Prazto

Herokuの“クセ”を突破して最適なシステムを構築するには

Herokuの“クセ”を突破して

最適なシステムを構築するには

Prazto

目次

CHAPTER01

1. HerokuのWeb Dynoの30秒制限

【システム連携ポートフォリオ】Herokuの”クセ”を突破して、最適なシステムを構築するには

Herokuは、その簡便さと迅速な開発環境の提供により、多くのシステム構築で採用されています。インフラ設定を極力簡略化したプラットフォームであるため、インフラエンジニアを必要とせず、アプリケーション開発者が開発に専念できる環境を提供しています。

しかし、この簡略化のために一定の制約やカスタマイズの難しさが存在することも認識する必要があります。本記事では、**Herokuの特徴的な制約、いわゆる”クセ”とその解決策について詳しく解説します。**また、これらの特性を正しく理解し、システム要件に適合する場合にHerokuの採用を検討することの重要性についても触れていきます。

具体的には以下の3点について詳しく解説していきます：

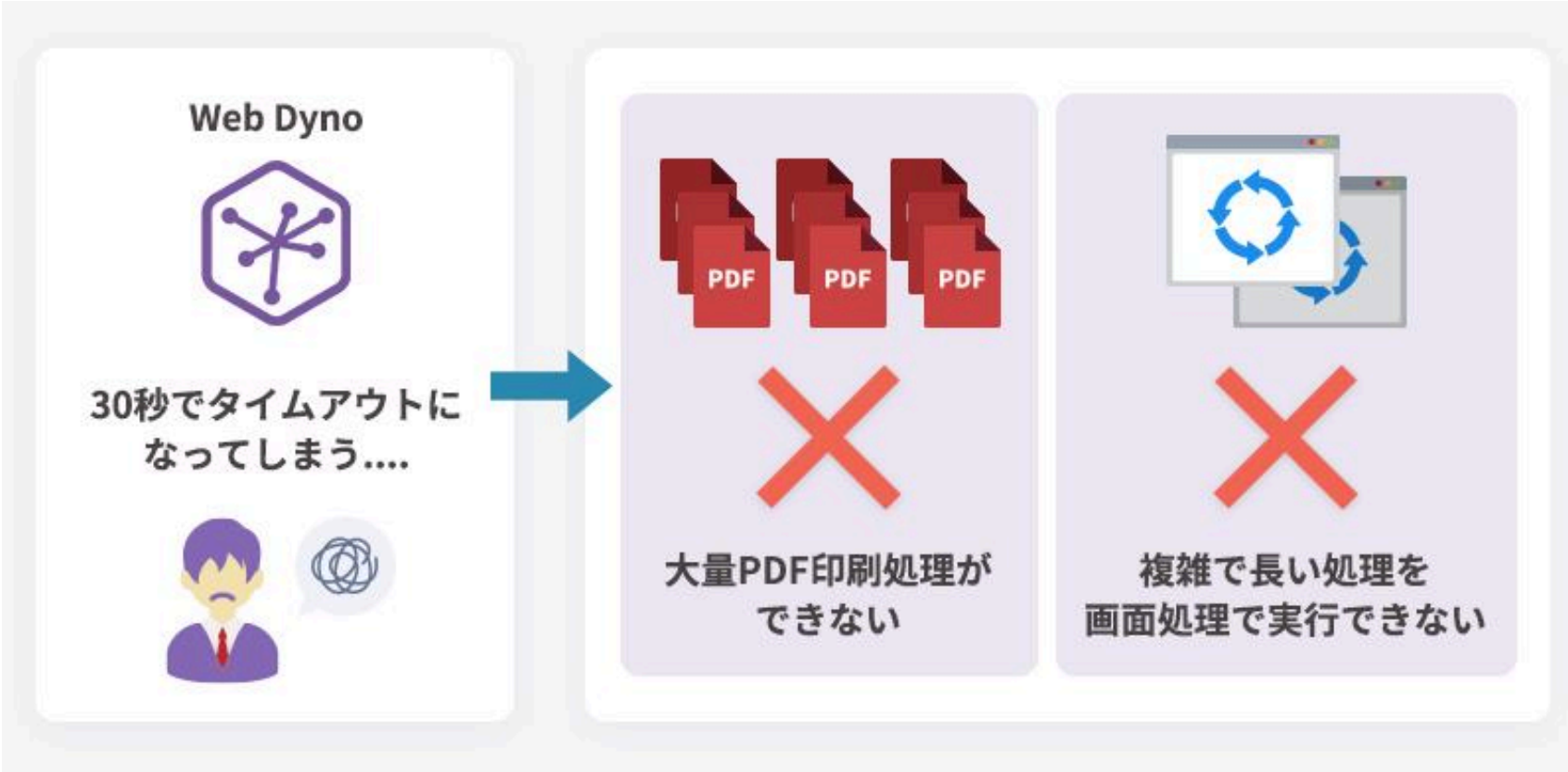
1. HerokuのWeb Dynoの30秒制限
2. IPアドレスが動的に変化する問題
3. 静的ストレージが無い問題

それでは、各項目について詳しく見ていきましょう。

1. HerokuのWeb Dynoの30秒制限

問題の概要

HerokuのWeb Dynoには、ユーザーからのリクエストを受け付けてから30秒でタイムアウトし、処理が強制終了されるという制限があります。この30秒という時間は、多くの処理では十分ですが、データ集計や大量のレコード処理など、時間のかかる処理では簡単に超えてしまう可能性があります。Herokuを採用するかどうかを検討する際、このような長時間処理がシステムの主要な機能になるかどうかは重要な判断材料となります。



