したステータスが、**/profile/{user}** または **/profile/{user}/avatar**に対する**HTTP GET** リクエストのうち、***200番台***、***3XX*** または ***4XX* (excl. 429)** を返す割合

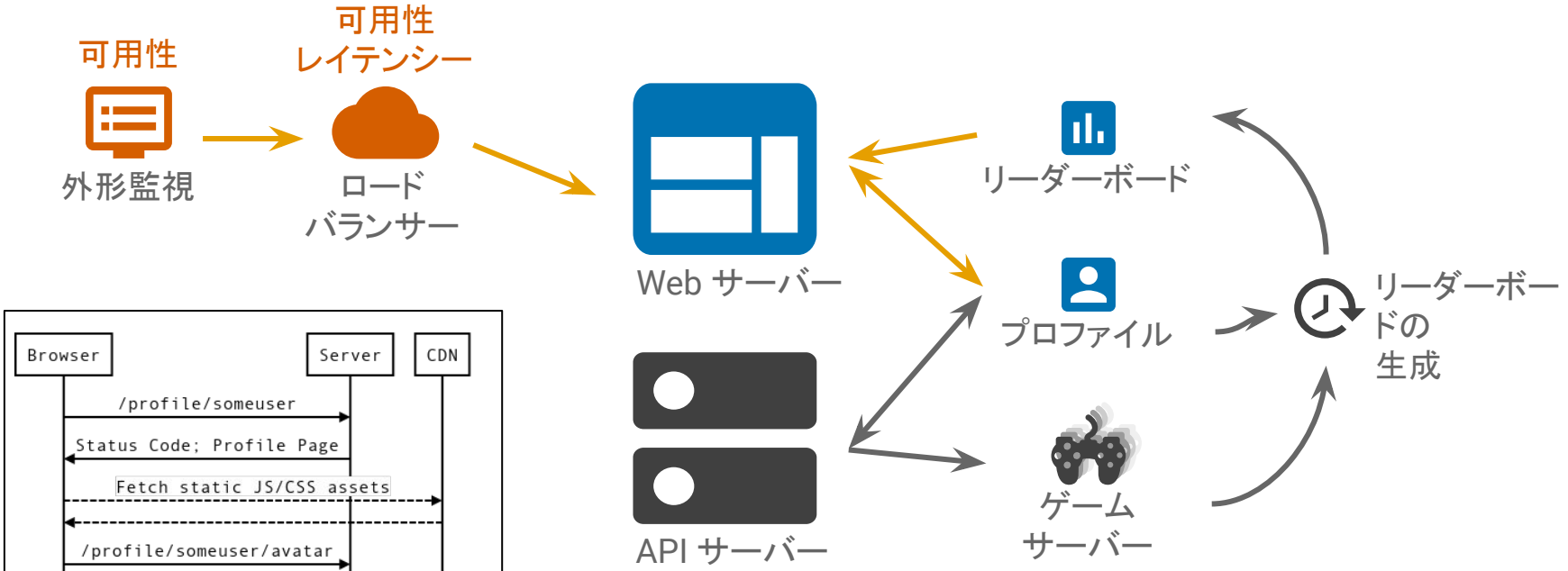
および

5秒ごとに**外形監視**で測定される **/profile/prober_user** 及び**リンクのある全てのリソース**に対する**HTTP GET**リクエストのうち **"ProberUser"を含む有効なHTML**を返す割合

レイテンシー

ロードバランサで測定した、**/profile/{user}** に対する **HTTP GET** リクエストのうち、***X* ms** の**範囲内にレスポンス全体**を送信したものの割合

SLI は障害モードを網羅している？



演習

SLO の目標値を決める

ジャーニーの信頼性の目標には何を設定しよう？

目標には**目標値**と**測定期間**の両方がないといけない

サービス	SLO の種類	目標
Web: プロファイル	可用性	**過去 28 日**で **99.95% 成功**
Web: プロファイル	レイテンシー	**過去 28 日**でリクエストの **90% < 500ms**
...	...	

まだ眠くない？

休憩！

ワークショップ: SLI と SLO を設定してみよう

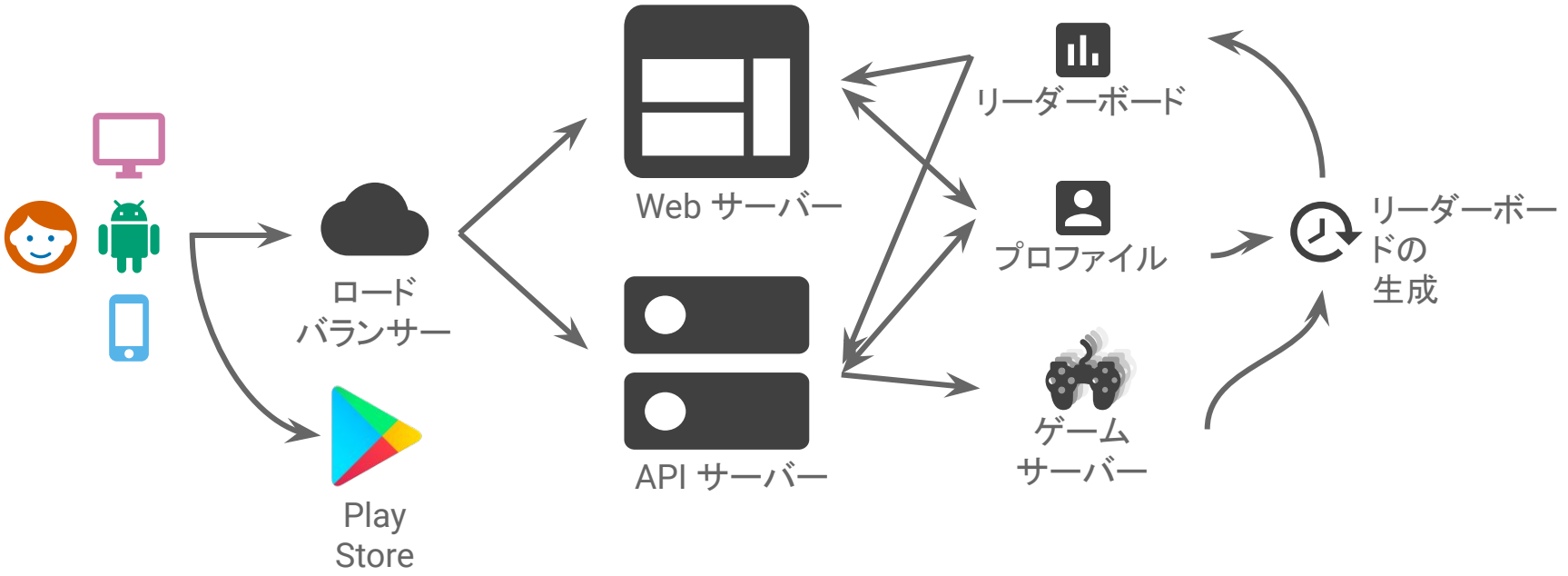
これまでの**手順**に沿って *ゲーム内通貨の購入* ジャーニーに取り組もう:

1. メニューから **SLI の仕様**を選ぶ (冊子の 6 ページを参照)
2. **定義** を置き換えて詳細な **SLI の実装**を作る
3. ユーザージャーニーを順に追って**網羅できている**か確認する
4. **ビジネス要件**から**野心的な SLO** を設定する

