IT資格道場

GCP PNE 学習用テキスト

予備知識ゼロから合格ラインへ —— 読んで理解する1冊目

無料サンプル(第1章まで)

Google Cloud 公式 Professional Cloud Network Engineer Exam Guide(2026-09-07 取得)準拠
2026年7月版

本書のご利用にあたって

- 本書は「IT資格道場」(www.shikaku-doujou.com)の購入者限定テキストのうち、学習用テキストの第1章までを収録した**無料サンプル**です。どなたでもご覧いただけます。
- 無料サンプルであっても著作物です。本書の転載・改変しての再配布・二次販売を禁じます。
- 本書を AI モデルの学習データとして利用すること(学習目的の入力・データセット化を含む)を禁じます。
- 製品版(学習用テキスト・頻出度別ノート・暗記用ノートの3点)は、ご購入者を識別できる透かし入りのPDFとして提供されます。
- 本PDFはスマートフォン・タブレットでの閲覧に合わせたA5版面・文字サイズで組版しています。

不正な配布を見つけた場合は info@shikaku-doujou.com までご連絡ください。

目次

3点セットの使い方

種類	役割
① 学習用テキスト(本書)	これだけで問題が解けるようになる本編
② 頻出度別ノート	テキストを読んだら次はこれ。頻出順に絞った問題と解説で「解ける」に橋渡し。試験直前の最終チェックにも
③ 暗記用ノート	覚えるべき要点だけを一問一答に凝縮。空き時間の反復と用語の再確認用

使い方: ① 学習用を読む → ② 頻出度別で解き方を掴む → ③ 問題演習で腕試し → ④ 頻出度別に戻って再確認(試験直前もここ)。暗記用は空き時間や用語の再確認に。

このテキストの位置づけ: 本書は①の学習用テキストです。まずはここを通読し、これだけで問題が解けるようになることを目標にしてください。

はじめに —— この教材の使い方

このテキストは、Google Cloud が実施する **Professional Cloud Network Engineer 認定** の合格を目的に作られています。

Professional 級の試験が問うのは、個々の製品の説明ではありません。**要件と制約を読み取り、複数の選択肢の中から最も適切な設計・運用・トラブルシューティングの判断を選べること**です。本書は Exam guide の各セクションに沿って、「どの場面でどれを選ぶか・なぜ他は不適か」を本文に書き込んでいます。

本書は Google Cloud の公式教材ではなく、IT資格道場が独自に書き下ろしたオリジナルの教材です。実際に出題された問題を再現したものではありません。

試験の基本情報

項目	内容
試験名	Google Cloud Certified – Professional Cloud Network Engineer
実施	Google Cloud（テストセンター、またはオンライン監督試験）
問題数	**50〜60問**（公式の案内）
試験時間	**120分**
出題形式	**択一（multiple choice）、複数選択（multiple select）**
合格ライン	非公表
受験言語	英語・日本語（日本語提供あり）

出題数・試験時間・試験ガイドは改定されることがあります。お申し込みの前に、Google Cloud 公式サイトで最新の情報をご確認ください。

出題範囲の全体像 —— 6つのセクションと本書の章

セクション	分野	採点対象に占める割合（公式）	本書
1	Google Cloud VPC ネットワークの設計と計画	**21%**	第1章

セクション	分野	採点対象に占める割合（公式）	本書
2	VPC ネットワークの実装	**20%**	第2章
3	マネージドネットワークサービスの構成	**16%**	第3章
4	ハイブリッド・マルチクラウドのネットワーク相互接続の構成と実装	**16%**	第4章
5	ネットワーク運用の管理・モニタリング・トラブルシューティング	**14%**	第5章
6	クラウドネットワークセキュリティソリューションの構成・実装・管理	**13%**	第6章

本書の章の並びは、Exam guide の並びとまったく同じです。節見出しの末尾にある `[1.1]` のような番号は、Exam guide のセクション番号です。Google Cloud はセクションごとの割合だけを公表しており、細目単位の出題数は公表していません。本書もそれ以上の数字は書きません。

サービス名・機能名は英語のまま覚えてください。本文では、製品名・機能名・用語を英語表記のまま書いています。本番の設問文と選択肢に出るのはこの表記であり、日本語訳だけを覚えても画面では出会いません。

全設問に効く判断軸 ——「3つの問い」

問い	何を切り分けるか	分かれ方
問い1: 要件は何を最優先しているか	可用性／性能／費用／セキュリティ／運用負荷	同じ「移行したい」でも、何を最優先に置くかで正解の製品と構成が変わります
問い2: マネージドで足りるか、自分で管理するか	サーバーレス／マネージド／セルフマネージド	「運用したくない」ならマネージド、「細かな制御が要る」ならセルフマネージド、と制約が構成を決めます
問い3: それは Google の責任か、自分の責任か	責任共有モデル	どの層まで自分で守り・運用するかは全章で一貫して効きます。迷ったらここへ戻ってください

第1章 Google Cloud VPC ネットワークの設計と計画（ドメイン1・採点対象の 21%）

このドメインは全6ドメインのなかで最大の配点です。特徴は「作り方」ではなく「決め方」を問うことにあります。同じ要件でも、Shared VPC を選ぶか Network Connectivity Center を選ぶかで運用コストも障害時の影響範囲も変わります。本章では、設計の分岐点ごとに「どれを選ぶか」「なぜ他が不適か」を対比で押さえていきます。

1-1 ネットワークアーキテクチャ全体の設計 [1.1]

1-1-1 Network Service Tiers —— Premium と Standard の選び分け [1.1]

Google Cloud の外向きトラフィックには **Network Service Tiers**（ネットワークサービスティア）という2段階の選択肢があります。これは「Google のバックボーンをどこまで使うか」の選択です。

観点	Premium Tier	Standard Tier
経路	送信元/宛先に最も近い **PoP (Point of Presence)** から Google の専用バックボーンに乗せ、リージョンまで Google 網内を通す（cold potato routing）	できるだけ早く一般のインターネット（ISP 網）へ受け渡す（hot potato routing）
品質	低レイテンシ・低パケットロス・高可用。SLA 対象	ベストエフォート。インターネットの混雑の影響を受ける

観点	Premium Tier	Standard Tier
スコープ	**グローバル**。単一の anycast IP で全世界から受けられる	**リージョナル**。IP はリージョンに紐づく
使えるロードバランサ	全種類（global external Application Load Balancer など）	regional external Application Load Balancer、regional external proxy Network Load Balancer、regional external passthrough Network Load Balancer のみ
Cloud CDN	使える	使えない
料金	高い	安い
既定値	プロジェクト既定は Premium	明示指定が必要

設計判断の型：エンドユーザー向けの本番サービス、グローバル配信、CDN 併用、SLA が要る ── ここは Premium です。逆に、バッチのログ転送・開発環境・単一リージョン内で完結する社内向けなど、「ユーザー体感より費用が支配的」な用途は Standard に落とす余地があります。

よく出る取り違え：

- 「Standard にすればグローバル LB のコストが下がる」は誤りです。**グローバルなロードバランサは Standard Tier では構成できません**。Standard を選んだ時点でリージョナルな構成に縛られます。
- Tier は VM の外部 IP 単位・転送ルール単位でも指定できます。プロジェクト既定を Standard にしても、個別リソースで Premium を上書きできま

す。

- **VPC 内部の VM 間通信は Tier の対象外**です。Tier が効くのは「インターネットとの間の egress」だけで、内部通信の品質は変わりません。