回避策

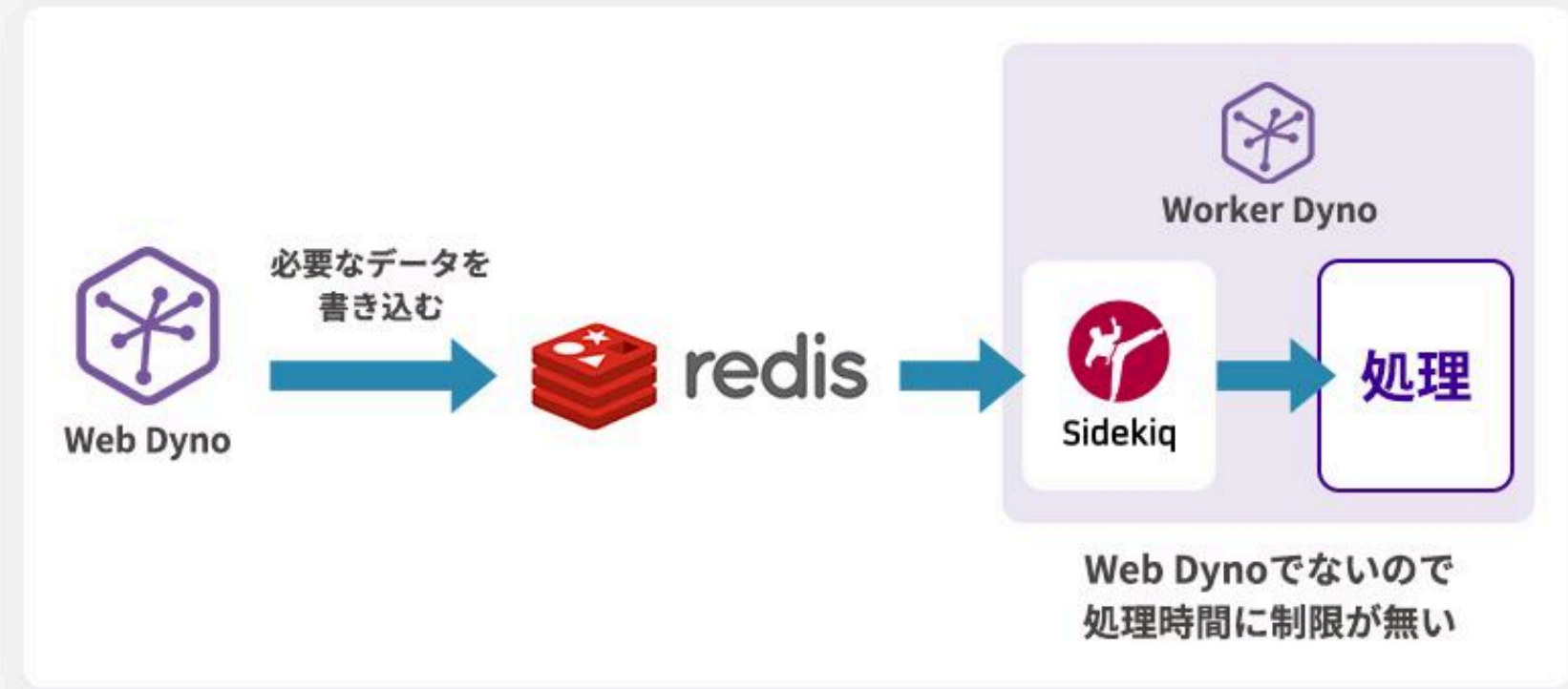
Web Dyno以外のDynoタイプを使用すれば、30秒の制限を回避することができます。以下に、具体的な回避策をいくつか紹介します。

【解決・回避策①】Web Dynoから非同期処理を実行する

Web Dyno以外のDynoを使用することで、30秒の制限なく処理を実行できます。この方法では、Heroku Redisを使用してWeb Dynoと非同期用のDyno(以下ではWorker Dynoとします)間でメッセージをやり取りします。

Heroku Redisは、HerokuのAdd-onとして提供されるインメモリ型の Key-Value データストアです。これを使用することで、Web DynoからWorker Dynoへ処理を委譲し、長時間の処理を実行することができます。
Railsを使用して実装する場合には、Redis からJobを取り出しバックグラウンドで実行を行うための便利なライブラリ(Gem)もあり、弊社はSidekiqというものをよく使用しておりお勧めです。Sidekiqを使用することで、以下のような流れで処理を実行できます：