もし終わったら、**別のジャーニー**にも取り組んでみる

45 分ぐらい でやってみよう

みんなのゲーム: Fang Faction



休憩！

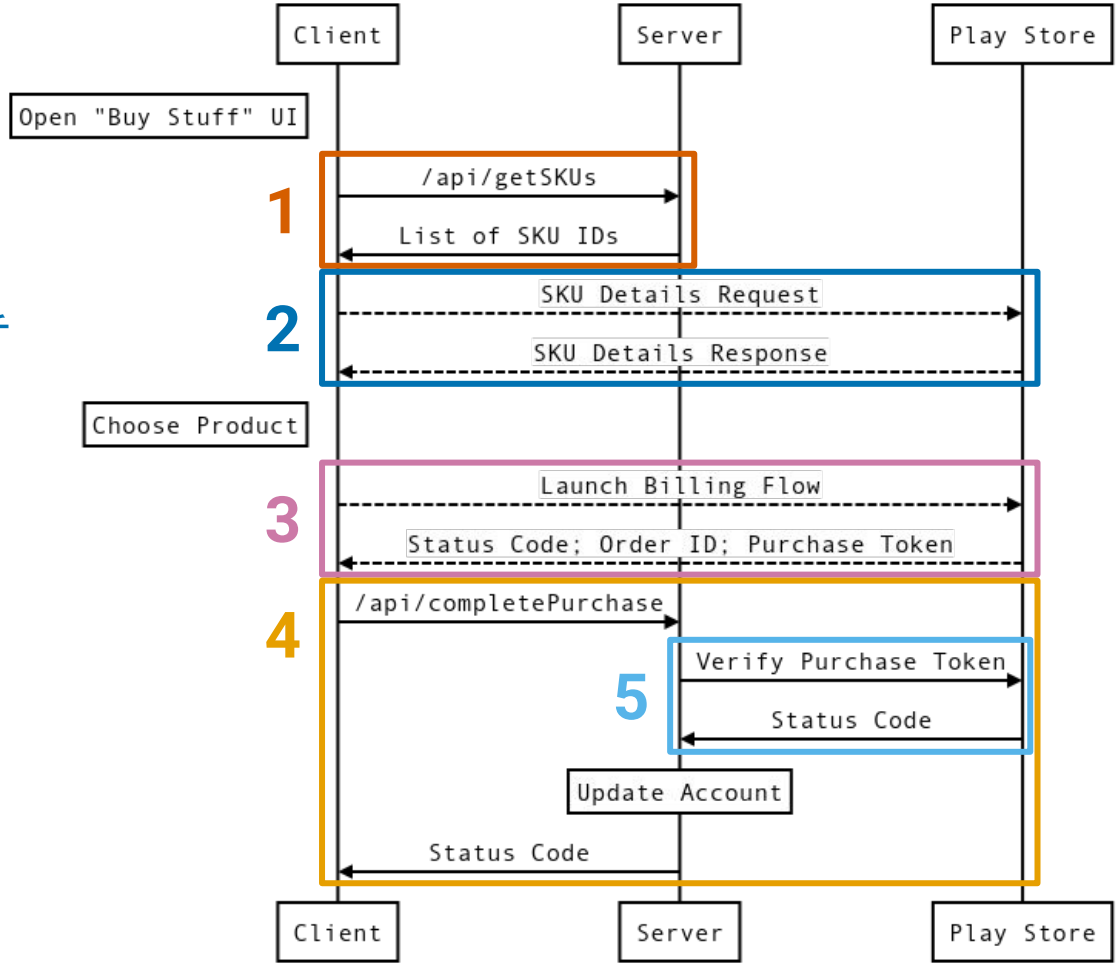
ゲーム内通貨の購入

モデル解答

ジャーニーの分解

5 つのリクエストとレスポンスのペア

1. APIサーバからSKUのリストをフェッチする
2. Play ストアからSKUの詳細を フェッチする
3. ユーザーはPlayの課金フローを 起動する
4. APIサーバにトークンを送信する
5. Playストアはトークンを確認する

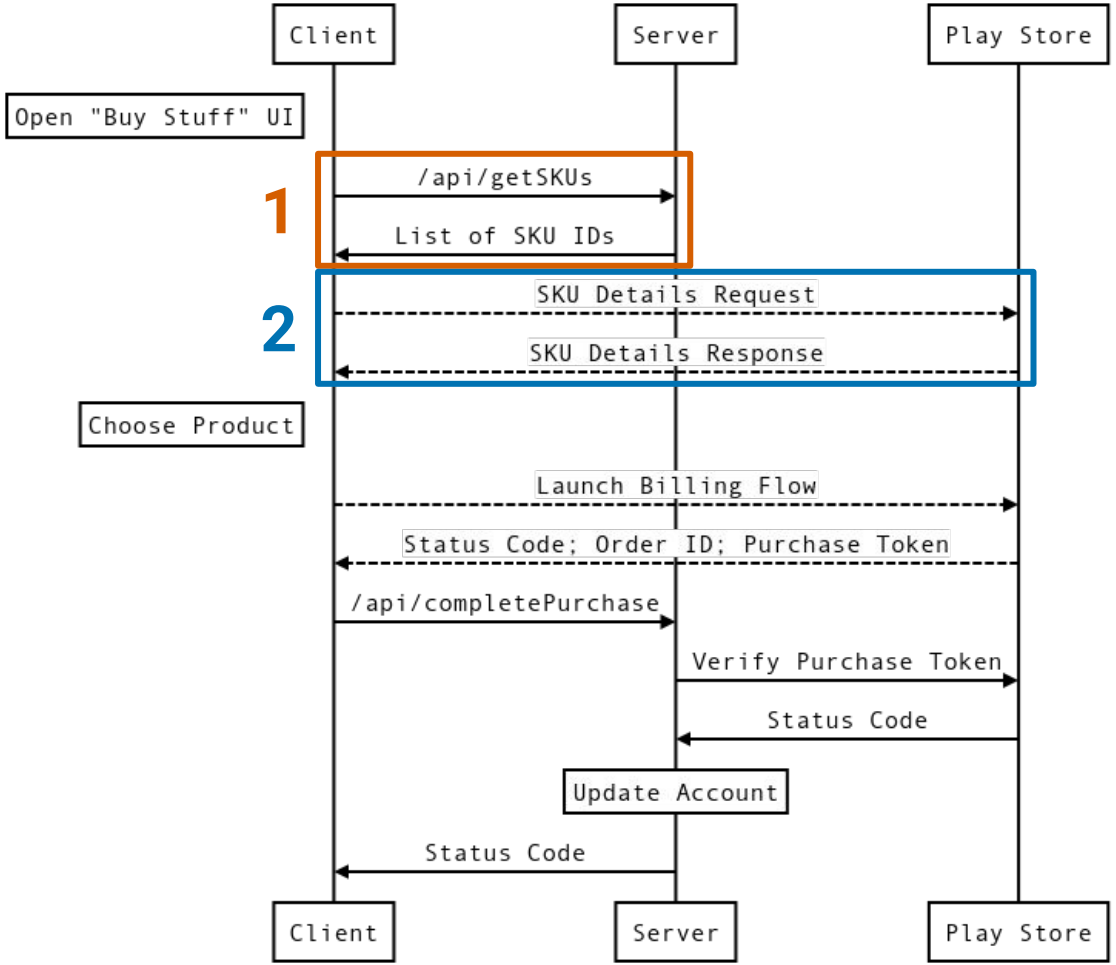


Google

ジャーニーの分解

ジャーニーには **2 つの**部分がある
A: SKU のフェッチ

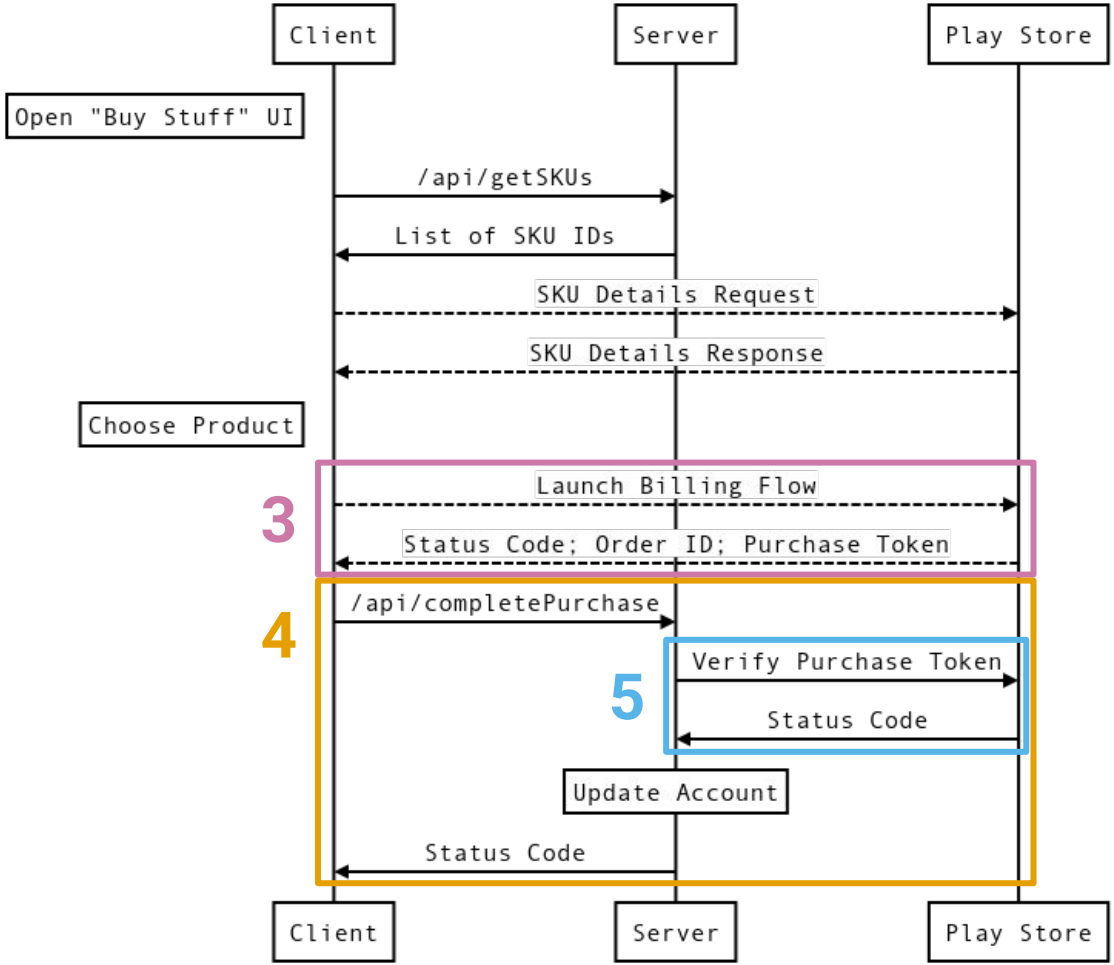
1. APIサーバからSKUのリストをフェッチする
2. PlayストアからSKUの詳細を フェッチする
3. ユーザーはPlayの請求フローを 起動する
4. API サーバにトークンを送信する
5. Playストアはトークンを確認する



