Google Cloud

Solution Session Report

推奨プラクティスで EC サイトを “モダン” なアーキテクチャへの移行を実践

徹底解説！ Google Cloud モダン アーキテクチャ パターン

Session overview

昨今、サービスを高速かつ高品質に開発、リリースし、運用の負荷を軽減するという普遍的なゴールに対し、さまざまなプラクティスが登場しています。たとえば、コンテナやサービスメッシュ、DevOps を中心に、各テクノロジーが何を目的としているのか、そのメリットとデメリットを理解するとともに、Google Cloud で実現するためにはどのようなアーキテクチャが適切なのか、具体例を示しながら説明します。

Speaker

Google Cloud
ソリューションズ アーキテクト
長谷部 光治

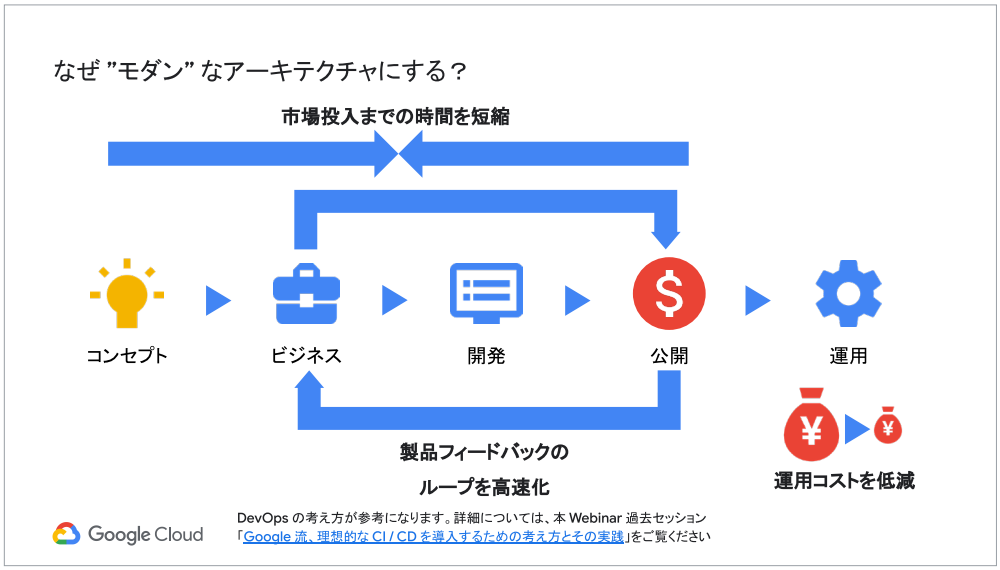
ゲーム会社でインフラ エンジニアとしてキャリアをスタート。その後、ユーザー企業でプライベート クラウド開発、パブリック クラウド導入をリード。外資系ソフトウェア ベンダーでは、クラウドのプリセールスからデリバリまで幅広く担当し、2018 年 9 月に Google Cloud Japan に参画。 現在は Solutions Architect として、「Hybrid」「Retail」を軸に、お客様のクラウド利用促進に従事しています。

Index

- Google Cloud が考えるアプリケーションの未来 ····· P.2
- コンテナ化とマイクロ サービスの課題を解消する Kubernetes ····· P.3
- サービス メッシュを利用してグローバル展開 ····· P.5
- 宣言型で定義して単一の仕組みで管理する ····· P.7
- 参照リンク ····· P.11