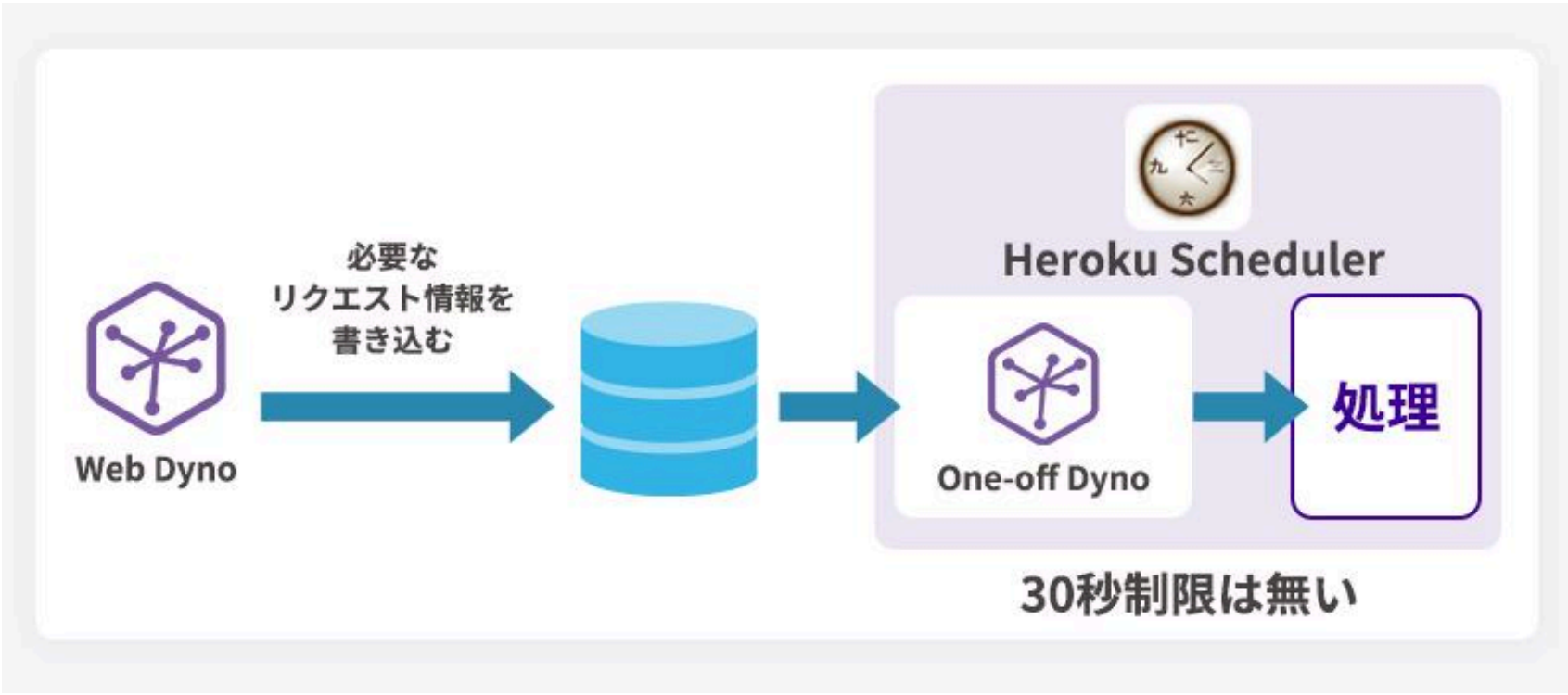
1. Web Dynoで受け付けたリクエストをRedisのキューに登録
2. Worker Dynoが定期的にキューをチェックし、ジョブを取得
3. Worker Dynoでジョブを実行（30秒制限なし）



【解決・回避策②】Web Dynoでリクエストを受け付け、定期バッチ処理で実行（Heroku Scheduler使用）

この方法では、Web Dynoでリクエストを受け付けた際に、ジョブのリクエスト情報をテーブルにレコードとして登録します。その後、Heroku Schedulerから実行される定期的ジョブがそのレコードの有無を確認し、処理を実行するかどうかを決定します。

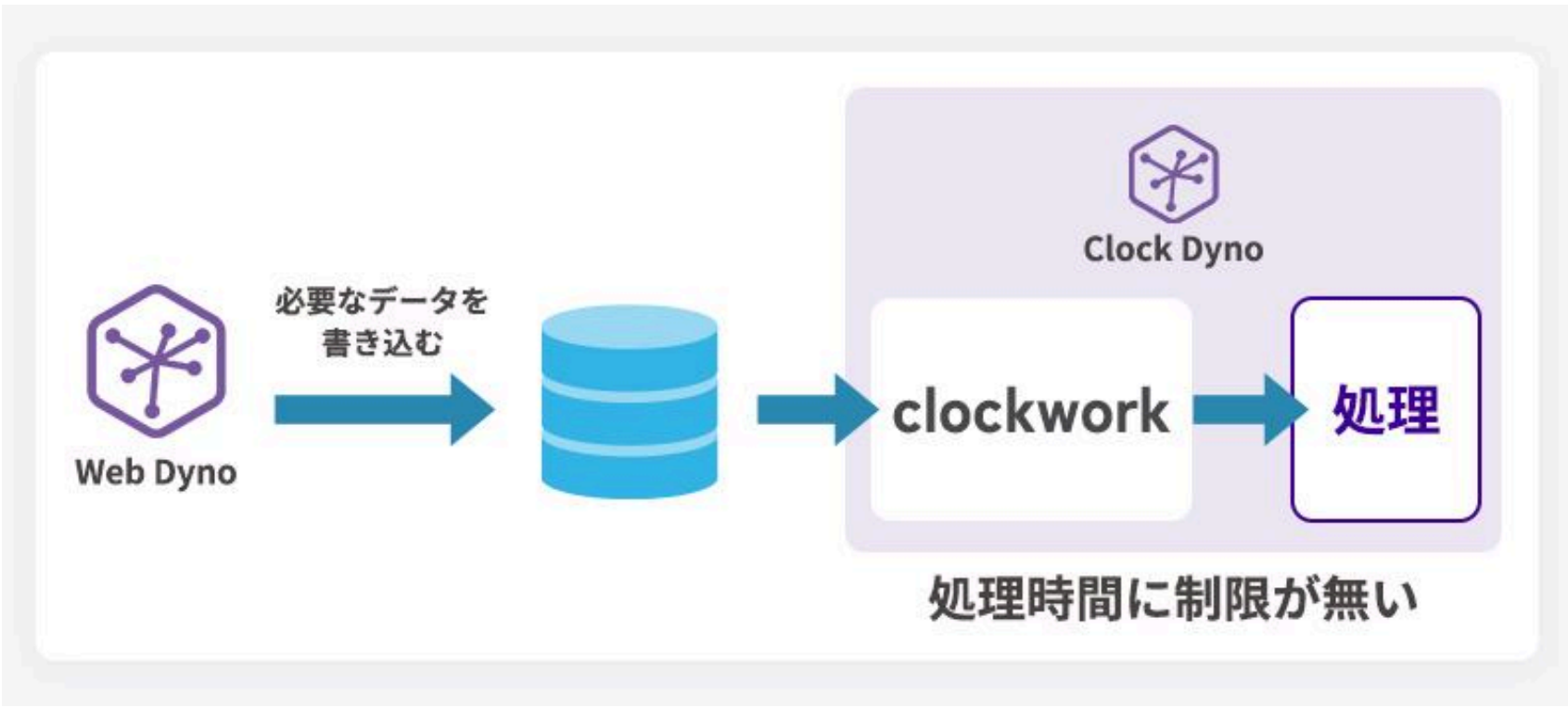
Heroku Schedulerは、指定した時間間隔で自動的に処理を実行するAddonです。この際に動作するDynoは、One-off Dynoと呼ばれます。One-off Dynoは、特定のタスクを実行するために一時的に起動されるDynoで、タスク完了後に自動的に停止します。