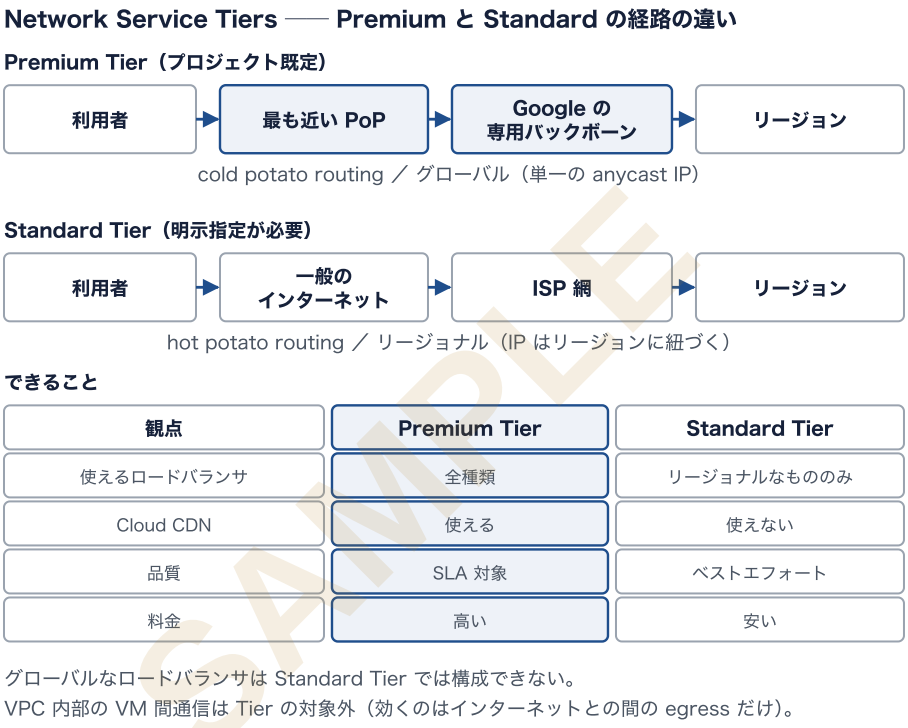


図 1-1　Network Service Tiers —— Premium と Standard の経路の違い

1-1-2 高可用性・フェイルオーバー・災害復旧・スケールの設計 [1.1]

high availability（高可用性）、**failover（フェイルオーバー）**、**disaster recovery（災害復旧）**、**scale（スケール）**の4語は、ネットワーク設計では別々の対策に落ちます。混同すると設問で必ず外します。

語	守る対象	ネットワーク側の実装
High availability	単一障害点の除去（ゾーン・機器・回線の1つが落ちても継続）	HA VPN（2トンネル）、Dedicated Interconnect の冗長構成、複数ゾーンにまたがる managed instance group、リージョナル LB
Failover	落ちた時に自動で別経路・別バックエンドへ切り替わる	BGP の MED（route priority）、Cloud DNS の failover routing policy、internal passthrough Network Load Balancer の failover group
Disaster recovery	リージョン全損からの復旧	別リージョンに待機環境＋global dynamic routing、Cross-Cloud Interconnect や2拠点の Interconnect
Scale	平常時の増加に耐える	サブネットの IP address space の余裕、autoscaling、quotas の事前引き上げ

設計の勘所：Google Cloud の SLA は「構成の作り方」で段階が変わります。Cloud Interconnect は単一の VLAN attachment では SLA 対象外で、**99.9%** と **99.99%** はそれぞれ決められた冗長トポロジを満たしたときに得られます（詳細は第4章）。「SLA を上げたい」という要件は、機器を高性能にする話ではなく、**冗長の配置を規定どおりにする**話だと読み替えてください。

誤答肢の切り方：「可用性を上げたいので Premium Tier にする」「可用性のため MTU を上げる」は、いずれも別の軸の対策です。可用性の問題には冗長経路と自動切り替えで答えます。

1-1-3 DNS トポロジの設計 —— オンプレミスと Cloud DNS [1.1]

DNS topology の設計は、「どの名前空間を誰が正とするか」を決める作業です。ハイブリッド環境では次の4方向を必ず整理します。

方向	目的	使う機能
クラウド → オンプレ	VM が社内の名前を引く	**Cloud DNS の forwarding zone**（転送先にオンプレ DNS サーバー）、または **outbound server policy**
オンプレ → クラウド	社内端末が private zone の名前を引く	**inbound server policy**（VPC にインバウンドフォワーディングの IP を作る）
VPC → VPC	別 VPC の private zone を引く	**DNS peering**
プロジェクト間	別プロジェクトの VPC に zone を見せる	**cross-project binding**

split-horizon DNS（同じドメイン名で内部と外部に別々の答えを返す）は、public zone と private zone を同名で並立させることで実現します。VPC 内からの問い合わせは private zone が優先されます。

よく出る取り違え：forwarding zone と **server policy** は似て非なるものです。forwarding zone は「特定のドメインだけ」を外へ転送します。outbound server policy は「private zone にも Compute Engine 内部 DNS にも当たらなかった残り全部」を転送します。「社内ドメイン example.internal だけをオンプレへ」なら forwarding zone、「クラウドからの名前解決を全部オンプレの DNS で集中管理」なら outbound server policy です。

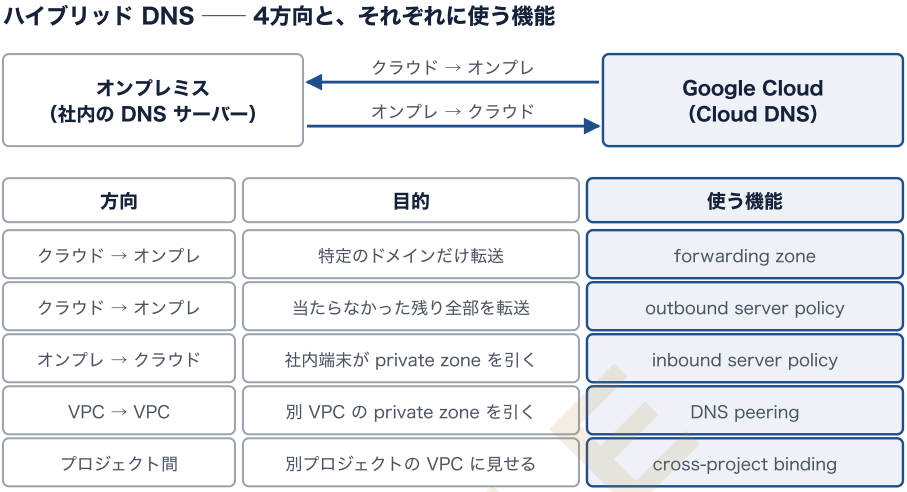


図 1-2　ハイブリッド DNS —— 4方向と、それぞれに使う機能

1-1-4 実装に合ったロードバランサの選択 [1.1]

ロードバランサの選択は、この試験で最も出題密度が高い判断のひとつです。決め方は次の4問を順に答えるだけです。

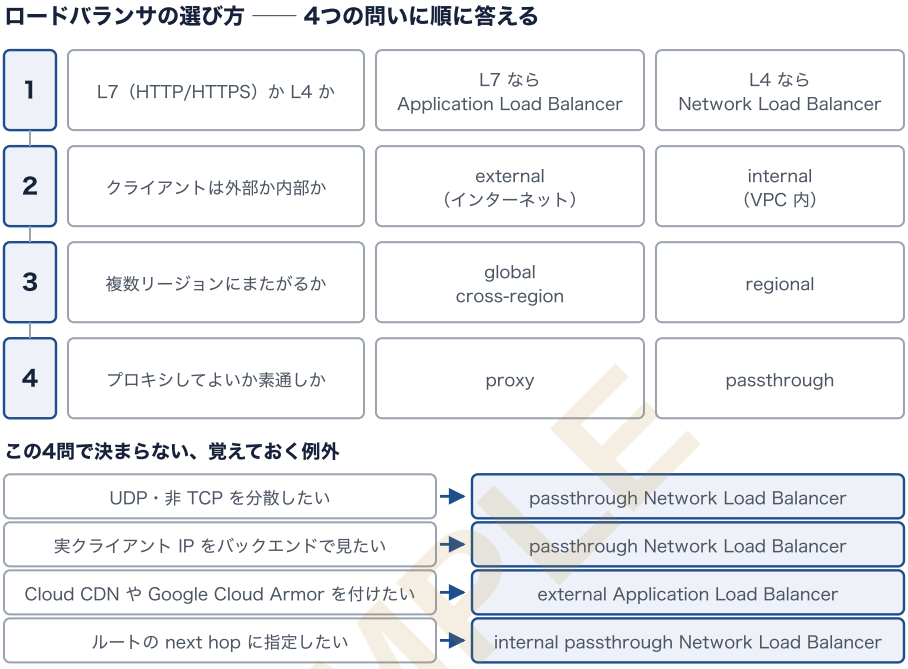
1. **L7（HTTP/HTTPS）か L4 か** → L7 なら Application Load Balancer、L4 なら Network Load Balancer
2. **クライアントは外部（インターネット）か内部（VPC 内）か** → external / internal
3. **複数リージョンにまたがるか** → global / cross-region / regional
4. **プロキシしてよいか、クライアント IP を保ったまま素通しか** → proxy / passthrough