Google Cloud が考えるアプリケーションの未来

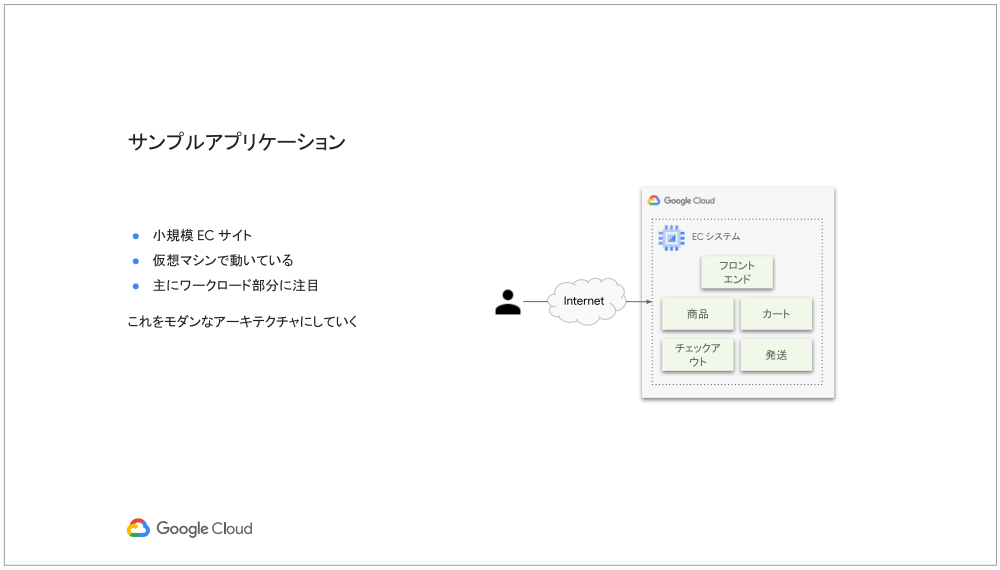
Google Cloud では、可用性、性能・拡張性、運用保守性、セキュリティのそれぞれの項目を高いレベルで満たしたアプリケーションを構築するためのアーキテクチャを、“モダン” なアーキテクチャと位置づけています。新しいサービスをリリースするときの、コンセプト、ビジネス、開発、公開、運用という手順を、モダンなアーキテクチャにする理由は、市場投入までの時間を短縮したい、製品フィードバックのループを高速化したい、運用コストを低減したいという 3 つです。



モダンなアーキテクチャにする理由は、市場投入までの時間を短縮、製品フィードバックのループを高速化、運用コストの低減の 3 つ。

モダンなアーキテクチャと同義になりますが、Google Cloud が考えるアプリケーションの未来では、それを動かすインフラは、「コンテナ化されたマイクロ サービス」で作られ、「宣言型で定義され、単一の仕組みを通じて管理」されます。また、「サービス メッシュを利用して、あらゆるロケーションにサービスを提供」できます。Google Cloud が提供する各種サービスも、このモダンなアーキテクチャに基づいて実現されています。

このセッションでは、フロントエンド、商品、カート、チェックアウト、発送の 5 つの機能で構成され、仮想マシンで動いている小規模な EC サイトを想定したサンプル アプリケーションを、モダンなアーキテクチャに変更する手順を紹介します。



サンプル アプリケーションは、フロントエンド、商品、カート、チェックアウト、発送の 5 つの機能で構成。

コンテナ化とマイクロ サービスの課題を解消する Kubernetes

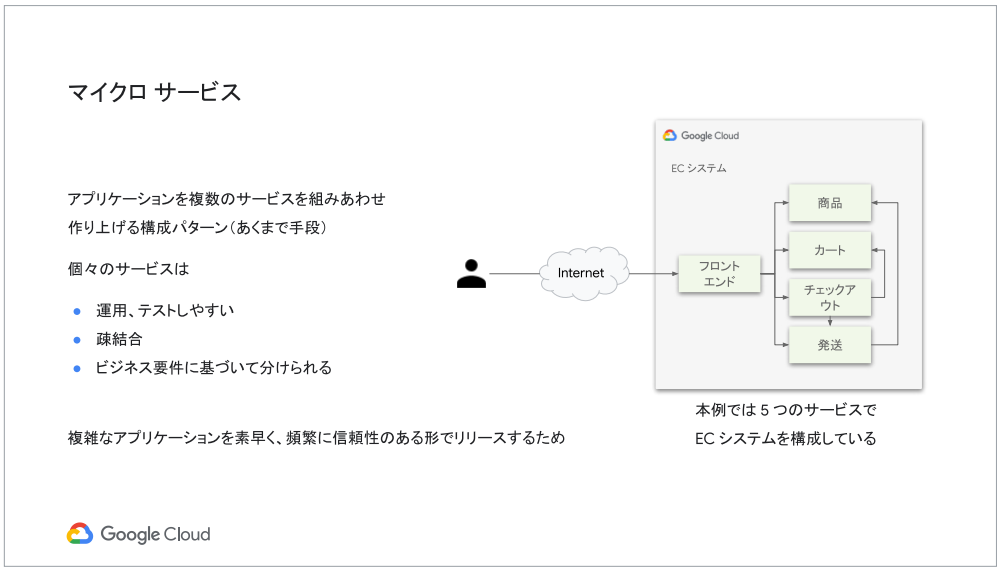
まずは、コンテナ化とマイクロ サービスについて紹介します。コンテナは、アプリケーションと、その実行に必要なものをまとめたイメージです。通常、1 コンテナあたり 1 プロセスで開発されます。コンテナは、形式が標準化されているので、コンテナ ランタイムが動く環境であれば、オンプレミスでも、クラウドでも動かすことができます。