【解決・回避策③】Web Dynoでリクエストを受け付け、定期バッチ処理で実行（Clock Dyno使用）

Heroku Schedulerを使わずとも、Web Dyno以外の追加Dyno（以下、Clock Dyno）とRuby Gemを活用することで、定期バッチ処理を構成し、30秒制限の問題を回避できます。
Ruby Gemの中でも、clockworkは定期処理に適しており、Rubyで実行可能なcron相当の機能を提供します。

処理の流れとしては、まずWeb Dynoでリクエストを受け取った際、ジョブの情報をデータベースのテーブルに記録します。次に、常時稼働しているClock Dynoからclockworkの処理が定期的に実行され、テーブル内のレコードを確認します。レコードが存在する場合、Clock Dynoで対応する処理を実行します。この方法では、Clock Dynoが Web Dynoとは異なり30秒制限がないため、長時間の処理実行が可能となります。



CHAPTER02

2. IPアドレスが動的に変化する問題

2. IPアドレスが動的に変化する問題

問題の概要

HerokuのDynoは、リブートするたびにIPアドレスが変更されるという特徴があります。これは、HerokuからAPIを使って他のシステムに接続する際に問題になる可能性があります。

多くのSaaSでは、セキュリティ対策としてログインIPアドレスの制限を設けています。例えば、Salesforceもこのような機能を提供しており、企業規模が大きくなるほど、IPアドレス制限を適用しているケースが増えています。

HerokuからこのようなSaaSにAPIを実行する際、Dynoがリブートする度にIPアドレスが変わってしまうと、適切なIPアドレス制限を設定できなくなります。結果として、お客様や接続先のSaaSのセキュリティ要件を満たせなくなる可能性があります。

解決・回避策

この問題に対しては、以下の3つの方法で対応できます。

【解決・回避策①】Herokuの契約プランをPrivate Spaceに変更する

最も確実な対応方法は、HerokuのPrivate Spaceプランを利用することです。Private Spaceプランでは、DynoのIPアドレスがリブートによらず固定化されるため、上記の問題をそのまま解決できます。ただし、Private Spaceプランは、IPアドレスの固定化以外にも様々な高度な機能が提供されているため、通常のプランよりもライセンス費用が高くなります。そのため、費用対効果を十分に検討する必要があります。