ジャーニーの分解

ジャーニーには **2 つの**部分がある
B: アイテムの購入

1. API サーバからSKUのリストをフェッチする
2. PlayストアからSKUの詳細を フェッチする
3. ユーザーはPlayの課金フローを 起動する
4. API サーバにトークンを送信する
5. Playストアはトークンを確認する



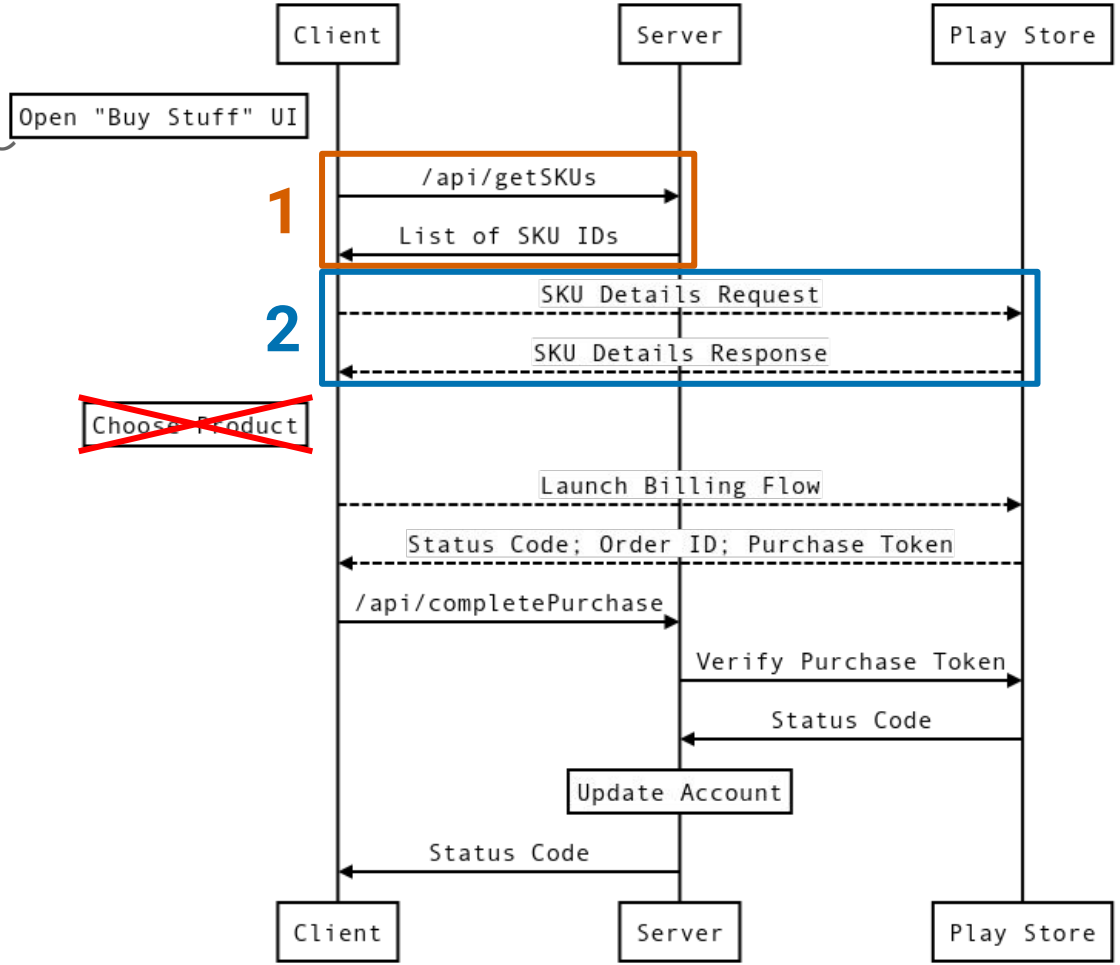
Google

ジャーニーの分解

ユーザーはアイテムを購入 **しない**　　かもしれない :-(

1. APIサーバからSKUのリストをフェッチする
2. PlayストアからSKUの詳細を　フェッチする
3. ユーザーはPlayの課金フローを　起動する
4. API サーバにトークンを送信する
5. Playストアはトークンを確認する

この部分は **別々** に考えないといけない！



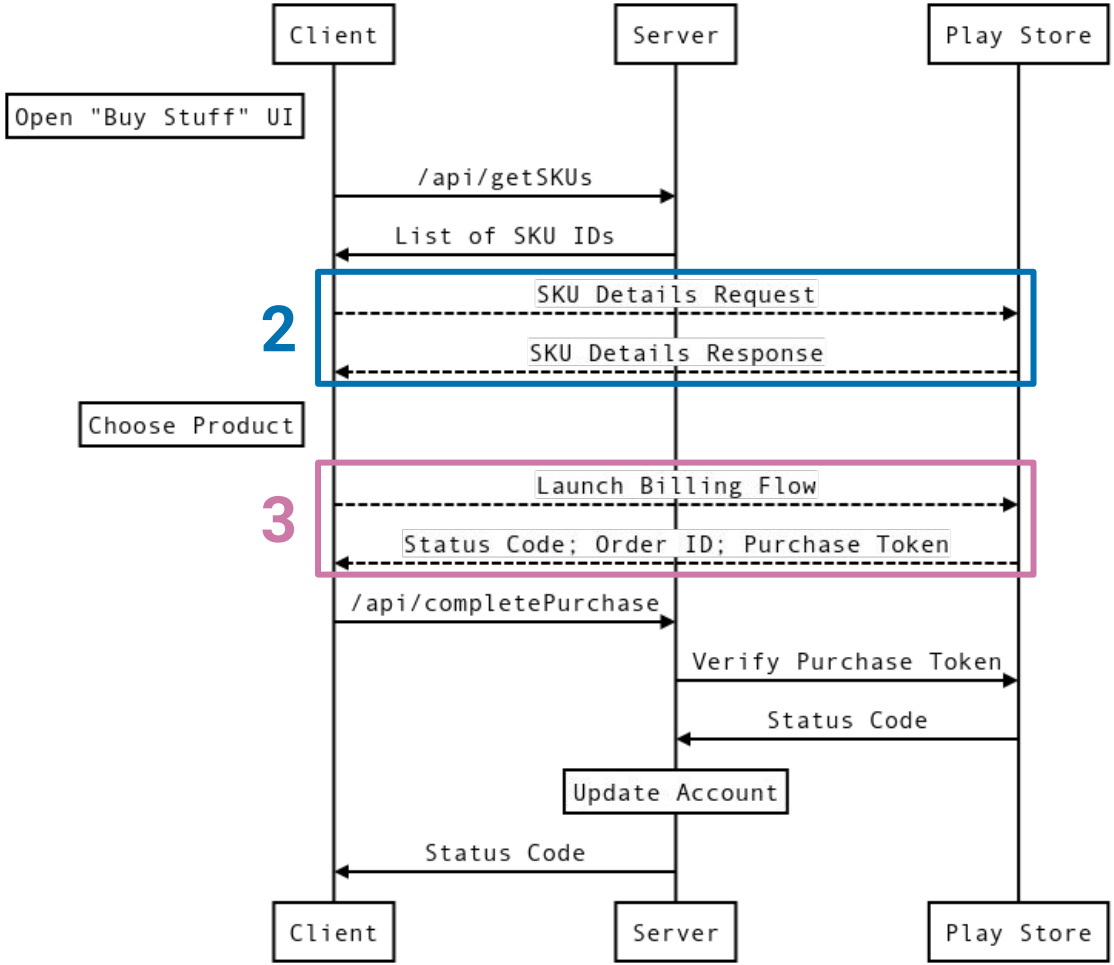
Google

ジャーニーの分解

2 つのリクエストはAPI サーバに　　全くヒットしない！

1. API サーバからSKUのリストをフェッチする
2. PlayストアからSKUの詳細を　フェッチする
3. ユーザーはPlayの課金フローを　　起動する
4. API サーバにトークンを送信する
5. Playストアはトークンを確認する

サーバーやロードバランサーのメトリックだとジャーニーを**網羅できないかも** :-(



Google

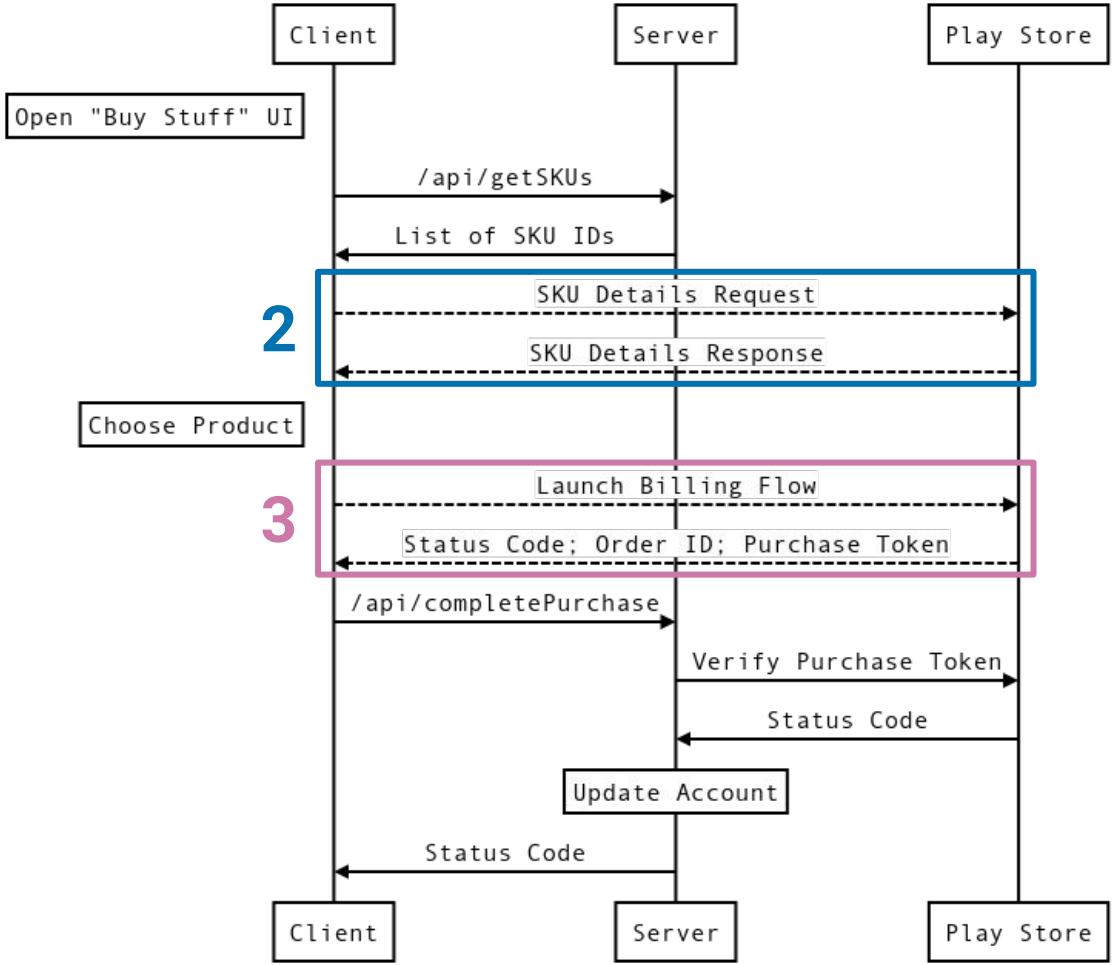
ジャーニーの分解

2 つのリクエストはAPI サーバに　　全くヒットしない！

1. API サーバからSKUのリストをフェッチする
2. PlayストアからSKUの詳細を　フェッチする
3. ユーザーはPlayの課金フローを　　起動する
4. API サーバにトークンを送信する
5. Playストアはトークンを確認する