コンテナを使うメリットは、VM より高速に、数ミリ秒で起動する高速性、コンテナ実行環境間であれば容易に移行できるポータビリティ、少ないオーバーヘッドでリソースを有効利用できる効率性です。たとえば、仮想マシンを 5 台しか動かせない物理サーバーでも、コンテナであれば数十台から 100 台程度動かすことができます。

コンテナは形式が標準化され、コンテナ ランタイムが動く環境であればどこでも動かすことができます。

マイクロ サービスは、複数の小さなサービスを組みあわせてアプリケーションを作る構成パターンです。個々のサービスは、運用やテストがしやすく、疎結合で、ビジネス要件に基づいて分割されます。複雑なアプリケーションを、素早く、頻繁に、信頼性のある形でリリースするための構成であり、複雑でなく、頻繁にリリースされないアプリケーションは、本当にマイクロ サービス化が必要かをよく検討する必要があります。

サンプル アプリケーションをマイクロ サービス化すると図のようになります。



フロントエンドで注文を受けて、商品、カート、チェックアウト、発送の各マイクロ サービスが連携しあって、1 つの EC サイトを構成していることがポイントです。

コンテナ化は、多くのメリットがありますが、コンテナ管理やネットワーク、スケーリング、デプロイ・更新、コンテナ間のディスカバリ、ヘルスチェック、永続データの扱い、ロギング・監視、デバッグなどの細かい管理が煩雑になります。この課題を解決するのが、コンテナ オーケストレーションのデファクトである Kubernetes です。