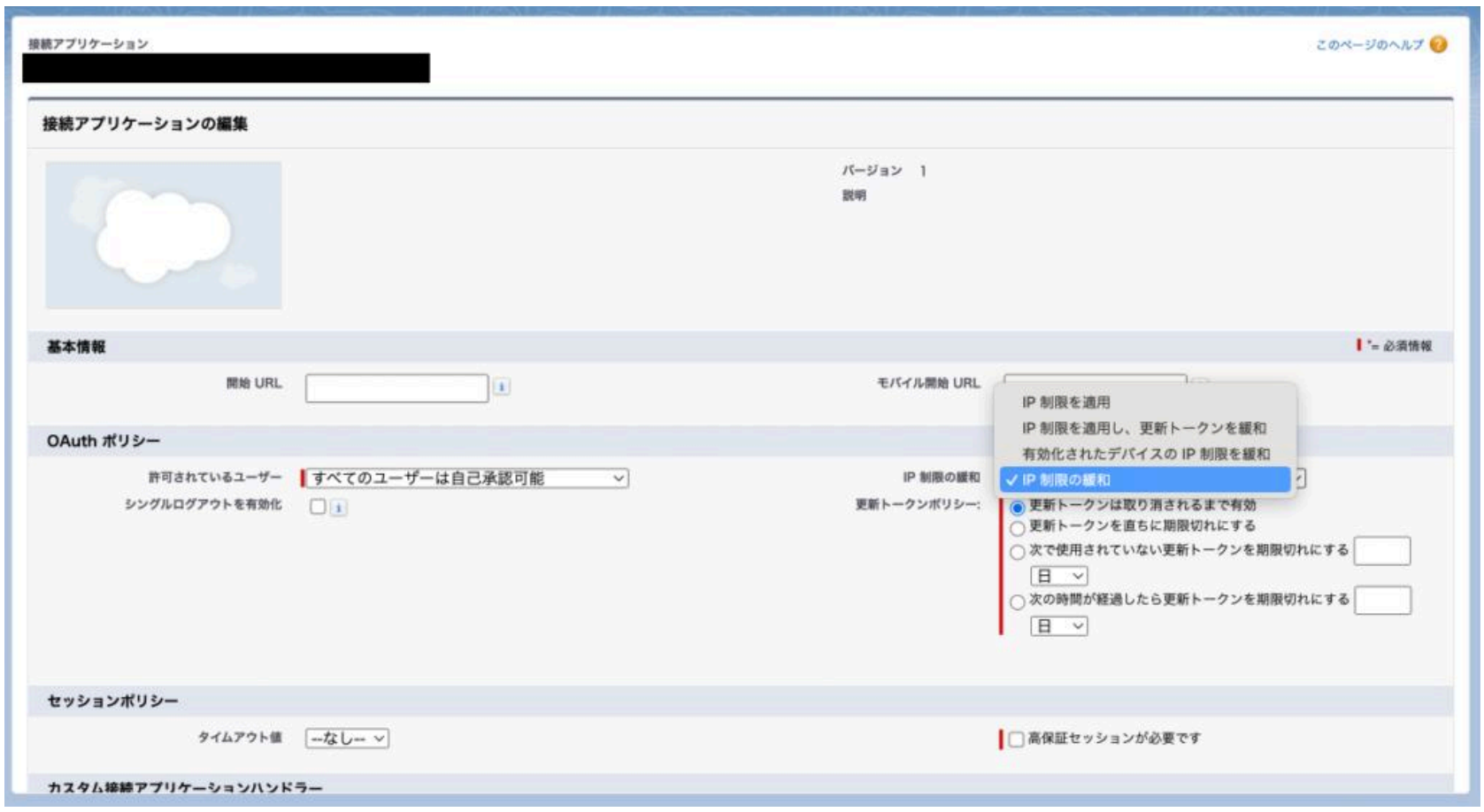
【解決・回避策②】Proximo Addonを使用する

Private Spaceプランを使用しない場合、HerokuのProximo Addonの利用をお勧めします。Proximoは、プロキシサーバーの役割を果たします。

HerokuからアウトバウンドでのAPIコールを行う際に、Proximoを経由して実行することで、Dynoの可変なIPアドレスに関係なく、Proximoの固定IPアドレスからのアクセスとなります。

【解決・回避策③】（Salesforceへの接続の場合のみ）接続アプリケーションのIP制限の緩和を使用する

HerokuからSalesforceへOAuth2.0認証を使用してAPIを実行する場合、認可に使用するSalesforceの接続アプリケーションでIP制限の緩和を設定することで、IPアドレス制限の問題を回避できます。



この設定により、指定した接続アプリケーションからのアクセスに限り、IPアドレス制限が緩和されます。ただし、この方法はSalesforceへの接続に限定されるため、他のSaaSへのAPI接続には適用できないことに注意してください。