サーバーやロードバランサーのメトリックだとジャーニーを**網羅できないかも** :-(

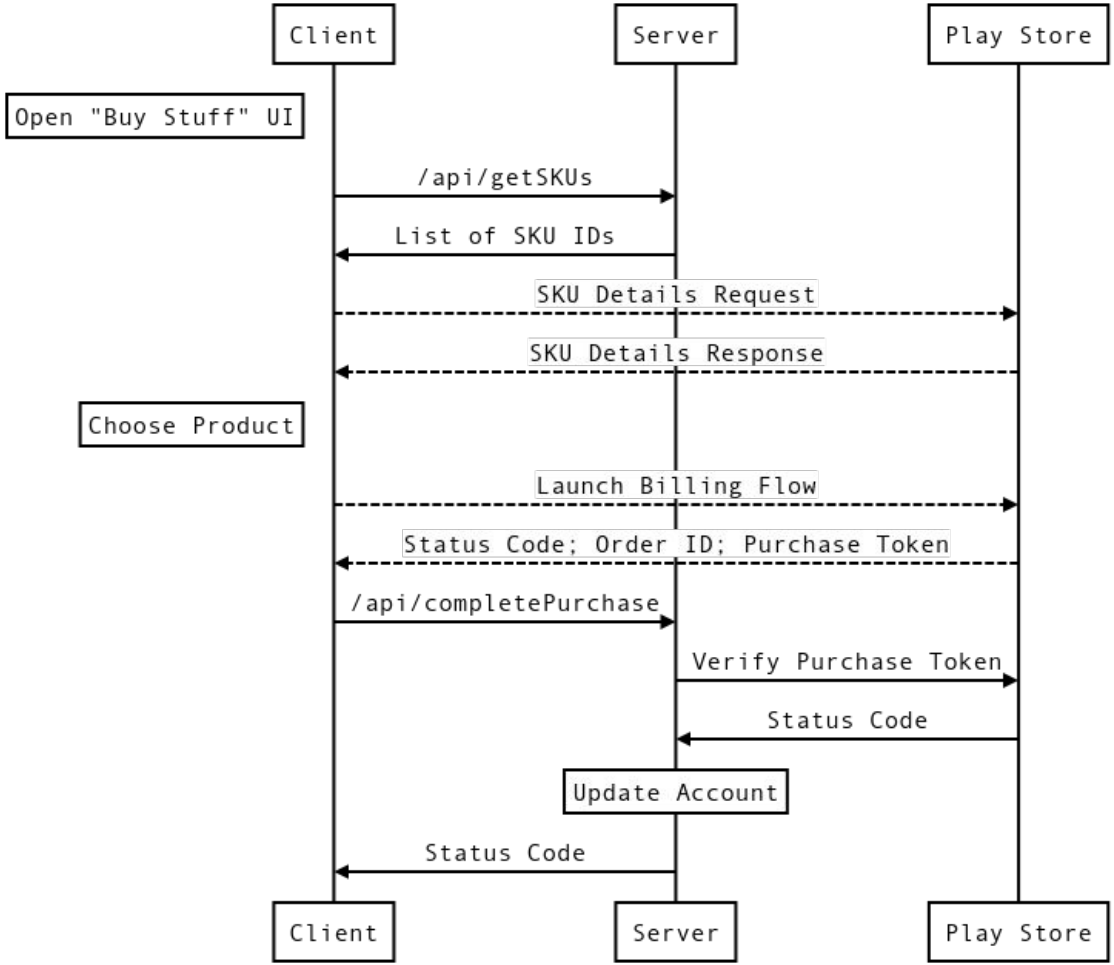
...クライアントから情報収集するならユーザーに**同意**を得ないといけない



購入フロー
どのSLIを使う？

購入フローのジャーニーは
リクエスト/レスポンス

SLI メニューをみると
可用性と**レイテンシの** SLI
を使えばよさそうだ



Google

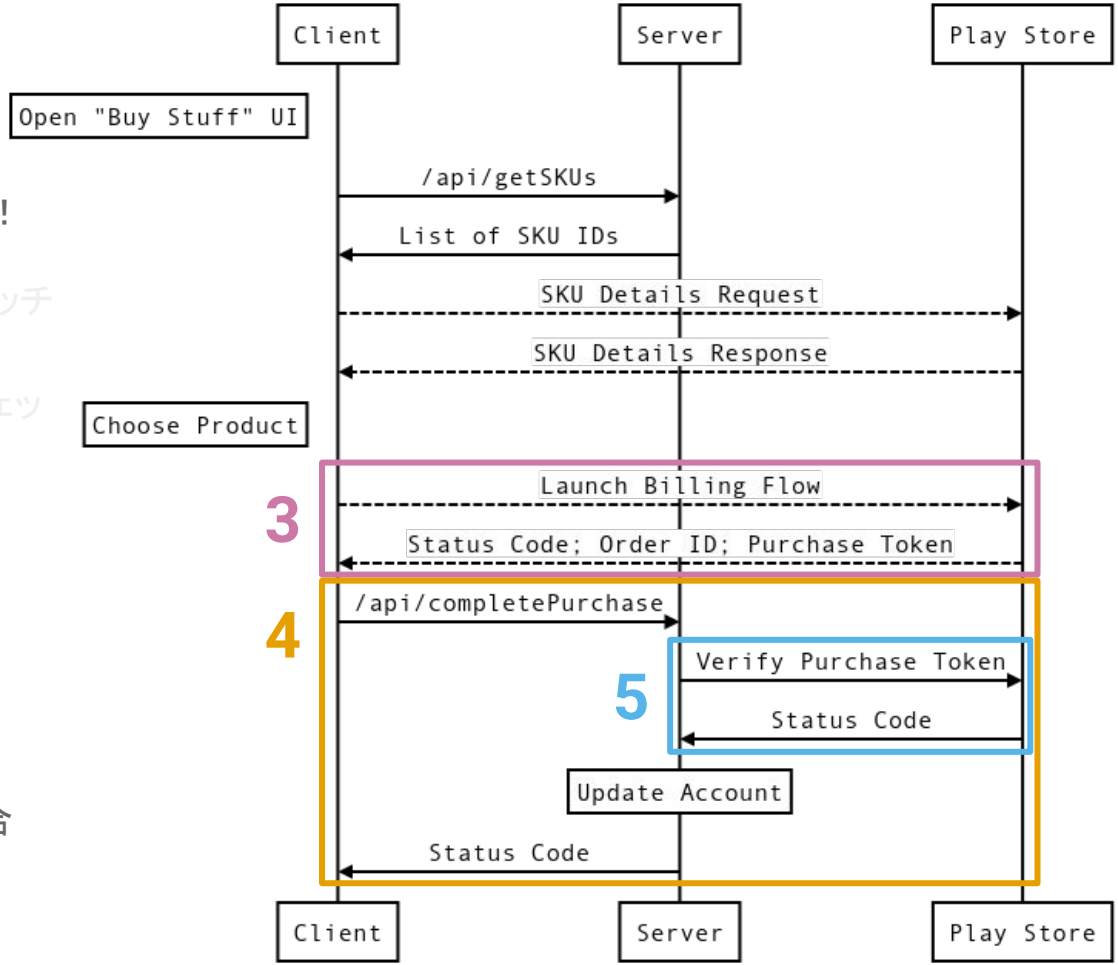
購入フローの可用性仕様

Bは売上につながるので、そこから始めよう！

1. API サーバからSKUのリストを　フェッチする
2. PlayストアからSKUの詳細を　フェッチする
3. ユーザーはPlayの課金フローを起動する
4. API サーバにトークンを送信する
5. Playストアはトークンを確認する

<u>可用性SLI の仕様</u>

正常に処理された***有効***なリクエストの割合



Google

購入フローの可用性
有効なリクエスト

可用性SLI

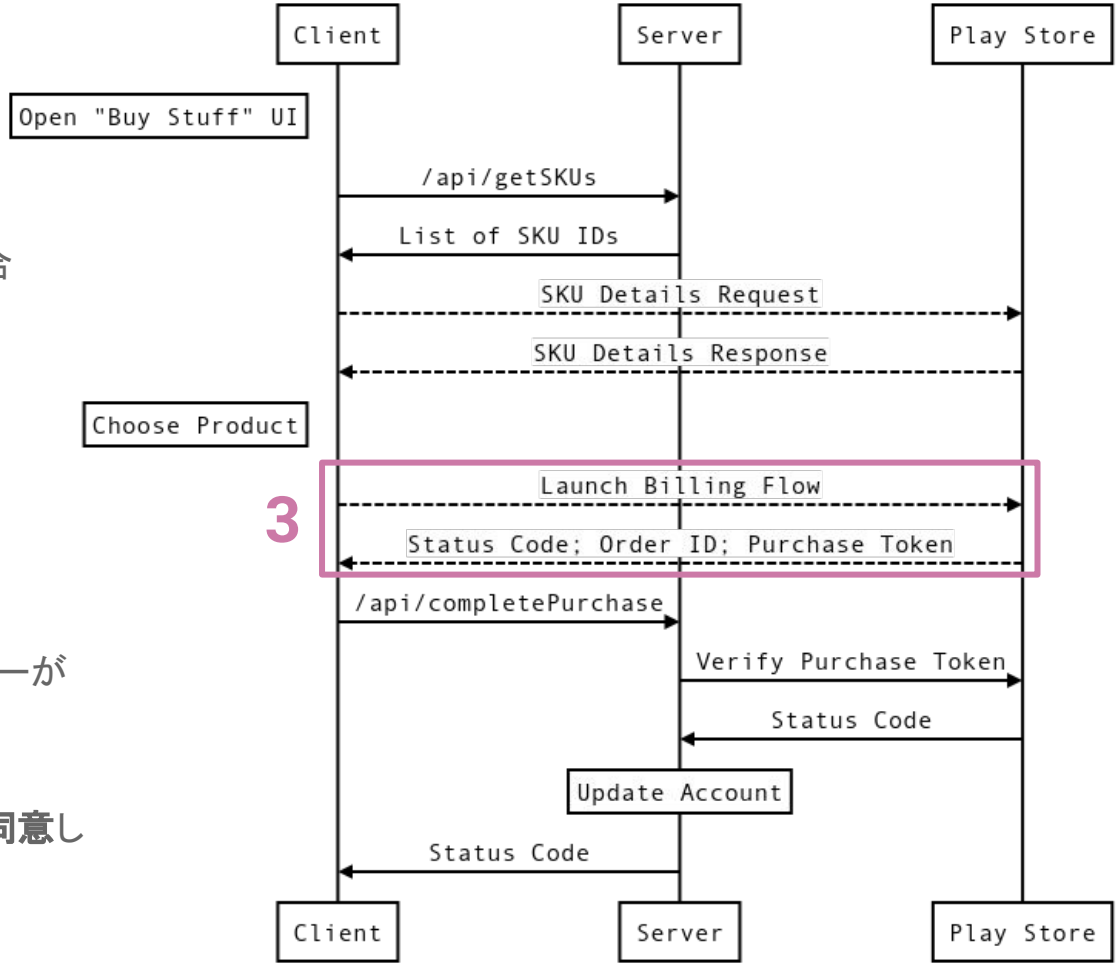
正常に処理された***有効***なリクエストの割合

… どのリクエストが***有効***なんだろう？

3. ユーザーはPlayの請求フローを起動する