ロードバランサ	層	範囲	代表的な適用場面
global external Application Load Balancer	L7	グローバル（anycast IP）	世界配信の Web／API。Cloud CDN・Google Cloud Armor と組み合わせる標準形
regional external Application Load Balancer	L7	単一リージョン	データ所在の制約がある、または Standard Tier を使いたい Web
cross-region internal Application Load Balancer	L7	複数リージョン・VPC 内のみ	社内向け多リージョン Web。リージョン障害時に別リージョンへ
regional internal Application Load Balancer	L7	単一リージョン・VPC 内	マイクロサービス間の HTTP ルーティング
global external proxy Network Load Balancer	L4（TCP、SSL オフロード可）	グローバル	HTTP 以外の TCP を世界配信
regional internal proxy Network Load Balancer	L4（TCP）	単一リージョン・VPC 内	内部 TCP サービスをプロキシ経由で
external passthrough Network Load Balancer	L4（TCP/UDP/ESP/GRE/ICMP）	リージョナル（Maglev）	UDP・非 TCP プロトコル、クライアント IP の保持が必須

ロードバランサ	層	範囲	代表的な適用場面
internal passthrough Network Load Balancer	L4（TCP/UDP ほか）	リージョナル（Andromeda）	内部の L4 分散。**next hop としてルートに指定できる唯一の LB**

誤答肢の切り方：

- 「UDP を分散したい」→ Application Load Balancer も proxy Network Load Balancer も**不可**。passthrough 一択です。
- 「バックエンドで実クライアント IP をそのまま見たい」→ proxy 型は送信元が Google のプロキシ IP に置き換わるため不適。passthrough を選びます（ALB では `X-Forwarded-For` で渡りますが、L3 の送信元 IP そのものではありません）。
- 「Cloud CDN を付けたい」→ CDN は **external Application Load Balancer** に紐づきます。passthrough には付きません。
- 「third-party appliance への経路として LB を使いたい」→ **internal passthrough Network Load Balancer as a next hop** だけが該当します。



proxy 型は送信元がプロキシの IP に置き換わる（ALB では X-Forwarded-For に元の IP が入る）。

図 1-3　ロードバランサの選び方 —— 4つの問いに順に答える

1-1-5 GKE ネットワーキングの計画 [1.1]

Google Kubernetes Engine (GKE) の設計は、クラスタを作る前に IP の見積もりを終えていないと後から詰みます。VPC-native クラスタは、サブネットの **primary range** をノード用に、**secondary ranges**（エイリアス IP 範囲）を Pod 用と Service 用に使います。

範囲	割り当て先	見積もりの考え方
primary range	ノードの VM	ノード台数の上限＋α

範囲	割り当て先	見積もりの考え方
secondary range（Pod）	Pod	**ノードあたりの最大 Pod 数から逆算**。既定の 110 Pod/ノードなら、GKE は1ノードに /24 を割り当てる
secondary range（Service）	ClusterIP Service	Service 数の上限

scale potential based on IP address space（IP アドレス空間から決まる拡張余地）がこの節の核心です。Pod 用 secondary range に /21 を割り当て、ノードあたり /24 を消費する設定なら、そのクラスタは最大 8 ノードまでしか伸びません。「ノードを増やそうとしたら IP が足りずに失敗した」という設問は、この計算で答えます。ノードあたりの最大 Pod 数を 110 から下げれば、1ノードあたりの消費が小さくなり、同じ範囲でより多くのノードを収容できます。

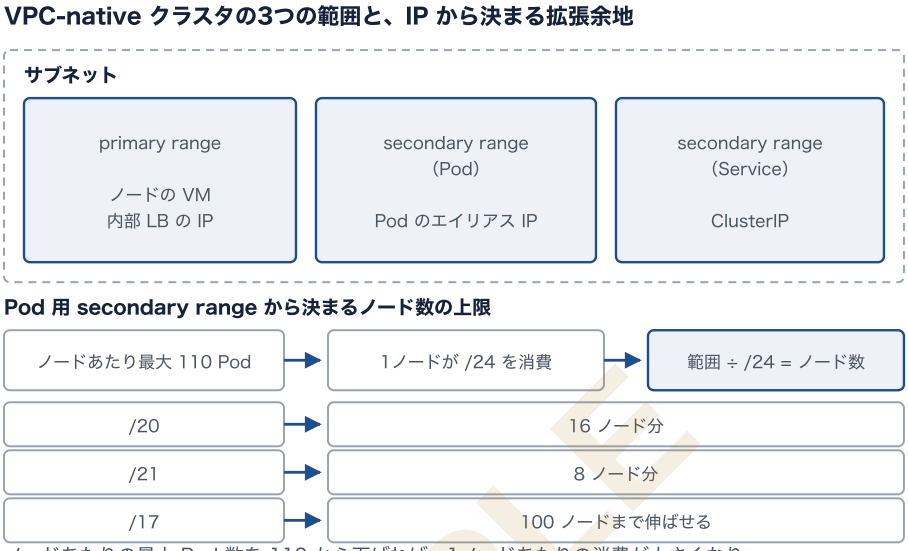


図 1-4　VPC-native クラスタの3つの範囲と、IP から決まる拡張余地

access to GKE control plane（コントロールプレーンへのアクセス）も設計事項です。public endpoint（インターネット経由・**authorized networks** で送信元を制限）、private endpoint（VPC 内部の IP のみ）、**DNS-based endpoint**（IAM で認可し DNS 名で到達）の3方式があり、オンプレから private endpoint に到達させるには Cloud Interconnect / HA VPN 側で経路と、コントロールプレーン側のカスタムルート広報の設定が要ります。

1-1-6 ネットワーク設計に適した IAM ロールの特定 [1.1]

Identity and Access Management (IAM) のロール選択は、ネットワーク設計と一体で問われます。最小権限で回すために覚えるべき組み合わせは次のとおりです。

ロール	付ける場所	できること
`roles/compute.networkAdmin`	プロジェクト	ネットワーク・サブネット・ルート・**load balancer provisioning**（転送ルール、URL マップ、バックエンドサービス）。ただし**ファイアウォールルールは触れない**、インスタンスも作れない
`roles/compute.securityAdmin`	プロジェクト	ファイアウォールルールと SSL 証明書
`roles/compute.networkUser`	**Shared VPC のホストプロジェクト、または個別サブネット**	サービスプロジェクトからそのサブネットを使ってリソースを作る
`roles/compute.xpnAdmin`	**組織またはフォルダ**	Shared VPC ホストプロジェクトの有効化とサービスプロジェクトの紐づけ
`roles/compute.loadBalancerAdmin`	プロジェクト	ロードバランサ関連リソースの管理
`roles/dns.admin`	プロジェクト	Cloud DNS のゾーンとレコード

Shared VPC subnet permissions の要点は、`roles/compute.networkUser` **はサブネット単位で付けられる**ことです。ホストプロジェクト全体に付けると全サブネットが使えてしまうため、「開発チームには dev サブネットだけ」という要件は、**該当サブネットのリソースに対して networkUser を付与**して満たします。

よく出る取り違え：「ネットワーク管理者にファイアウォールも任せたい」→ networkAdmin だけでは足りず securityAdmin が要ります。逆に「監査担当

に構成を見せたいが変更させたくない」→ `roles/compute.networkViewer` です。**xpnAdmin は組織レベル**であり、プロジェクトに付けても Shared VPC の有効化はできません。

1-1-7 マネージドサービスへの接続の計画 [1.1]

マネージドサービス（Cloud SQL、Memorystore、Vertex AI など）にプライベート接続する方法は3つあり、それぞれ性質が違います。

方式	仕組み	特徴と注意点
private services access（プライベートサービスアクセス）	自分の VPC に確保した予約範囲（allocated range）を Google 側の producer VPC に渡し、**VPC Network Peering** で接続	予約範囲を消費する。ピアリングなので**推移的でない**（オンプレから使うには custom route の export/import が要る）
Private Service Connect (PSC)	consumer 側のサブネットに**エンドポイント IP を1つ**作り、そこから producer のサービスへ	IP 消費が少ない。**推移性の問題を回避しやすい**。オンプレからも到達させやすい。NCC で **PSC propagation** も可能
Serverless VPC Access	Cloud Run / Cloud Functions / App Engine から VPC 内部 IP へ出ていくためのコネクタ（/28 のサブネットを消費）	方向は「サーバーレス → VPC」。逆方向ではない