コンテナ化とマイクロ サービスを、Google Cloud 上で実現するための推奨プラクティスは、以下の 4 つです。

（1）権限、コスト、リソース上限値の分離

Google Cloud のプロジェクトという概念を適切に使い、利用可能な API、課金、コスト情報、リソースに対する権限、リソース上限値を分離します。

（2）あらゆるレイヤーで可用性、拡張性を担保

Kubernetes のポッド、マルチゾーンやマルチ リージョンへの展開など、あらゆるレイヤーで可用性、拡張性が担保されていることが必要です。

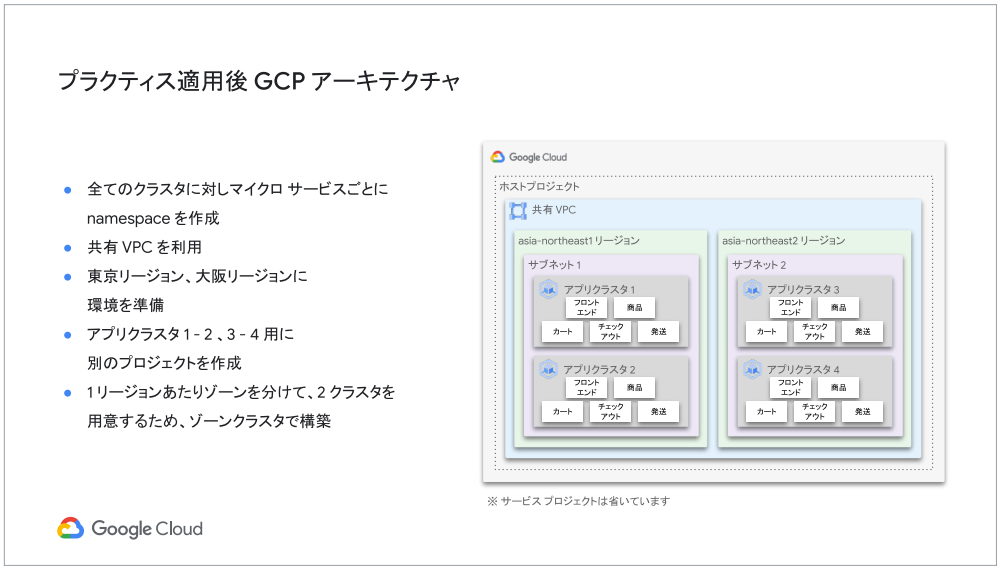
（3）クラスタの有効利用、一貫した管理

クラスタを複数のテナントで共有するマルチテナントを想定してクラスタを構成することが重要です。クラスタでは、namespace でテナントを分離します。

（4）ネットワーク構成をシンプルに

共有 VPC を使い、フラットなネットワークにして、プロジェクトを分離することで、ネットワーク構成をシンプルにすることが必要です。

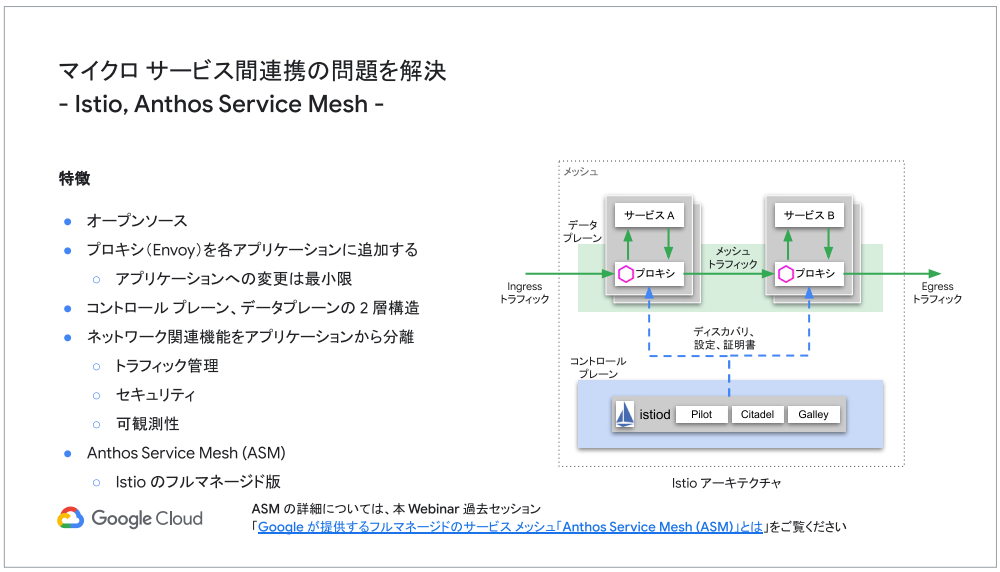
4 つの推奨プラクティスを、サンプル アプリケーションに適用することで、図のようにコンテナ化、マイクロ サービス化が実現できます。



共有 VPC でネットワークをシンプルにして、マルチ リージョン、マルチゾーンで可用性、拡張性を担保し、一貫した管理を実現します。

サービス メッシュを利用してグローバル展開

サービス メッシュは、アプリケーションを構成するマイクロ サービス間のネットワークとそれらの連携の問題を解決する方法です。多くのマイクロ サービスで構成されたアプリケーションは、サービス間の連携が複雑化し、管理が困難になります。この課題を解決するオープンソースが Istio です。Google Cloud では、Istio のフルマネージド サービスである Anthos Service Mesh（ASM）を提供しています。



Istio は、コントロール プレーン、データプレーンの 2 層構造で、ネットワーク関連機能をアプリケーションから分離できます。

また、アーキテクチャ設計の最初段階から、ネットワーク、クラスタ、ロードバランサのすべてで、グローバル展開を見据えたアーキテクチャ構成を採用することが必要です。たとえば、東京リージョン、大阪リージョンの 2 リージョン構成から検討を進めておくことで、グローバル展開する場合にも、簡単にリージョンを増やすことができます。

サービス メッシュ、グローバル展開における推奨プラクティスは、以下の 3 つです。

(1) コントロール プレーンの可用性、拡張性を確保

Istio アーキテクチャ設計では、可用性、拡張性の向上でマルチクラスタ、マルチ ネットワークを、設定反映範囲の分離でマルチ コントロールプレーンを、構成のシンプルさでシングル メッシュを基本とします。