4. API サーバにトークンを送信する？
5. Playストアはトークンを確認する？

課金フローが開始されたってことは、ユーザーが購入する***意志***を示した、ということ

ユーザーがクライアントからの情報収集に **同意**してくれれば、その意志を **捕捉**できる



購入フローの可用性正常の基準

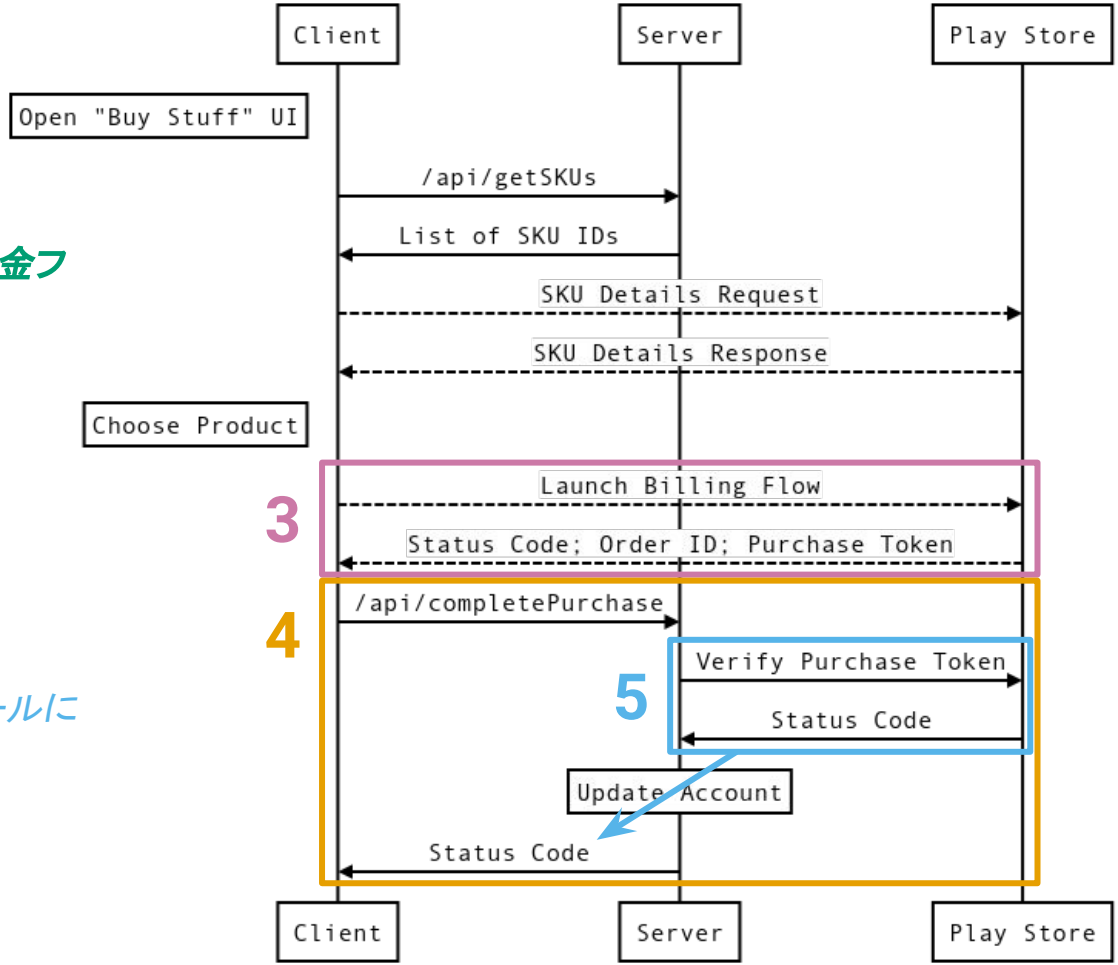
可用性SLI

情報収集に同意したユーザーが開始した課金フロー のうち ***成功した*** ものの割合

… ***成功*** したことはどうやってわかる？

一連のやりとりが成功するってことだ！

3. 良いステータスコード:購入トークン
4. 良いステータスコード:口座情報更新
5. 良いステータスコード:有効なトークン
   - *トークンが無効な場合、APIコールに対して **402** を返答する*



Google

購入フローの可用性測定

可用性SLI

情報収集に同意したユーザーが開始した課金フロー のうち ***成功した*** ものの割合

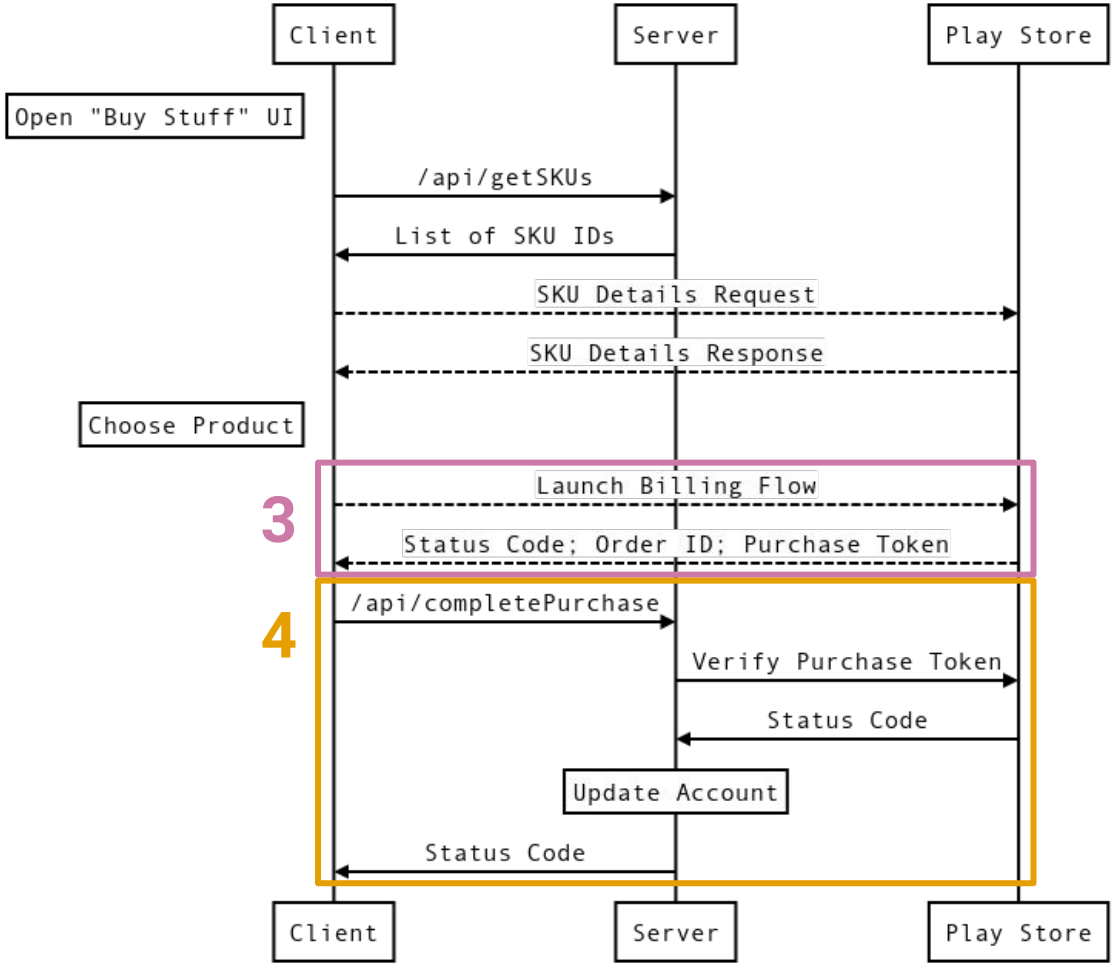
課金フローは以下を返す

- OKおよび購入トークン
- または FEATURE_NOT_SUPPORTED
- または ITEM_UNAVAILABLE
- または USER_CANCELED