設計判断：新規設計では PSC を第一候補にします。理由は、消費する IP が小さく、ピアリング特有の非推移性やルート交換の制約に悩まされにくいためです。private services access は、対応がその方式しかないサービスや既存構成で使います。

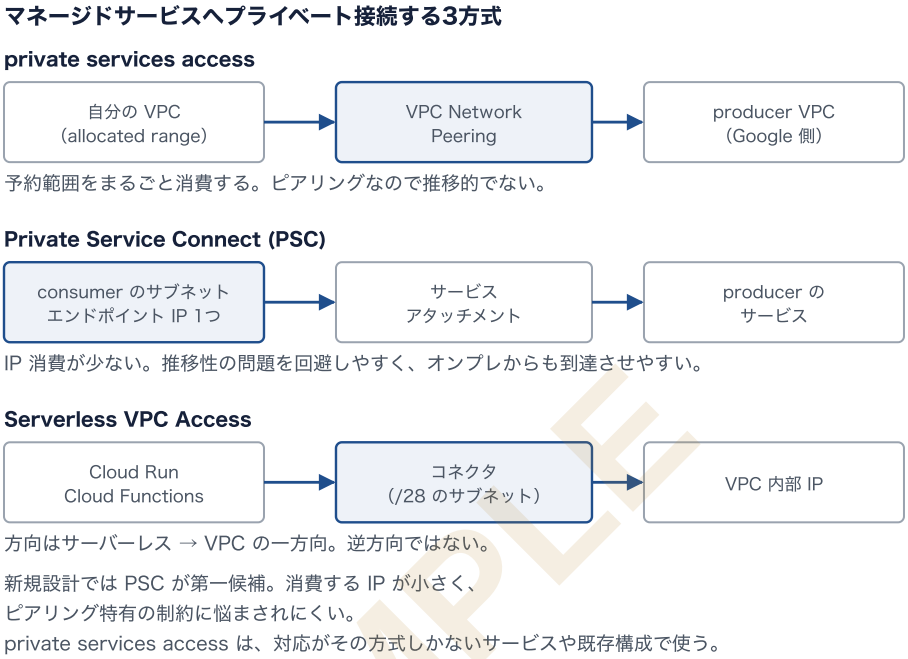


図 1-5　マネージドサービスへプライベート接続する3方式

Planning for quotas and limits（割り当てと上限の計画）も 1.1 の明示項目です。設計段階で確認すべき代表例は、VPC あたりのサブネット数・ピアリング数、プロジェクトあたりの転送ルール数、Cloud Router あたりの学習ルート数、ピアリンググループ全体で共有される**ルート数の上限**です。特に **VPC Network Peering はピアリンググループ全体でルート上限を共有**するため、ピアを増やしていくと突然どこかで上限に当たります。これが Network Connectivity Center や PSC へ寄せる実務上の動機になります。

1-1-8 設問パターンと切り方 [1.1]

場面1：小売企業が、日本・北米・欧州の顧客に単一ドメインで Web を配信します。要件は「最も低いレイテンシ」「静的アセットのキャッシュ」「WAF」。

- 正解の構成：**Premium Tier ＋ global external Application Load Balancer ＋ Cloud CDN ＋ Google Cloud Armor**。
- 切り方：「リージョンごとに regional external Application Load Balancer を置き Cloud DNS の geolocation で振る」は動きますが、**anycast 1 IP に劣ります**（DNS キャッシュのせいで障害時の切り替えが遅い、運用対象が3倍）。「Standard Tier でコストを下げる」は **global LB と Cloud CDN が使えない**ので即消えます。

場面2：社内の分析チームが 200 ノードまで伸びる GKE クラスタを立てたい。既存サブネットの Pod 用 secondary range は /20 です。

- 判定：ノードあたり 110 Pod（/24 消費）なら /20 は 16 ノード分。**要件を満たしません**。
- 正解：Pod 用 secondary range を /16 相当で設計し直す、または `--max-pods-per-node` を下げる、あるいは **non-RFC 1918（100.64.0.0/10）** を Pod 範囲に充てます。「サブネットの primary range を広げる」はノード用の話で、Pod 枯渇には効きません。

場面3：中央のネットワークチームが全社の VPC を管理し、開発チームには開発用サブネットだけ使わせたい。

- 正解：**Shared VPC ＋ 個別サブネット共有 ＋ 対象サブネットへの `roles/compute.networkUser`**。
- 切り方：「ホストプロジェクト全体に networkUser」は全サブネットが開くので分離になりません。「`roles/compute.networkAdmin` を開発チームに」は権限過多で、ファイアウォールは触れないのに VPC 構成は壊せます。

場面4：Cloud SQL にプライベート接続したいが、VPC の RFC 1918 の残りが少ない。

- 正解：**Private Service Connect (PSC)**。エンドポイント IP を1つ消費するだけです。
- 切り方：**private services access** は allocated range（通常 /16〜/24）をまるごと確保するため、IP が逼迫している状況では不利です。この「IP 消費量」がこの2択の決め手になります。

設問を読むときの着眼点：「グローバル／リージョナル」「暗号化の要否」「クライアント IP の保持」「IP の余裕」「誰が管理するか」の5語のどれが要件文に出ているかを最初に拾うと、選択肢の半分は機械的に消えます。

1-2 VPC ネットワークの設計 [1.2]

1-2-1 VPC の種類と数の決定 —— standalone か Shared VPC か [1.2]

Choosing the VPC type and quantity（VPC の種類と数の選択） は、組織構造をネットワークに写す作業です。

型	形	向く組織
standalone VPC（単独 VPC）	プロジェクトごとに VPC を持つ	小規模、チームが完全に独立、相互接続が要らない
Shared VPC（共有 VPC）	host project が VPC を持ち、service project がそのサブネットにリソースを作る	ネットワークを中央チームが統制し、アプリチームは自分のプロジェクトで開発する。**大企業の標準形**

Shared VPC の本質は「**ネットワークの管理権限とリソースの課金・IAM を分離できる**」ことです。中央のネットワークチームがサブネットとファイアウォール

を一元管理し、各サービスプロジェクトには使ってよいサブネットだけを `roles/compute.networkUser` で開放します。

the number of VPC environments（VPC 環境の数） は、分離要件から決めます。本番・ステージング・開発を分けるのが基本形で、さらに規制要件があれば別 VPC・別ホストプロジェクトに割ります。**VPC を分けすぎると相互接続の複雑さが爆発する**ため、「分離が要る理由を説明できる単位」でだけ分けます。

1-2-2 ネットワーク相互接続方式の決定 [1.2]

VPC 同士をつなぐ選択肢は3つあり、決定的な差は**推移性（transitivity）**と**規模**です。

方式	推移性	スケール	適用場面
VPC Network Peering	なし（A-B、B-C があっても A-C は通らない）	ピア数・ルート数の上限に当たりやすい	少数の VPC を1対1でつなぐ
Network Connectivity Center	ハブ経由で**あり**（**mesh topology** なら any-to-any、**star topology** ならスポーク間はハブ経由）	多数の VPC を集約できる	全社規模のマルチ VPC、ハイブリッドとの統合
Private Service Connect (PSC)	サービス単位の到達（ネットワーク全体はつながない）	エンドポイント単位	特定サービスだけを見せたい

設計判断：3〜4 VPC 程度で構成が固定なら VPC Network Peering で十分です。VPC が10を超える、または今後増える、あるいはオンプレとの接続を集