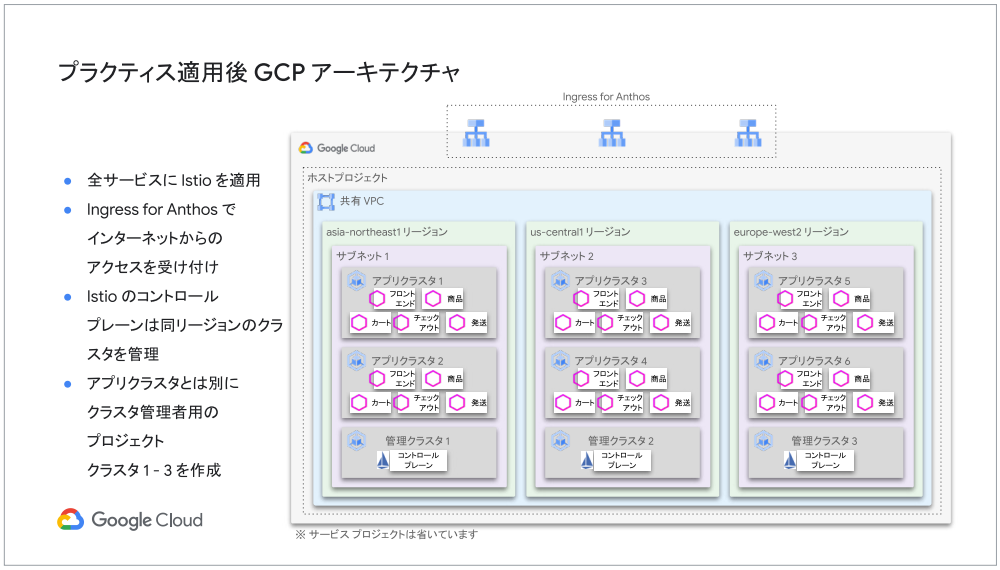
(2) サービスの配置をシンプルに

サービスの配置をシンプルにするには、ゾーン、リージョンごとの構成を同じにすることです。これにより、管理もしやすくなり、可用性も担保できます。さらに、各ロケーションでスケーリングもできます。

(3) 利用者へのレスポンスを第一に考える

アプリケーションに関しては、利用者に近いところにアプリケーションを配置することで、レスポンスを第一に考えます。Google Cloud では、複数のクラスタでアプリケーションを稼働したときに、最も利用者に近いところにルーティングするマルチクラスタ Ingress コントローラである Ingress for Anthos を提供しています。

サービス メッシュ、グローバル展開の推奨プラクティスを、サンプル アプリケーションに適用することで、図のようになります。



コントロール プレーンの可用性、拡張性を確保し、サービスの配置をシンプルにして、利用者へのレスポンスを第一に考えます。

プラクティス適用後のデータベースは、要件によって変化しますが、スケーラブルで可用性の高いミッション クリティカル RDBMS である Cloud Spanner を利用します。Cloud Spanner は、ゾーンやリージョン間の可用性の機能をデータベース自体が持っているので、バックエンドの構成に悩むことなく、要件に応じた適切な構成を実現できます。

宣言型で定義して単一の仕組みで管理する

宣言型による定義とは、あるべき状態を記述することです。あるべき状態が B の場合、いまの状態が A なのか、C なのかということは定義上記述していません。あるべき状態が B ですと記述しておけば、自動的に B の状態に移行します。ソフトウェアとしては、Kubernetes や Istio、Terraform などが主流になりつつあります。

宣言型は、手続きの部分であるべき状態に至る手順が隠蔽されているので、問題が発生した場合、修正や細かい制御が難しい課題もあります。

EC サイトの仕組みが分散すると管理対象が増えてしまうので、単一の仕組みで管理できることも必要です。設定情報のような簡単なものから UI まで、1 つの画面で管理できることが重要になります。