/api/completePurchase は以下を返す

- 200 OKおよびパース可能なJSON
- または 402 Payment Required

… でも、どこでそれを ***測定*** しよう？



Google

購入フローの可用性測定

可用性SLI

情報収集に同意したユーザーが開始した課金フロー のうち ***成功した*** ものの割合

課金フローは以下を返す

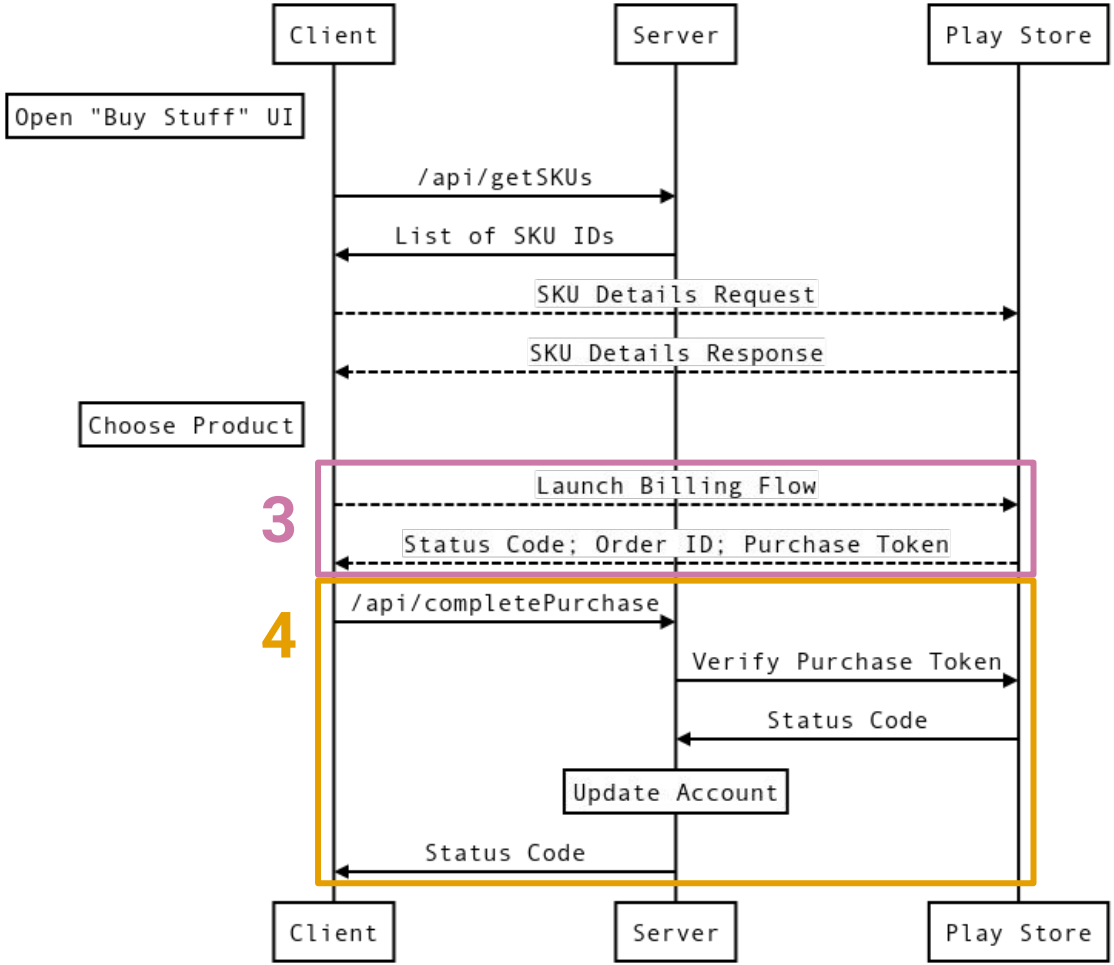
- OKおよび購入トークン
- または FEATURE_NOT_SUPPORTED
- または ITEM_UNAVAILABLE
- または USER_CANCELED