約したい —— ここは Network Connectivity Center に寄せます。「特定の API だけを他部門に開放」は PSC です。

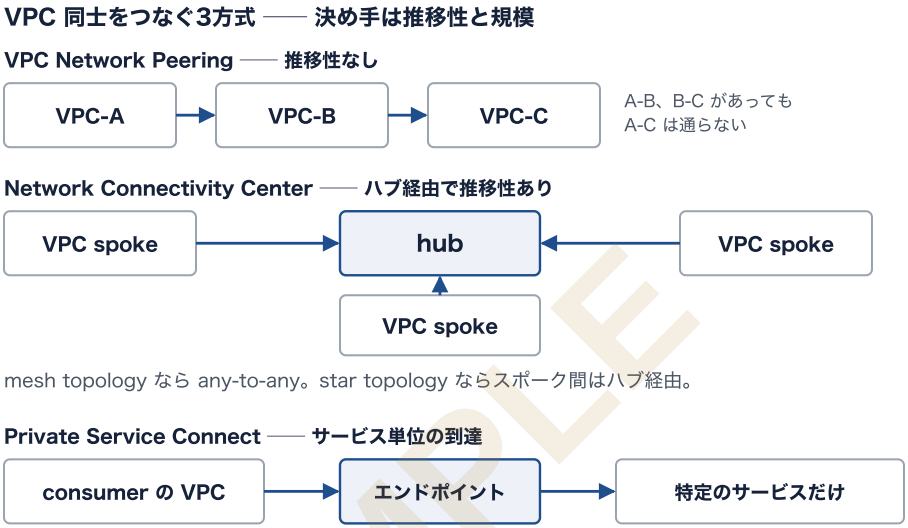


図 1-6　VPC 同士をつなぐ3方式 —— 決め手は推移性と規模

mesh topology と star topology の使い分け：mesh は全スポークが互いに通信できる形で、社内のどのシステムからどのシステムへも通す前提のときに使います。star（hub and spoke）は**スポーク同士を直接つながせない**形で、中央で検査・制御したい、あるいはテナント同士を隔離したいときに使います。

1-2-3 IP アドレス管理（IPAM）戦略の計画 [1.2]

IP address management (IPAM) の設計は、後から直せない領域です。次の要素を最初に決めます。

要素	決めること
subnets	リージョンごとの範囲。VPC のサブネットはリージョナルリソース
IPv6	internal IPv6（ULA・VPC 内のみ）か external IPv6（公開ルーティング可）か。**auto mode VPC は IPv6 非対応**で、custom mode への変換が必要
bring your own IP（BYOIP）	自社保有のパブリック範囲を Google Cloud に持ち込む
privately used public IP (PUPI)	パブリックとして割り当て済みの範囲を、自分の VPC 内で私的に使う。RFC 1918 が枯渇した大企業でよく使う
Private NAT	**重複した内部範囲同士**を通信させるための NAT。Network Connectivity Center のスポーク間や、ハイブリッドの範囲衝突の解消に使う
non-RFC 1918 addresses	100.64.0.0/10（CGNAT）や 192.0.0.0/24 など。GKE の Pod 範囲を逃がす先としても定番
managed services	private services access の allocated range を先に確保しておく
IPAM automation techniques	Terraform や **internal ranges** リソースで範囲の予約・重複検知を機械化する

PUPI の落とし穴：PUPI を使う VPC の VM は、**同じパブリック IP を実際に使っている外部リソースへ到達できなくなります**。たとえば 1.2.3.0/24 を PUPI として使うと、インターネット上の本物の 1.2.3.0/24 には行けません。「PUPI 導入後、特定の外部サービスにだけ繋がらない」という障害はこれです。またその範囲を外部へ広報してはいけません。

1-2-4 グローバル/リージョナルなネットワーク環境の計画 [1.2]

VPC 自体はグローバルリソースで、サブネットがリージョナルです。ここで設計上の分岐になるのが **dynamic routing mode** です。

モード	挙動	使いどころ
regional dynamic routing	Cloud Router は**自リージョンのサブネットだけ**をオンプレへ広報し、学習したルートも**自リージョン内にだけ**適用	リージョンを厳密に分離したい。障害の波及を抑えたい
global dynamic routing	全リージョンのサブネットを広報し、学習ルートを全リージョンへ適用	オンプレから全リージョンへ1本で到達させたい。リージョン障害時に別リージョンの Interconnect へ回り込ませたい

設計判断：災害復旧のためにリージョン間フェイルオーバーを効かせたいなら global です。ただし global にすると、あるリージョンの経路障害が他リージョンの経路選択にも影響しうるため、**障害ドメインを広げるトレードオフ**を受け入れることになります。この「可用性と障害波及のトレードオフ」は Professional 級で頻出の論点です。

1-2-5 ワークロードに合った MTU サイズの決定 [1.2]

maximum transmission unit (MTU) は VPC ネットワーク単位の設定です。取りうる値は 1460（既定）、1500、8896（jumbo frames）などで、範囲としては 1300〜8896 を設定できます。

判断基準は次のとおりです。

- **8896（ジャンボフレーム）**：同一 VPC 内の高スループット処理（HPC、大規模データ転送、ストレージ）に有効。スループットが上がり CPU 負荷が下がります。
- **1460**：既定。ハイブリッド接続や peering 相手の MTU が揃わない環境で無難。
- **注意**：MTU は**ネットワーク全体の設定**で、VM 側のゲスト OS の MTU も合わせる必要があります。合っていないとパケットが落ちるか、断片化で性能が落ちます。
- **peering した VPC 同士は MTU を一致させる**のが原則です。不一致だと大きなパケットが落ちます。
- ハイブリッド接続では経路上の最小値が効きます。Cloud Interconnect は 1440/1460/8896 を選べ、HA VPN は暗号化ヘッダの分だけ小さくなります（詳細は 1-3-10）。

1-2-6 サードパーティ機器の挿入の計画 [1.2]

third-party device insertion（サードパーティ機器の挿入）とは、**network virtual appliance (NVA)** —— 他社製のファイアウォールや IDS/IPS の仮想アプライアンス —— を通信経路上に置く設計です。

実装手段は3つあり、組み合わせて使います。

手段	内容	特徴
custom routes（static）	宛先範囲に対する next hop を NVA の VM または内部 LB に向けた静的ルート	単純。宛先ベースでしか振り分けられない
policy-based routes	**送信元 IP・宛先 IP・プロトコル・ポート**の組み合わせで next hop を決める	送信元ベースの振り分けができる。宛先だけでは足りない要件に
load balancing	**internal passthrough Network Load Balancer as a next hop**	NVA を複数台にして冗長化できる。1台が落ちてもヘルスチェックで外れる

設計の定石：NVA は単体では単一障害点になります。**複数の NVA を internal passthrough Network Load Balancer の背後に置き、その LB を next hop にする**のが標準構成です。さらに multi-NIC の VM を使って VPC をまたいだ検査を行う構成もあります（詳細は第6章 6.4）。

誤答肢の切り方：「NVA を冗長化したいので external Application Load Balancer を next hop にする」は誤りです。next hop に指定できるのは **internal passthrough Network Load Balancer** だけです。

1-2-7 設問パターンと切り方 [1.2]

場面1：買収により、既存の 12 個の VPC に加えて相手企業の 6 個の VPC を統合します。今後さらに増える見込みです。

- 正解：**Network Connectivity Center の VPC spoke（mesh topology）**。
- 切り方：「全 VPC を VPC Network Peering でフルメッシュ」は 18 VPC で 153 本のピアリングになり、**ピアリング数とルート数の上限**に当たります。

管理も破綻します。「1つの巨大な Shared VPC に統合」は移行コストが大きく、組織の分離要件とも衝突します。

場面2：HPC のワークロードが同一 VPC 内で大量のデータをやり取りしており、スループットを上げたい。

- 正解：**VPC の MTU を 8896（jumbo frames）に設定し、ゲスト OS の MTU も合わせる**。
- 切り方：「Premium Tier にする」は**内部通信には効きません**（Tier はインターネットとの往来の話）。「global dynamic routing にする」はハイブリッドの経路の話で無関係です。

場面3：セキュリティ部門が、開発サブネットからインターネットへ出る通信だけをサードパーティのファイアウォールアプライアンスに通したい、と要求しています。

- 正解：**NVA を複数台で internal passthrough Network Load Balancer の背後に置き、policy-based routes で送信元が開発サブネットの通信だけを next hop に向ける**。
- 切り方：「`0.0.0.0/0` の静的ルートを NVA へ」は**全 VM が通ってしまいます**（ネットワークタグで絞れば近づきますが、タグは VM 側の属性で「サブネット単位」の要件とは一致しません）。「external Application Load Balancer を next hop に」は**そもそも next hop に指定できません**。

場面4：RFC 1918 が枯渇し、社内で 10.0.0.0/8 の空きがありません。

- 選択肢の整理：**PUPI**（自社が使っていないパブリック範囲を私的に使う）、**non-RFC 1918**（100.64.0.0/10 など）、**IPv6**、**Private NAT**（重複を許して NAT で解消）。

- 決め方：外部との重複による到達不能を許容できるか（PUPI の副作用）、相手も同じ範囲を使っていないか、GKE の Pod 範囲のように「外に出さない範囲」に限定できるか、で選びます。**PUPI を Pod 範囲に充てる**のが副作用が小さい定番です。