宣言型による定義、単一の仕組みで管理の推奨プラクティスは、以下の 2 つです。

(1)　すべてをコードで管理

Infrastructure as Code（IaC）で、設定ファイルから運用のダッシュボードまで、できる限りコード化して管理します。

(2) コードを開発プロセスと同様に管理

Git リポジトリに IaC のコードを格納することで、アプリケーションのソースコードと同様のプロセスでバージョン管理します。

ローカルで作成から本番に反映するまで、アプリケーション コードだけでなく、インフラコードも含めて同様に管理します。

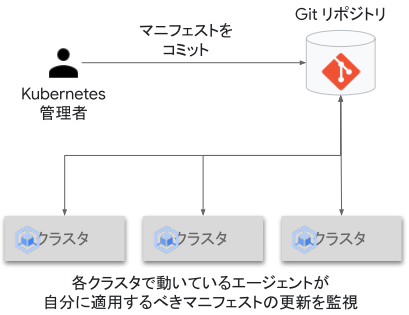
Kubernetes のマニフェストも GitOps で管理しますが、Google Cloud では、管理機能として Anthos Config Management（ACM）を提供しています。

Kubernetes のマニフェストを GitOps で管理
- Anthos Config Management (ACM) -

グローバルに展開されたクラスタ群のマニフェスト
（Namespace, Quota, RoleBinding など）を一元管理、適用

特徴

- Git リポジトリに構成の定義を格納
- 構成定義の検証
- 構成定義の自動適用



Google Cloud

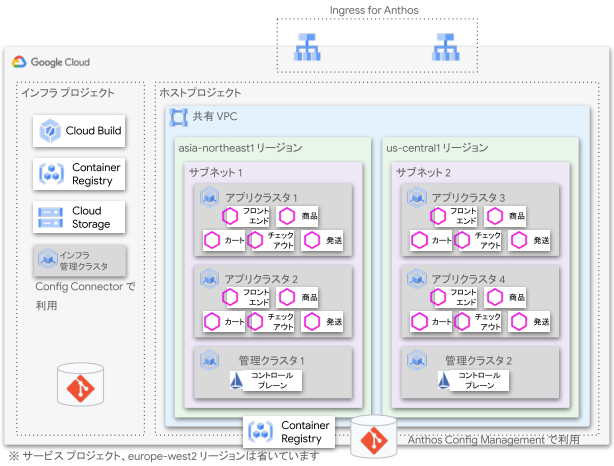
ACM により、Git リポジトリに構成の定義を格納して、構成定義の検証や構成定義の自動適用が可能です。

宣言型による定義、単一の仕組みで管理の推奨プラクティスを、サンプル アプリケーションに適用することで、図のようになります。

プラクティス適用後 GCP アーキテクチャ

- クラスタ管理者プロジェクトに Git リポジトリを作成し、ACM を通じて全クラスタを管理する
- 基礎となる GCP リソース（VPC、プロジェクトなど）を Config Connector* または、Terraform を使って管理する
  インフラ プロジェクトを作成

* Config Connector とは Kubernetes から GCP リソースを管理するツールで、ACM とも連携可能



Google Cloud

EC サイトをモダンなアーキテクチャに移行することで、サービスメッシュも含め、グローバルに分散するサービス群ができあがります。

モダンなアーキテクチャで実装したサンプル アプリケーションを本番導入する場合、“モダン”なテクノロジーの採用により、管理対象の増加、更新への追従対応増加、問題発生時の対応増加などの課題が生まれます。そこで、できる限りマネージド サービスを活用することが必要です。

マネージド サービスを最大限に活用することで、Google Cloud により事前にテスト済みのバージョンを利用できる、Google Cloud による運用を利用できる、利用状況により自動スケーリングできるなど、モダンなテクノロジーの良いとこ取りが可能です。

また、新たな作業、運用の負荷を抑え、モダンなテクノロジーのメリットを享受できるツールとして、Google Cloud では、Anthos を提供しています。Google Cloud のテクノロジーを活用することで、本来の業務に注力しつつ、モダンなアーキテクチャを実現することが可能になります。

Google Cloud では、モダンなテクノロジーをより簡単に使えるようにするソフトウェア スイート Anthos を提供しています。

Links

- Google Cloud のソリューション｜アプリケーションのモダナイゼーション :
  goo.gle/app_modern
- Google Kubernetes Engine 製品紹介ページ :
  goo.gle/gke
- Anthos Service Mesh 製品紹介ページ :
  goo.gle/service_mesh
- Istio 製品紹介ページ :
  goo.gle/istio
- インフラストラクチャのコード管理 紹介ページ :
  goo.gle/infra_code

製品、サービスに関するお問い合わせ

goo.gl/CCZL78

Google Cloud の詳細については、上記 URL もしくは QR コードからアクセスしていただくか、同ページ「お問い合わせ」よりお問い合わせください。