/api/completePurchase は以下を返す

- 200 OKおよびパース可能なJSON
- または 402 Payment Required

… でも、どこでそれを ***測定*** しよう？

ゲームクライアント で測定して、非同期で報告させよう！



Google

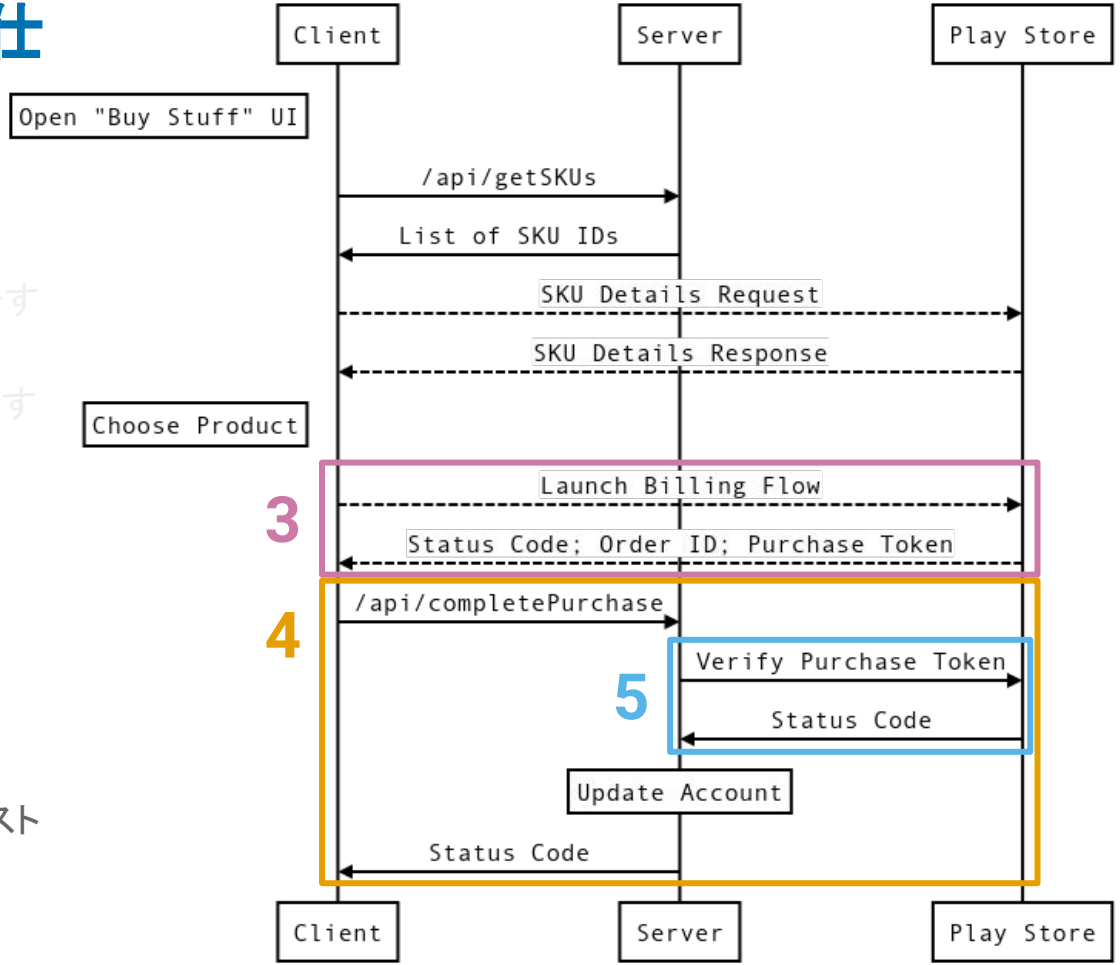
購入フローのレイテンシ仕様

Bのレイテンシも測定したいよね

1. APIサーバからSKUのリストをフェッチする
2. PlayストアからSKUの詳細を フェッチする
3. ユーザーはPlayの課金フローを起動する
4. APIサーバにトークンを送信する
5. Playストアはトークンを確認する

<u>レイテンシSLIの仕様</u>

しきい値よりも***速く***処理された***有効***なリクエストの割合



Google

購入フローのレイテンシ
有効なリクエスト

レイテンシSLI

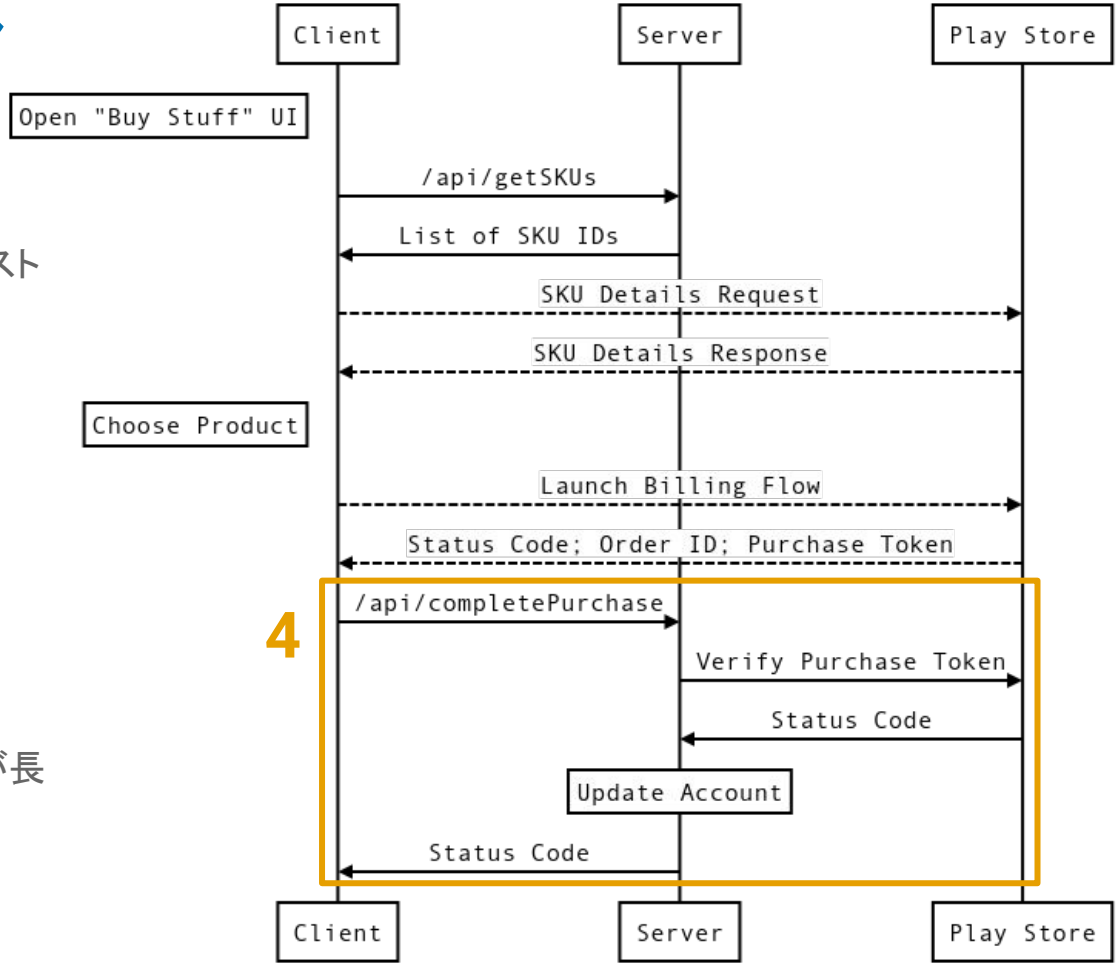
しきい値よりも***速く***処理された***有効***なリクエストの割合

… どのリクエストが***有効***なんだろう？

3. ユーザーはPlayの課金フローを起動する
4. API サーバにトークンを送信する
5. Playストアはトークンを確認する

3 じゃないの？

- 変動が大きすぎて、S/N 比が悪い
- 課金フローは、端末を操作する時間が長い



Google

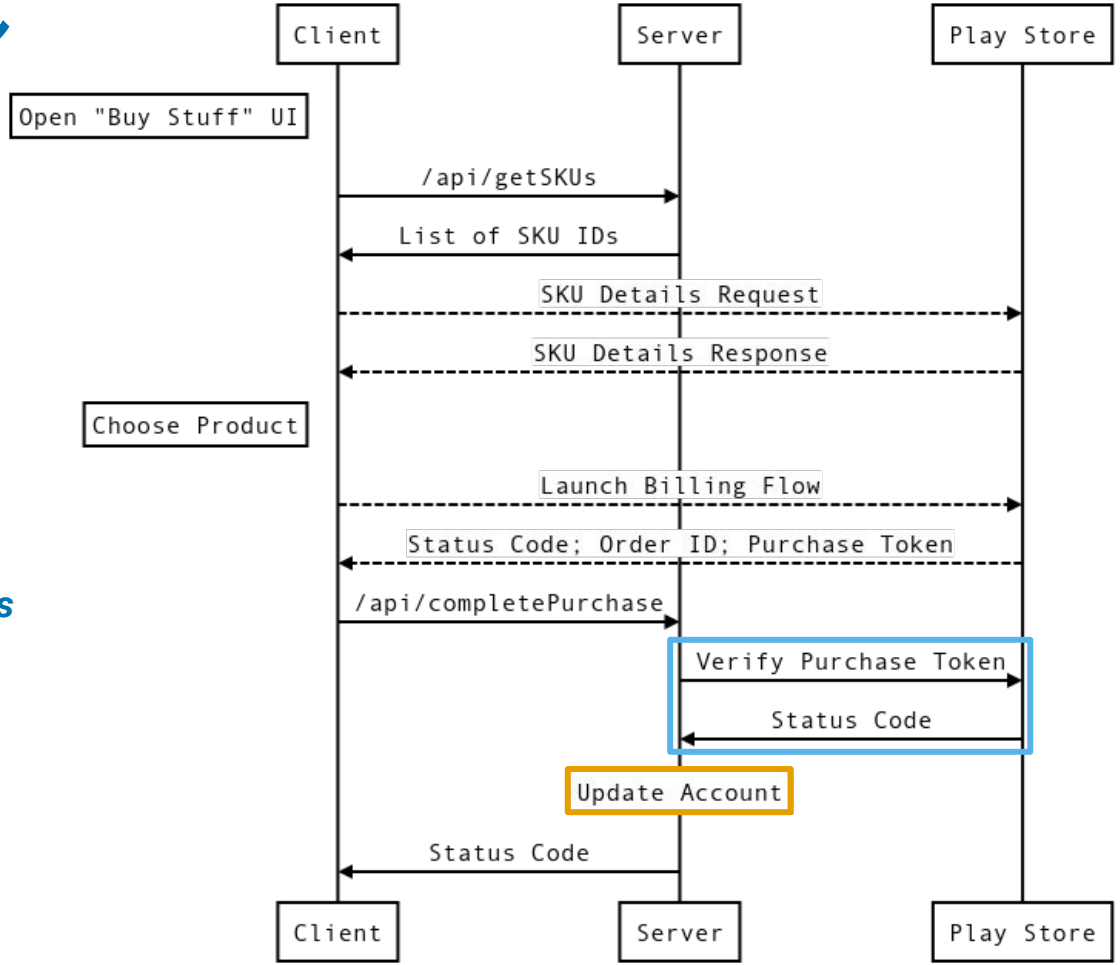
購入フローのレイテンシ「遅すぎる」しきい値

レイテンシSLI
しきい値よりも***速く***処理された
/api/completePurchaseリクエストの割合

… で、***十分に速い***って何だろう？

ざっくり見積もると