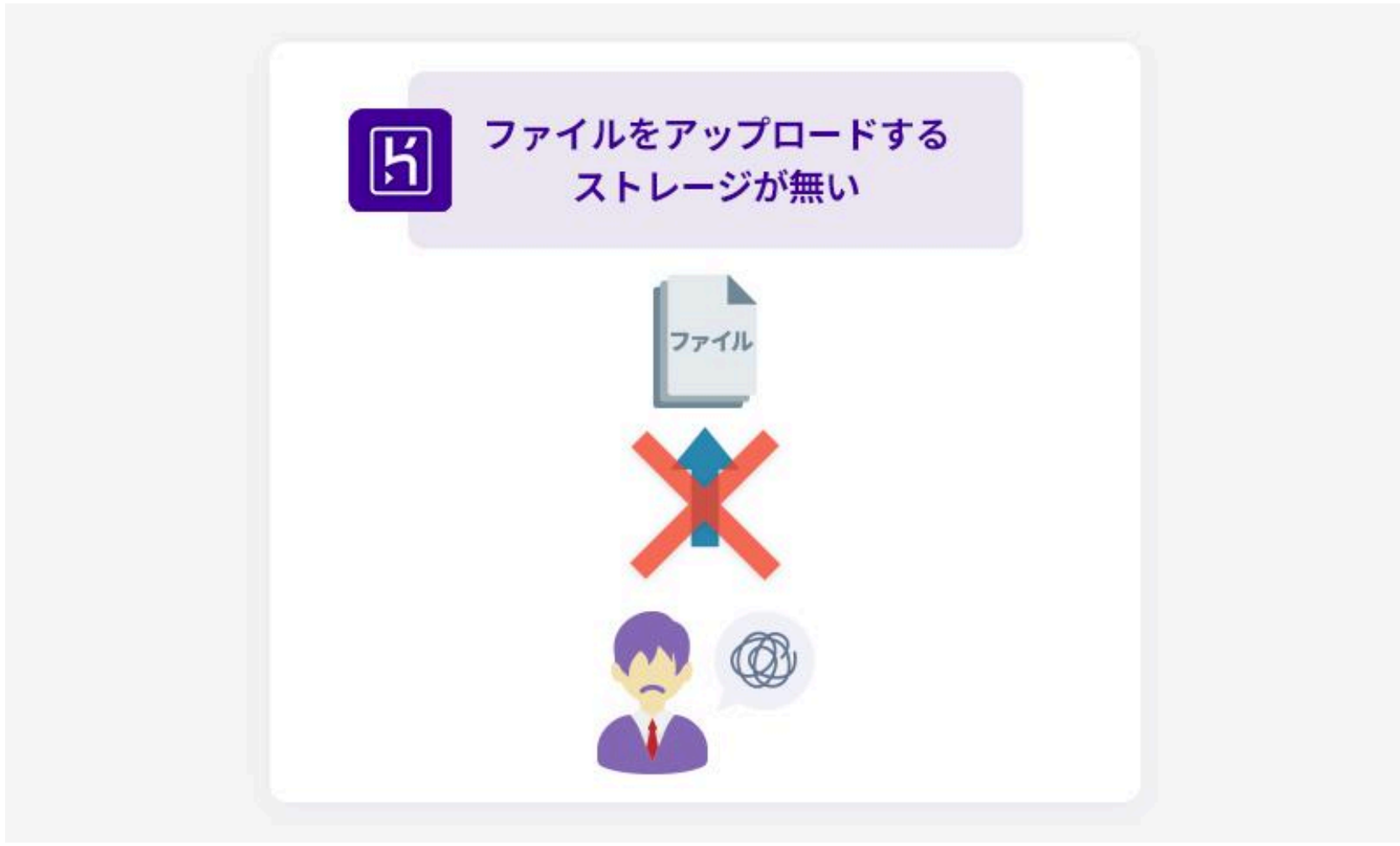
CHAPTER03

3. 静的ストレージが無い問題

3. 静的ストレージが無い問題

問題の概要

Herokuで作成するポータルには、デフォルトでユーザーがファイルをアップロードするためのストレージ領域がありません。Herokuのファイルシステムに直接保存しても、Dynoのリブート時にファイルは消えてしまいます。また、Heroku Postgresのレコードにファイルデータを直接保存するのも、容量やパフォーマンスの観点から現実的ではありません。



解決策

この問題に対する解決策は、外部のクラウドストレージを使用することです。例えば、Amazon S3を外部ストレージとして使用し、実際のファイルをS3に保存し、Heroku PostgresにはS3のファイルへのリンク情報などを保存する方法があります。

Amazon S3に関する認証情報はHerokuのサーバーサイドのみに保存すれば、ユーザーがAWSのライセンスを持たなくても連携が可能です。また、Heroku ConnectでSalesforce側に連携する際も、連携される情報自体はS3のファイルへのリンク情報等のテキスト情報なので、問題なく対応できます。

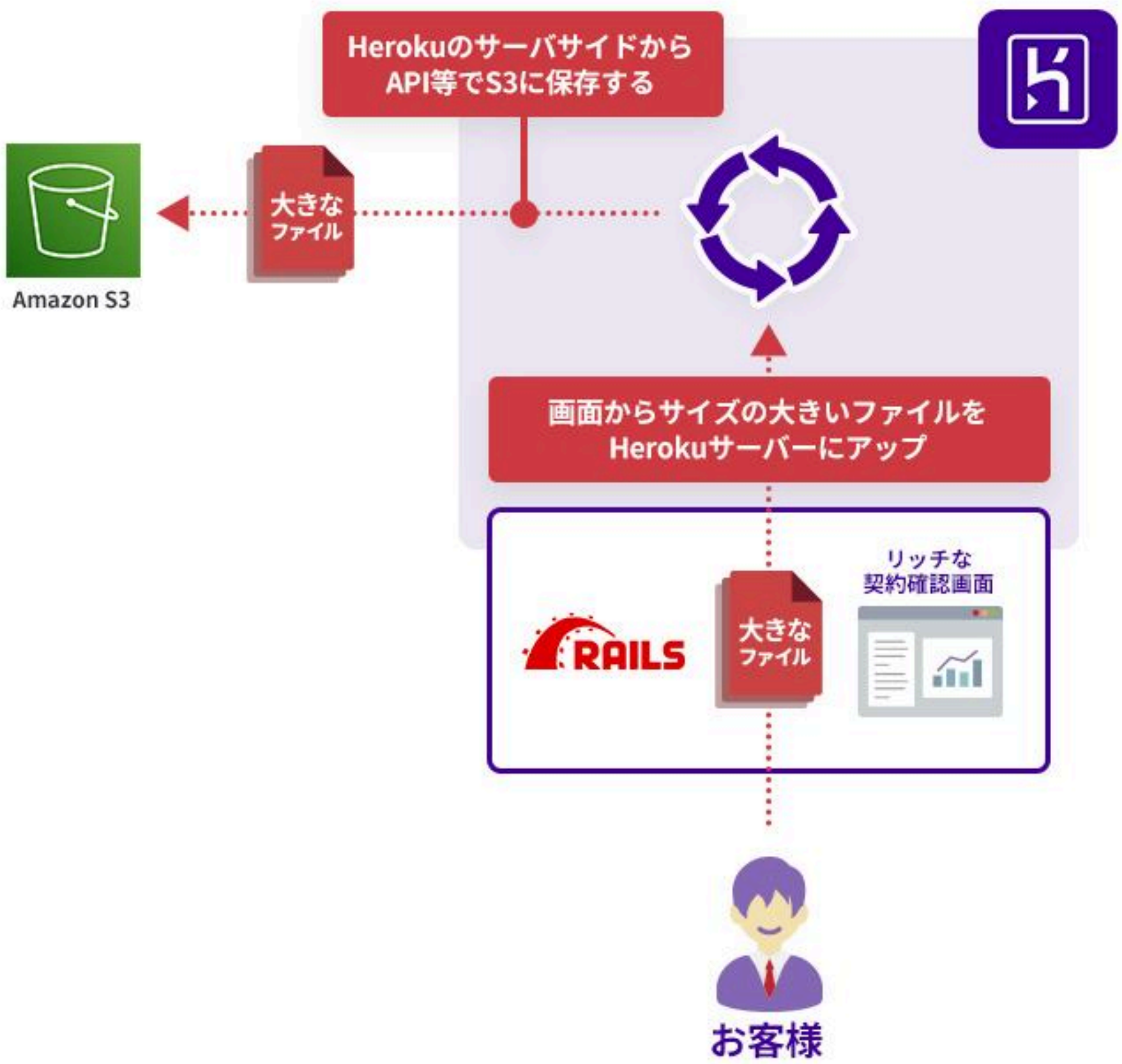
ただし、この方法を実装する際のシステム構成には注意が必要です。S3にもREST APIがあるため、よくある（しかし問題のある）システム構成として、以下のようなものが考えられます：