1-3 回復性と性能に優れたハイブリッド・マルチクラウドネットワークの設計 [1.3]

1-3-1 ハイブリッド接続の設計 —— 帯域とセキュリティの制約 [1.3]

オンプレミス（および branch office）とクラウドをつなぐ選択肢を、帯域・SLA・費用・構築期間で並べます。

方式	帯域	経路	構築期間	適用場面
Cloud VPN（HA VPN）	トンネルあたり最大 3 Gbps 程度	公衆インターネット（IPsec 暗号化）	数時間	小〜中規模、暗号化必須、すぐ繋ぎたい
Partner Interconnect	50 Mbps 〜 50 Gbps を段階的に選択	サービスプロバイダ経由でプライベート	数日〜数週間	Google の PoP が近くにない、細かい帯域刻みが欲しい
Dedicated Interconnect	10 Gbps または 100 Gbps の回線を本数分	Google と直結（コロケーション施設）	数週間〜数か月	大容量・定常的な大量転送・最低レイテンシ
SD-WAN appliances	機器依存	インターネット＋オーバーレイ	短い	多数の branch office を一括で束ねる

設計判断の型：

- 「暗号化が必須」→ Cloud VPN、または **MACsec for Cloud Interconnect**／**HA VPN over Cloud Interconnect**（1-3-11）。
- 「帯域が 10 Gbps 以上必要」→ Dedicated Interconnect。VPN では届きません。
- 「拠点が数十あって個別に回線は引けない」→ SD-WAN appliances を Google Cloud 内に置き、**router appliance** として Network Connectivity Center のスポークにする。
- 「Google の PoP に自社設備が置けない」→ Partner Interconnect。

bandwidth and security constraints（帯域とセキュリティの制約）は必ず両方を読む、が鉄則です。「高帯域だが暗号化必須」なら Dedicated Interconnect ＋ MACsec、あるいは Interconnect の上に HA VPN を重ねる構成になります。

1-3-2 マルチクラウド接続の設計 [1.3]

他クラウド（AWS、Azure、Oracle Cloud など）との接続 —— multicloud connectivity（マルチクラウド接続）—— には2つの選択肢があります。

方式	内容	判断基準
Cloud VPN（HA VPN）	相手クラウドの VPN ゲートウェイと IPsec で結ぶ	小帯域・低コスト・即日。トラフィックはインターネット経由
Cross-Cloud Interconnect	Google が相手クラウドまでの専用線を提供（10 Gbps / 100 Gbps）	大容量・専用線品質・SLA が要る。相手クラウドの対応リージョンが必要

Cross-Cloud Interconnect の要点：Google 側が回線を手配するため、自社でコロケーションを持つ必要がありません。冗長化は Dedicated

Interconnect と同じ考え方で、**99.9%** は1メトロ内で2本、**99.99%** は2メトロ・4本という構成の要件を満たすことで得られます。

1-3-3 Direct Peering と Verified Peering Provider の選択 [1.3]

これらは **Cloud Interconnect とは別枠**の、Google の公開サービスへ到達するためのピアリングです。VPC の内部 IP には到達しません。ここを取り違えると設問を落とします。

方式	内容	選ぶ場面
Direct Peering	自社 AS を Google の PoP で直接 Google のエッジと BGP ピアリングする	自社で AS と PI アドレスを持ち、PoP に設備を置ける。Google の公開 API・Workspace へ大量アクセスする
Verified Peering Provider	Google が検証したプロバイダ経由でピアリング相当の品質を得る	自社で設備・AS を持てないが、インターネット経由より安定した経路で Google サービスへ到達したい

よく出る取り違え：「VPC のリソースにプライベート接続したい」→ Direct Peering では**できません**。Cloud Interconnect か Cloud VPN です。Direct Peering / Verified Peering Provider は Google のパブリックサービス向けであり、SLA も Cloud Interconnect のものは適用されません。

1-3-4 複数リージョンの高可用性・災害復旧接続戦略 [1.3]

複数リージョンにまたがるハイブリッド接続の high-availability and disaster recovery connectivity strategies（高可用性・災害復旧の接続戦略）は、**接続の冗長**と**routing mode** の2層で設計します。

1. **接続の冗長**：異なるメトロの2つのコロケーション施設に Interconnect を引く。同一施設内の2本は機器障害には効きますが、施設全損には効きません。
2. **routing mode**：**global dynamic routing** にすると、リージョン A の Interconnect が落ちたとき、オンプレからリージョン A のサブネットへの経路がリージョン B の Interconnect 経由に切り替わります。**regional dynamic routing** ではこの回り込みが起きません。
3. **経路の優先度**：平常時の経路は BGP の **MED（route priority）** で制御します。優先させたい経路に小さい MED を広報します。

設計判断：「リージョン障害時にオンプレからのアクセスを別リージョンへ自動フェイルオーバーさせたい」という要件には、**global dynamic routing ＋ 複数リージョンの VLAN attachment ＋ MED による優先度設定**の3点セットで答えます。どれか1つでも欠けると自動切り替えになりません。

1-3-5 オンプレミスから複数 VPC へのアクセス [1.3]

オンプレから複数の VPC に到達させる方法は3つあり、それぞれ制約が違います。

方式	仕組み	制約
Shared VPC	1つの VPC に全サービスプロジェクトのリソースを収容し、その VPC にだけ Interconnect を引く	最もシンプル。VPC を分けたい要件とは両立しない
multi-VPC peering	接続用 VPC に Interconnect を引き、各 VPC と peering。**custom route の export/import** でオンプレのルートを渡す	**peering は推移的でない**。VPC 同士は直接通信できない。ルート数上限に注意

方式	仕組み	制約
Network Connectivity Center topologies	ハブに hybrid spoke（VLAN attachment / HA VPN）と VPC spoke を接続	推移性がある。多数 VPC でも管理が破綻しない。**推奨される現行の答え**

誤答肢の切り方：「Interconnect を持つ VPC-A と peering している VPC-B から、さらに peering した VPC-C にオンプレのトラフィックを流したい」→ **peering は推移的でないので不可**。ここで正解は NCC への移行か、VPC-C にも直接 peering して custom route を export することです。

1-3-6 オンプレミスからの Google サービス・API へのプライベートアクセス [1.3]

オンプレから **Vertex AI** や各種 **application programming interfaces (APIs)** にインターネットを経由せず到達したい、という要件は頻出です。

方式	到達先の IP	特徴
Private Google Access for on-premises hosts	`private.googleapis.com`（199.36.153.8/30）または `restricted.googleapis.com`（199.36.153.4/30）	Interconnect / VPN 経由。オンプレ DNS で `*.googleapis.com` をこの範囲へ解決させ、VPC 側でルートを広報する
Private Service Connect endpoint for Google APIs	自分で決めた VPC 内の IP	自社の IP 体系に合わせられる。オンプレからはその IP へ到達させるだけ。**きめ細かい制御がしやすい**

restricted と private の違い：`restricted.googleapis.com` は **VPC Service Controls** に対応した API だけに到達を限定します。データ持ち出しを防ぎた

い規制環境ではこちらです。`private.googleapis.com` は VPC Service Controls 非対応の API も含めて到達できます。

設定の3点セット：(1) オンプレの DNS で `*.googleapis.com` を該当範囲に解決させる、(2) Cloud Router の **custom advertised route** でその範囲をオンプレへ広報する、(3) オンプレ側でその範囲へのルートを VPN/Interconnect に向ける。どれかが欠けていると到達しません。トラブルシューティングの設問はこの3点のどれが欠けているかを問います。

1-3-7 PSC と VPC Network Peering によるマネージドサービスへのアクセス [1.3]

1-1-7 の内容をハイブリッド視点で見ると、選択の基準が変わります。

- **private services access（VPC Network Peering ベース）** は、オンプレからそのまま到達できません。**Cloud Router の custom advertised route** で allocated range をオンプレへ広報し、かつ producer 側 peering で **export custom routes** を有効にする必要があります。設定が二段構えで、抜けやすい。
- **PSC** はエンドポイントが自分の VPC のサブネット内にある通常の内部 IP なので、オンプレからは「VPC 内の IP に到達する」だけで済みます。**ハイブリッド環境では PSC のほうが素直**です。

1-3-8 オンプレミスとクラウドをまたぐ IP アドレス空間の設計 [1.3]

planning to avoid overlaps（重複の回避） が最優先事項です。オンプレとクラウドで同じ範囲を使っていると BGP でルートを交換できません。