- トークンの確認 ***<= 500ms***
- データベースへの書き込み ***<= 200ms***
- ちょっと切り上げると…



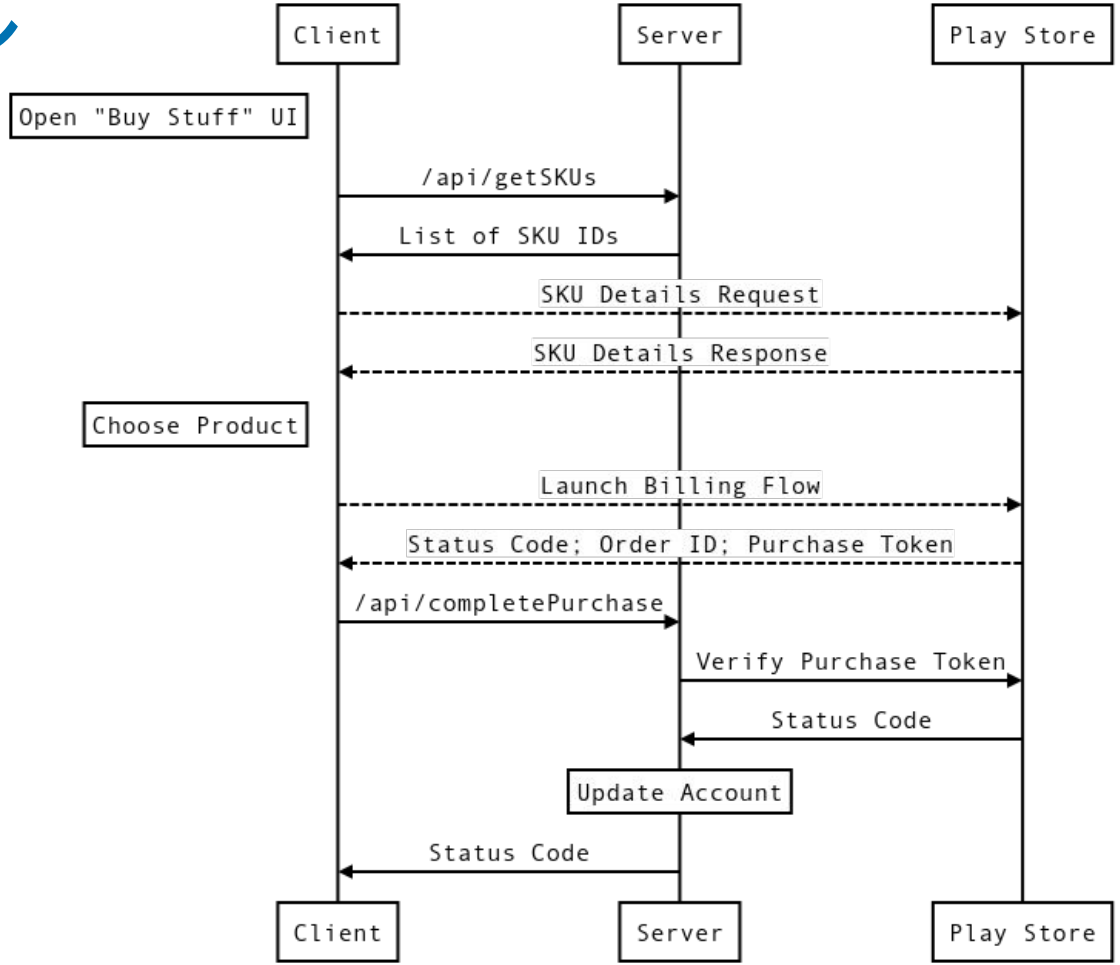
Google

購入フローのレイテンシ測定

レイテンシSLI

1000ms以内に 処理される
/api/completePurchaseリクエストの割合

… で、どこでそれを ***測定***しよう？
いつからいつまでを測る？

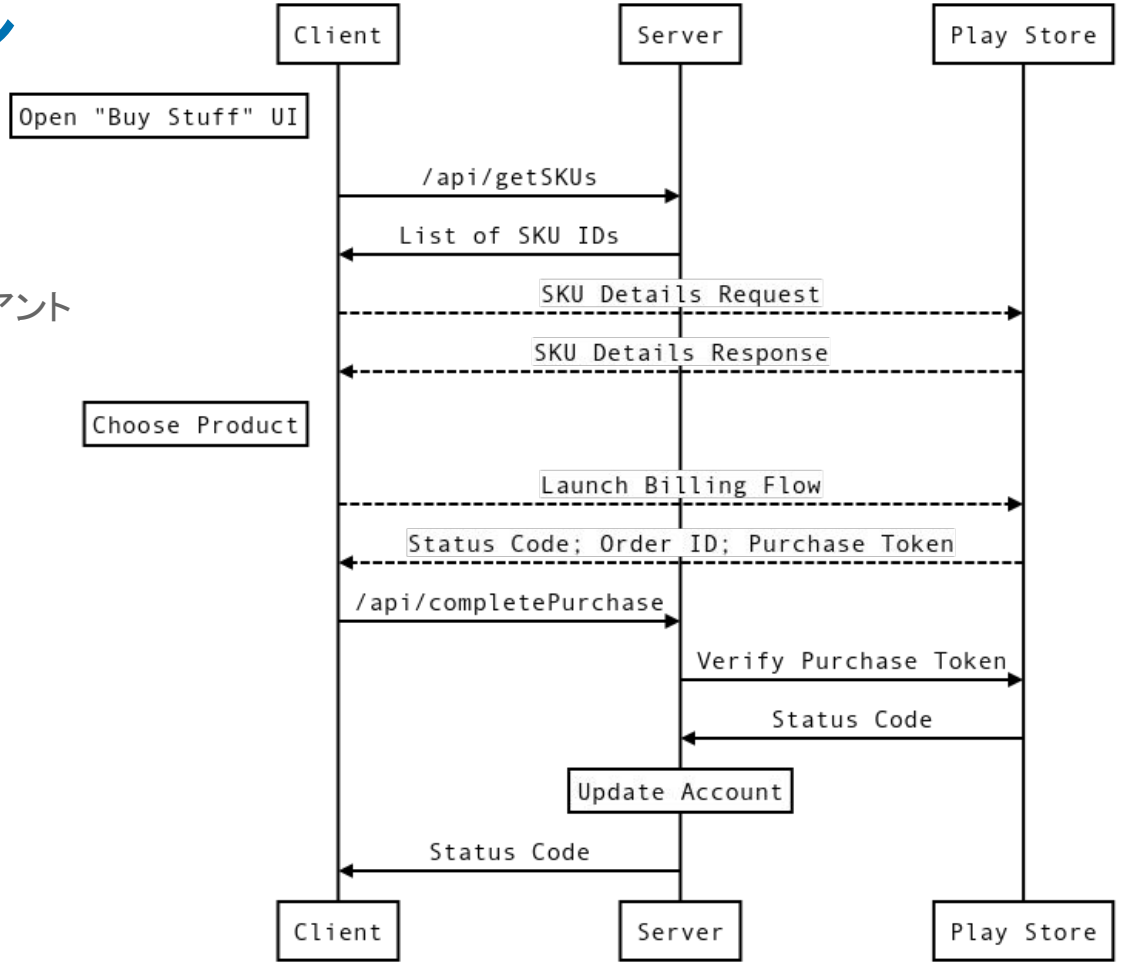


Google

購入フローのレイテンシ測定

レイテンシSLI

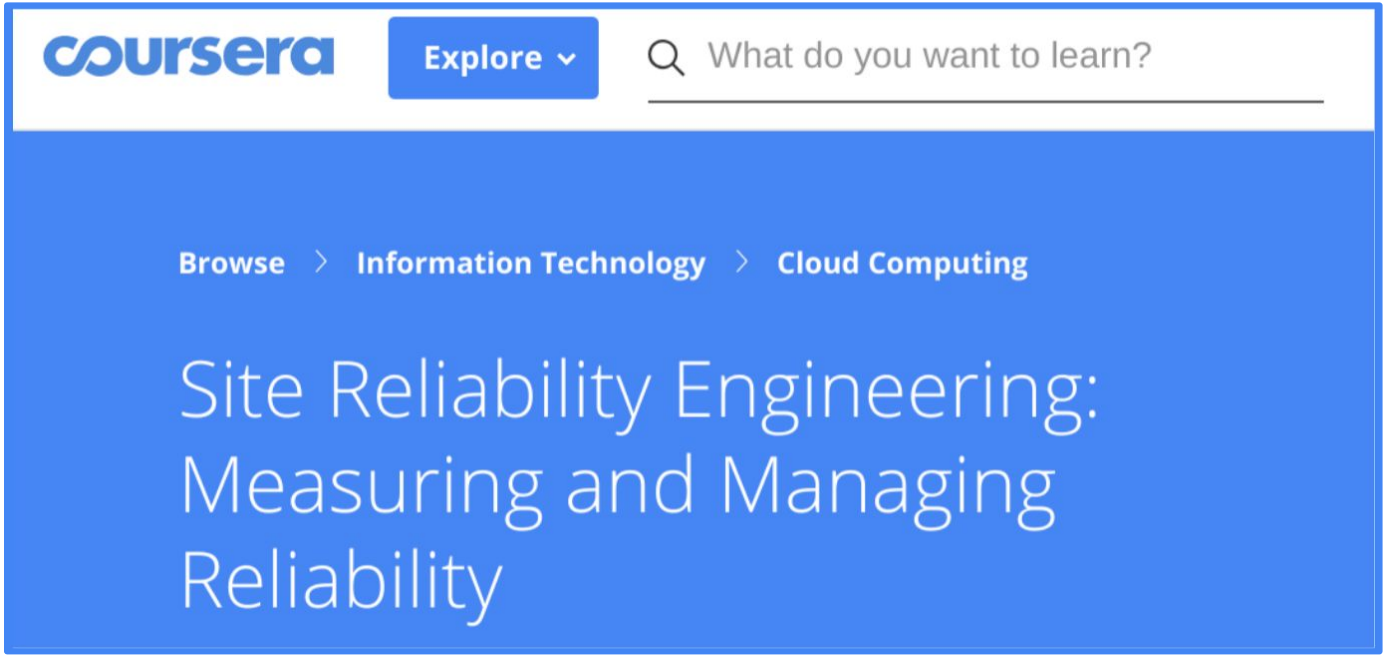
ロードバランサーで測定された
1000ms以内に完全なレスポンス がクライアントに返されている
/api/completePurchasリクエストの割合



Google

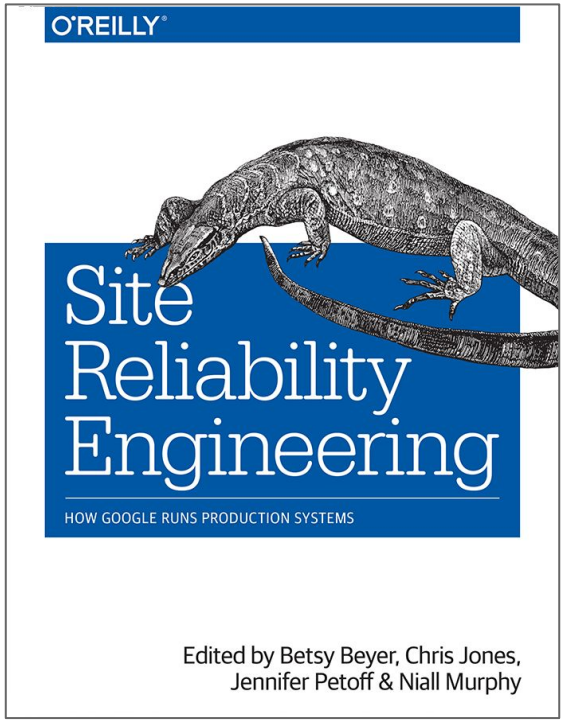
ちょっとだけ宣伝させて...

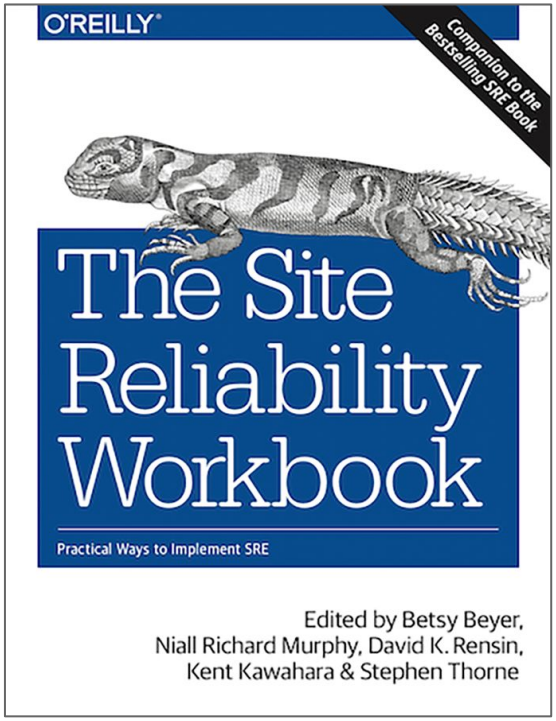
https://cre.page.link/art-of-slos



SLO をもっと勉強したい？ Coursera のコースを受けてみて:

https://cre.page.link/coursera





この本は HTML なら無料で読めるよ！

https://landing.google.com/sre/books/

Insert QR code link
to feedback form
in this space!

ありがとう！

Insert QR code link
to feedback form
in this space!

アンケートにも答えてね！

Q&A

パネリストに**質問**をどうぞ！