解決策

この問題に対する解決策は、外部のクラウドストレージを使用することです。例えば、Amazon S3を外部ストレージとして使用し、実際のファイルをS3に保存し、Heroku PostgresにはS3のファイルへのリンク情報などを保存する方法があります。

Amazon S3に関する認証情報はHerokuのサーバーサイドのみに保存すれば、ユーザーがAWSのライセンスを持たなくても連携が可能です。また、Heroku ConnectでSalesforce側に連携する際も、連携される情報自体はS3のファイルへのリンク情報等のテキスト情報なので、問題なく対応できます。

ただし、この方法を実装する際のシステム構成には注意が必要です。S3にもREST APIがあるため、よくある（しかし問題のある）システム構成として、以下のようなものが考えられます：



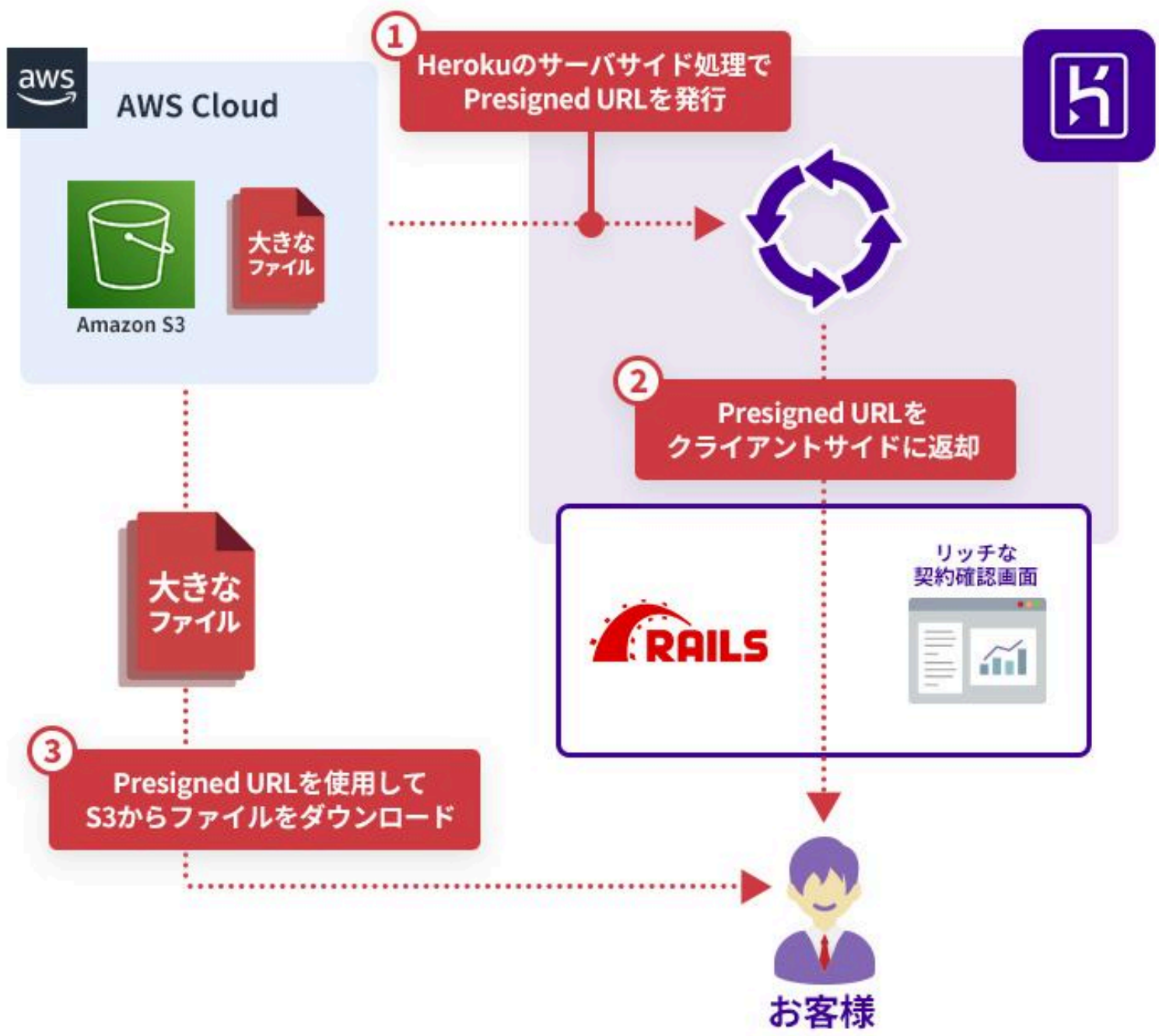
一見すると正しそうに見えますが、このシステム構成では大容量ファイルのアップロード時に問題が発生します。HerokuのWeb Dynoのメモリ使用量が急上昇し、メモリのスワップによりWebシステム全体のスピードが悪化し、最悪の場合Webシステムがダウンしてしまう可能性があります（メモリのオーバーフロー）。

大容量ファイルをアップロードするには

この問題を解決するためには、Amazon S3のPresigned URL（事前署名付きURL）を使用する方法があります。Presigned URLは一定時間のみ有効な特別なURLで、これを使用することでエンドユーザーのクライアントから直接S3にアップロードを行うことができます。