手段	内容
internal ranges	Google Cloud のリソースとして IP 範囲を「予約」し、重複を機械的に検知・防止する。IPAM automation の中核
範囲の分割方針	オンプレに 10.0.0.0/8 の一部、クラウドに別の一部、というように**最初に大きく分けて割り当てる**。リージョンごとにさらに分ける
Private NAT	どうしても重複が解消できない場合、NAT で変換して通す。買収した企業の網を統合するときの定番
non-RFC 1918 addresses	RFC 1918 が枯渇したら 100.64.0.0/10 などを使う。GKE の Pod 範囲を逃がす先としても有効

設計判断:「合併で相手企業と 10.10.0.0/16 が重複している。すぐに再アドレッシングはできない」→ **Private NAT** で答えます。「まだ設計段階で将来の重複を防ぎたい」→ **internal ranges で予約する**が答えです。

1-3-9 ハイブリッド DNS トポロジのアーキテクチャ [1.3]

1-1-3 の各機能を、実際のトポロジとして組み立てます。

構成要素	役割
forwarding paths（転送パス）	Cloud DNS の forwarding zone で「このドメインはオンプレの DNS へ」を定義。転送先はオンプレの DNS サーバー IP
inbound policies（インバウンドポリシー）	VPC 内にインバウンドフォワーディング用の IP を作る。オンプレの DNS サーバーはこの IP を条件付きフォワーダに設定する

構成要素	役割
cross-project binding（クロスプロジェクトバインディング）	ある プロジェクトの private zone を、別プロジェクトの VPC に紐づける。Shared VPC 構成で DNS を集中管理するときに使う
DNS peering strategy（DNS ピアリング戦略）	VPC-A の名前解決を VPC-B の設定に委譲する。ハブ VPC に DNS 設定を集約し、各スポーク VPC からハブへ DNS peering する形が定番

推奨トポロジ：ハブ VPC に forwarding zone と inbound policy を集約し、各スポーク VPC からハブへ **DNS peering** します。こうすると、オンプレの DNS 設定変更が1か所で済み、スポークが増えても設定が増えません。

よく出る取り違え：DNS peering の向きは「委譲する側 → 委譲される側」です。スポークがハブの設定を使いたいなら、スポーク側に「ハブへの DNS peering zone」を作ります。逆向きに作ると解決しません。

1-3-10 ハイブリッド接続の MTU サイズの決定 [1.3]

ハイブリッドの MTU は、経路上の最も小さい値が実効値になります。

接続	MTU の考え方
Cloud Interconnect	VLAN attachment ごとに 1440 / 1460 / 1500 / 8896 を選べる。VPC 側の MTU と揃える。オンプレ側の機器も同じ値に設定する
HA VPN	IPsec のオーバーヘッドがあるため、実効ペイロードは小さくなる。VM 側の MTU は 1460 より下げる（1370 前後）ことが推奨される場面がある

症状からの逆引き:「小さいパケットは通るが大きいファイル転送だけ止まる」「TLS ハンドシェイクは通るが本文で固まる」—— これは **MTU の不一致か PMTUD（Path MTU Discovery）の失敗**です。経路上で ICMP の "fragmentation needed" がファイアウォールに落とされていないかを確認し、MTU を揃えるか VM 側を下げます。この症状パターンは第5章のトラブルシューティングでも再登場します。

1-3-11 Interconnect の暗号化オプション [1.3]

Cloud Interconnect は専用線ですが、既定では**暗号化されていません**。規制で暗号化が必須の場合は次の2択です。

方式	暗号化の層	特徴
MACsec for Cloud Interconnect	L2（回線区間）	Dedicated Interconnect の物理リンク上で暗号化。スループットの劣化が小さい。**回線区間だけ**を守る
HA VPN over Cloud Interconnect	L3（IPsec）	Interconnect の上に HA VPN トンネルを張る。エンドツーエンドに近い暗号化。トンネルの帯域上限がボトルネックになりうる

設計判断:「専用線の帯域を活かしつつ、回線業者に対する盗聴を防ぎたい」→ MACsec。「暗号化を Google Cloud のゲートウェイまで通したい／Partner Interconnect でも暗号化したい」→ HA VPN over Cloud Interconnect。MACsec は Dedicated Interconnect が前提であり、Partner Interconnect では使えない点が判断の分かれ目です。

1-3-12 設問パターンと切り方 [1.3]

場面1：金融機関が、東京と大阪の2データセンターから Google Cloud へ接続します。要件は「99.99% の SLA」「暗号化必須」「20 Gbps」。

- 正解：**2つのメトロ（東京・大阪）にそれぞれ2本の Dedicated Interconnect（計4本）＋ global dynamic routing ＋ MACsec**。
- 切り方：「HA VPN 2トンネル」は 99.99% の SLA は付きますが **20 Gbps に届きません**。「同一メトロに4本」はメトロ障害に耐えられず 99.9% 止まりです。暗号化要件で MACsec か HA VPN over Cloud Interconnect のどちらかが必ず入ります。

場面2：オンプレのバッチが Vertex AI の API を呼びますが、社内規程でインターネット経由が禁止されています。

- 正解：オンプレ DNS で `*.googleapis.com` を **restricted.googleapis.com（199.36.153.4/30）** に解決させ、Cloud Router の **custom advertised route** でその範囲をオンプレへ広報し、オンプレ側でその範囲を Interconnect へ向けます。
- 切り方：「Direct Peering を使う」は**インターネット経由ではないものの、VPC のプライベート接続とは別物**で、VPC Service Controls とも噛み合いません。「Cloud NAT を使う」は方向が逆（クラウドから外へ）です。

場面3：合併先の網が 10.10.0.0/16 で自社と完全に重複しており、3か月以内に接続が必要です。再アドレッシングは1年かかります。

- 正解：**Private NAT** で変換して通します。
- 切り方：「VPC Network Peering」は**範囲が重複していると張れません**。「HA VPN で繋ぐ」は張れますが、重複した範囲へのルーティングが成立しません。期限の制約があるので「再アドレッシング」も選べません。

場面4：スポーク VPC が 15 個あり、全てからオンプレの DNS を引きたい。オンプレの DNS サーバー IP は将来変わる可能性があります。

- 正解：**ハブ VPC に forwarding zone を1つ置き、各スポークからハブへ DNS peering**。
- 切り方：「各 VPC に forwarding zone を作る」は動きますが、**IP 変更時に 15 か所の変更**が必要で、要件の意図（変更の局所化）に反します。

場面5：Interconnect 経由で大きなファイル転送だけが失敗し、小さなリクエストは通ります。

- 正解：**MTU の不一致**。VLAN attachment・VPC・オンプレ機器・ゲスト OS の MTU を揃え、経路上で ICMP の "fragmentation needed" がブロックされていないかを確認します。
- 切り方：「帯域不足」なら大小に関係なく遅くなるだけで、特定サイズだけ落ちる説明になりません。

1-4 Google Kubernetes Engine (GKE) の設計 [1.4]

1-4-1 パブリックノードとプライベートノードの選択 [1.4]

public or private cluster nodes and node pools（パブリック/プライベートのクラスタノードとノードプール） の判断です。

種別	ノードの IP	外部への出方
public node	外部 IP を持つ	直接インターネットへ
private node	内部 IP のみ	**Cloud NAT** 経由でのみ外向き通信。インターネットからは到達不可

設計判断：本番は private node が既定の選択です。攻撃面が小さく、egress を Cloud NAT に集約できるため、送信元 IP を固定して外部サービスの許可リストに登録することもできます。代償として、コンテナイメージの取得や OS 更新のために **Cloud NAT または Private Google Access** の構成が必須になり、これを忘れるとノードが起動時に固まります。「private cluster を作ったらイメージの pull に失敗する」設問はこれです。

ノードプール単位でも public / private を選べるため、「大半は private、外部と直接通信するワーカーだけ public」という混在構成も可能です。

1-4-2 コントロールプレーンのエンドポイントの選択 [1.4]

public or private control plane endpoints（パブリック/プライベートのコントロールプレーンエンドポイント） はノードの公開設定とは独立です。

方式	到達経路	注意点
public endpoint	インターネットから API サーバーへ	**authorized networks** で送信元 CIDR を絞るのが必須
private endpoint	VPC 内の内部 IP から	オンプレから使うには Interconnect / VPN ＋ **コントロールプレーン範囲の経路広報**が必要。ピアリング越しの到達には custom route の export が要る
DNS-based endpoint	DNS 名で解決し、**IAM で認可**。IP ベースの許可リストが不要	経路設計が単純になる。VPC からもオンプレからも同じ名前で使える