この方法を使用すれば、大容量ファイルはアプリケーションサーバーを経由しないため、アプリケーションサーバーのメモリが急上昇するような問題は発生しません。S3自体は大容量ファイルにも十分に対応できるクラウドストレージであるため、ファイルサイズの問題も解決できます。

Amazon S3のPresigned URLと同様の機能は他のクラウドプロバイダーでも提供されており、同じ活用方法が可能です。例えば、Google Cloud PlatformのCloud StorageではSigned URL、Microsoft AzureではShared Access Signatures (SAS)という名称で同様の機能が提供されています。

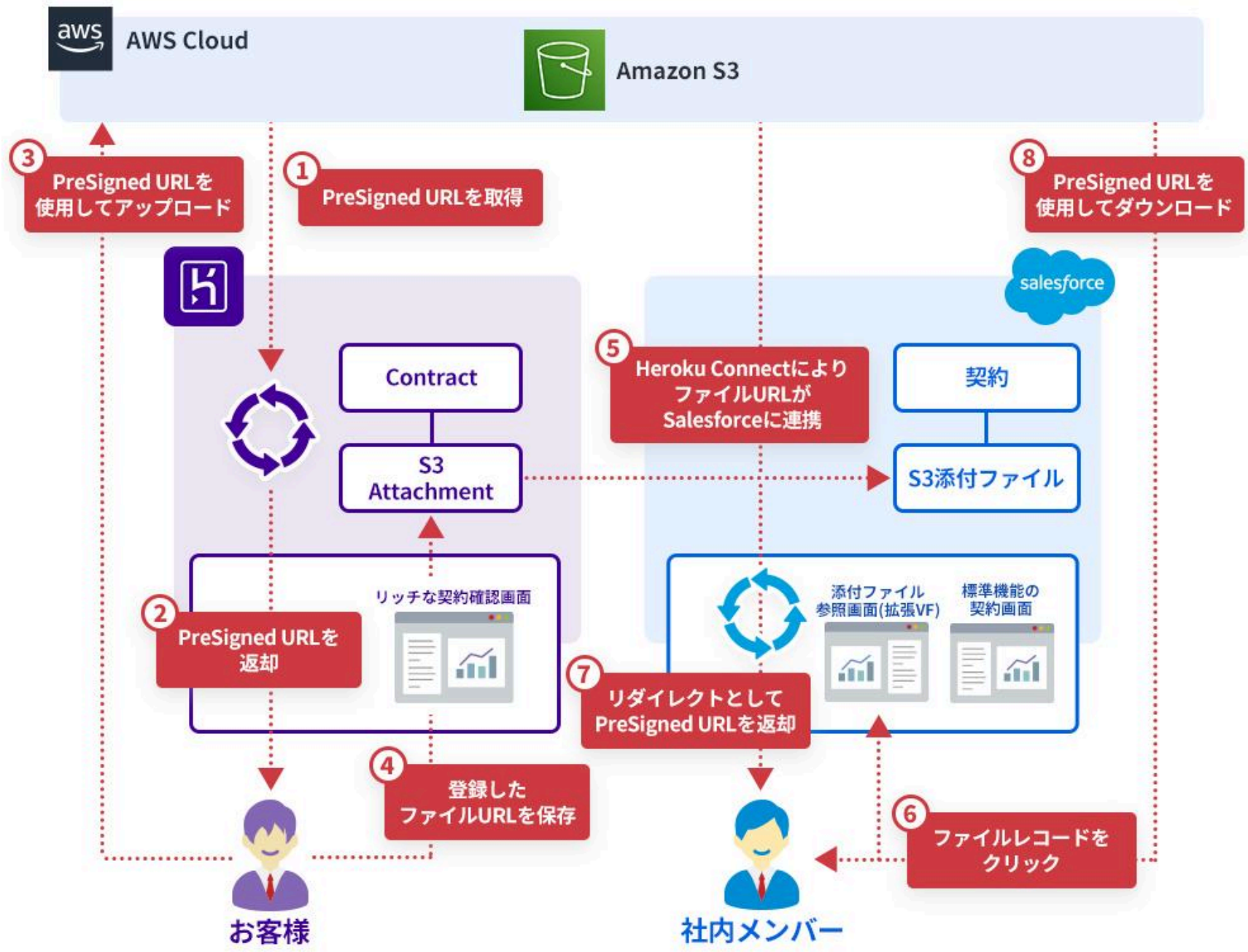


Salesforceとの連携における考慮点

Heroku Connectを使用してファイルの識別情報のみをSalesforce組織に連携し、そのファイル情報を参照するケースを考えてみましょう。

SalesforceのApexにも厳しいガバナ制限が適用されているため、サーバーサイドで大容量ファイルの実体を経由させてしまうと、ダウンロードする前にガバナ制限エラーになってしまいます。

これを解決するには、Herokuの場合と同様に、サーバーサイドのApexからPresigned URLを発行し、Salesforceユーザーにそのpresigned URLを渡します。ユーザーは自身のクライアントから直接S3からファイルをダウンロードすることで、問題を回避できます。



CHAPTER04

まとめ

まとめ

Herokuは、インフラ設定を極力簡略化したプラットフォームです。そのため、インフラエンジニアを必要とせず、アプリケーション開発者が開発に専念できる環境を提供しています。
しかし、その簡略化ゆえの制約も存在します。本記事で紹介した「30秒制限」「動的IPアドレス」「静的ストレージの不在」といった”クセ”は、適切に対処しなければシステムの大きな制約となってしまう可能性があります。
これらを適切に理解し対処することで、Herokuは多くのプロジェクトにおいて非常に強力なツールとなります。本記事で紹介した方法を参考に、プロジェクトの要件に合わせてHerokuを最適に活用していただければ嬉しいです。