設計判断：CI/CD が Google Cloud の外にあり IP が固定できない、という要件では DNS-based endpoint が有効です。IP ベースの authorized

networks では固定 IP が必要になるためです。逆に「クラスタ管理は社内ネットワークからのみ」という要件なら private endpoint です。

1-4-3 サブネットの計画 —— プライマリ範囲とセカンダリ範囲 [1.4]

Planning subnets: Primary and secondary ranges は 1-1-5 の内容を実装単位に落としたものです。

範囲	役割	サイズ決定
primary range	ノードの内部 IP、および内部 LB の IP	ノード上限＋LB 分
secondary range（Pod）	Pod の エイリアス IP	ノード数 × ノードあたり Pod 範囲
secondary range（Service）	ClusterIP	Service 上限

計算の例：ノードあたり最大 110 Pod（GKE は /24 を割り当て）で 100 ノードまで伸ばしたいなら、Pod 用 secondary range は 100 × /24 ＝ **/17** が必要です。ここを /20（16ノード分）で作ってしまうと、後から拡張が要ります。

なお、既存クラスタに **additional Pod ranges**（追加の Pod 範囲）を後から足すことは可能で、これが枯渇時の主な救済策になります（実装は第2章 2.4）。

1-4-4 GKE の IP アドレスの計画 [1.4]

GKE で消費できるアドレスの種類を整理します。

種類	用途
RFC 1918	標準。10.x / 172.16-31.x / 192.168.x

種類	用途
non-RFC 1918	100.64.0.0/10 など。Pod 範囲を逃がして RFC 1918 を温存する定番手法
Google-managed services range	private services access の allocated range。クラスタから Cloud SQL 等へ繋ぐときに消費
PSC	コントロールプレーン接続や producer サービスへの接続に使うエンドポイント IP
shared IP ranges	Shared VPC で、ホストプロジェクトのサブネットとセカンダリ範囲をサービスプロジェクトのクラスタが使う
PUPI	自分の VPC 内で私的に使うパブリック範囲。**Pod 範囲を PUPI にすると、peering 越しにその範囲が広報されない**（既定では export されない）ため、他 VPC との重複を許容しつつ隔離できる

設計の勘所：PUPI を Pod 範囲に使う手法は、「多数のクラスタで Pod 範囲が重複するが、ノード同士は通信させたい」という大規模環境で有効です。Pod 範囲だけが peering の外に置かれるためです。

1-4-5 IPv6 の計画 [1.4]

GKE のデュアルスタックは、**VPC-native クラスタ**かつ **dual-stack サブネット**が前提です。要点は次のとおりです。

- サブネットに internal IPv6 range（ULA）か external IPv6 range を割り当て、クラスタを dual-stack で作成します。
- **auto mode VPC は IPv6 に対応していない**ため、custom mode に変換します。

- Service に対しては `ipFamilies` で IPv4 / IPv6 / 両方を指定します。
- IPv6 は Pod 範囲の枯渇問題そのものへの根本対策になりますが、外部との相互運用（クライアント側の IPv6 対応）を確認してから採用します。

1-4-6 GKE ネットワーキングのロードバランシングの設計 [1.4]

GKE から Google Cloud のロードバランサへ繋ぐ道は3本あります。

方式	生成されるもの	特徴
Service type LoadBalancer	passthrough Network Load Balancer（internal / external）	L4。単純。`networking.gke.io/load-balancer-type: "Internal"` で内部化
GKE Ingress controller	external / internal Application Load Balancer	L7。既存資産が多い。単一クラスタが基本
GKE Gateway controller	Application Load Balancer（マルチクラスタ対応）	L7。Gateway API 準拠。**マルチクラスタ・複数チームでの権限分離**に強い、現行の推奨

network endpoint groups (NEGs) の理解が要点です。**container-native load balancing** では、LB のバックエンドが「ノード（インスタンスグループ）」ではなく **Pod の IP そのもの**（NEG）になります。これにより、ノード上での二重ホップ（kube-proxy による転送）がなくなり、レイテンシが下がり、ヘルスチェックが Pod に直接効き、クライアント IP の保持も自然になります。VPC-native クラスタが前提です。

設計判断：新規なら **GKE Gateway controller ＋ NEG** が基本形です。Ingress は既存構成の維持と単純な用途に。L4 で十分・UDP が要る場合の

み Service type LoadBalancer です。

1-4-7 ノードプール構成の追加と管理 [1.4]

Adding and managing node pool configuration（ノードプール構成の追加と管理） は、ネットワーク観点では次が問われます。

- ノードプールごとに **network tags** を付け、ファイアウォールルールや静的ルートの適用対象を分けられます。マイクロセグメンテーションの基礎です。
- ノードプールごとに public / private を選べます。
- ノードプールごとに別のサブネット（マルチサブネットのクラスタ）を割り当て、**primary range の枯渇を回避**できます。ノード用 IP が足りなくなったときの現実的な解です。
- ノードプールの追加・削除でノードが入れ替わるため、**Cloud NAT の同時接続ポート**や**ファイアウォールルールのタグ**が新ノードにも効くかを事前に確認します。

よく出る取り違え：「Pod IP が枯渇した」と「ノード IP が枯渇した」は対策が別です。前者は **additional Pod ranges の追加**、後者は**別サブネットのノードプール追加**（またはサブネットの primary range 拡張）で解きます。

1-4-8 設問パターンと切り方 [1.4]

場面1：金融系の GKE クラスタで、ノードをインターネットに出したくないが、コンテナイメージは Artifact Registry から取得したい。

- 正解：**private node ＋ Private Google Access**（Artifact Registry は Google API なのでこれで足ります）。汎用の外向き通信も要るなら **Cloud NAT** を追加します。

- 切り方：「public node にして firewall で塞ぐ」は攻撃面が残るため設計として劣ります。「Cloud NAT だけ有効化して Private Google Access は不要」も動きますが、**Google API 宛は NAT を経由させずに済む**分、Private Google Access のほうがコストと経路の点で望ましい判断です。

場面2：外部 SaaS の CI/CD からクラスタを操作したいが、送信元 IP が公開されておらず頻繁に変わります。

- 正解：**DNS-based endpoint ＋ IAM（`container.clusters.connect`）**。
- 切り方：「authorized networks に `0.0.0.0/0` を追加」は要件を満たしますが、**アクセス制御を捨てる**ので正解になりません。「private endpoint ＋ VPN」は外部 SaaS には適用しにくい構成です。

場面3：既存クラスタで Pod IP が枯渇し、ノードを追加できません。無停止で拡張が必要です。

- 正解：**サブネットに新しい secondary range を追加し、additional Pod range として登録して、新しいノードプールに割り当てる**。
- 切り方：「既存の secondary range を拡張」は**secondary range は拡張できません**。「Service 範囲を広げる」は別の範囲の話で、しかも Service 範囲は変更できません。「クラスタを作り直す」は無停止要件に反します。

場面4：GKE のバックエンドを ALB に繋いだが、ノードの CPU が高いわりにレイテンシが安定しません。

- 正解：**container-native load balancing（NEG）** を有効にして、LB のバックエンドを Pod 直結にします。
- 切り方：インスタンスグループをバックエンドにしていると、LB → ノード → kube-proxy → 別ノードの Pod という二重ホップが発生し、レイテンシと負荷の偏りを生みます。この症状の設問は NEG が答えです。

場面5：複数チームが1つのクラスタを共有し、チームごとにルーティング設定を管理させたい。

- 正解：**GKE Gateway controller**。インフラ担当が `Gateway`、アプリ担当が `HTTPRoute` を持つ形で権限を分離できます。
- 切り方：**Ingress** では1つのリソースに全設定が集まるため、権限分離ができません。

サンプルはここまでです

続き(第2章〜巻末)は、GCP PNE 問題集のご購入で、頻出度別ノート・暗記用ノートと合わせて3点すべてダウンロードできます。

GCP PNE 学習用テキスト(無料サンプル)

版

2026年7月版(Google Cloud 公式 Professional Cloud Network Engineer Exam Guide (2026-09-07 取得) 準拠)

発行

IT資格道場(https://www.shikaku-doujou.com)

お問い合わせ

info@shikaku-doujou.com