IT資格道場

GCP PSE 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 Security 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

はじめに —— この教材の使い方

第1章 Configuring access(ドメイン1・採点対象の 25%)

1-1 Cloud Identity の管理 [1.1]

1-2 サービスアカウントの管理 [1.2]

1-3 認証の管理 [1.3]

1-4 認可制御の管理と実装 [1.4]

1-5 リソース階層の定義 [1.5]

1-6 ドメイン1の設計判断まとめと誤答の切り方 [1.4]

サンプルはここまでです

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 Security 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 Security 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 —— 5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	Configuring access	**25%**	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	Securing communications and establishing boundary protection	**22%**	第2章
3	Ensuring data protection	**23%**	第3章
4	Managing operations	**19%**	第4章
5	Supporting compliance requirements	**11%**	第5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	何を切り分けるか	分かれ方
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 Configuring access(ドメイン1・採点対象の25%)

この章は「誰が(アイデンティティ)」「何に(リソース)」「どこまで(権限)」を設計する領域です。Professional Cloud Security Engineer の出題は、機能の名前を答えさせるのではなく、**要件を読んで最小権限を満たす構成を選ばせる**形で来ます。ですから本章では、各機能の定義に加えて「似た機能との使い分け」「なぜ他の選択肢が誤りか」を必ず添えます。

ドメイン1でつまずきやすいのは、次の4つの取り違えです。最初にここを頭に入れてから読み進めてください。

よくある取り違え	正しい理解
Cloud Identity と Google Cloud の IAM は同じもの	Cloud Identity は**アイデンティティ(誰か)の管理**、IAM は**認可(何ができるか)の管理**。別レイヤ
Workforce Identity Federation と Workload Identity Federation	前者は**人**、後者は**ワークロード(アプリ・CI/CD・他クラウド)** をフェデレートする
IAM の allow policy を消せばアクセスは止まる	上位階層(組織・フォルダ)から**継承**した allow policy は下位では消せない。止めるには **IAM deny policy** か組織のポリシー
サービスアカウントキーを作れば「認証できた」	キーは**最後の手段**。impersonation・Workload Identity Federation・アタッチが先

1-1 Cloud Identity の管理 [1.1]

1-1-1 Cloud Identity と組織リソースの関係 [1.1]

Cloud Identity は、Google Cloud で使うユーザーアカウントとグループを保持する**アイデンティティ プロバイダ兼ディレクトリ**です。Google Workspace を契約していない企業でも、Cloud Identity(Free または Premium)だけを取得すれば、自社ドメイン(example.com)のユーザーアカウントを作れます。

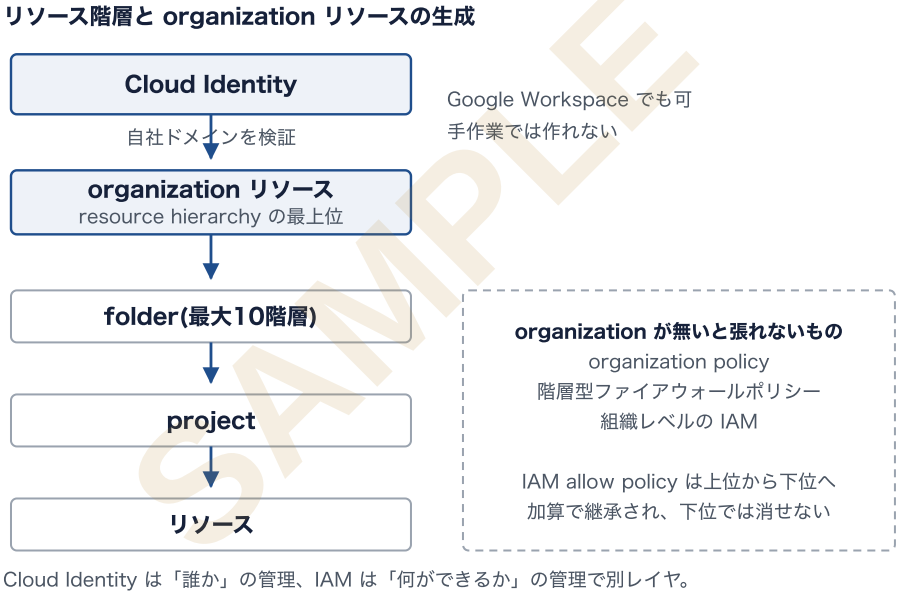


図 1-1　リソース階層と organization リソースの生成

重要なのは、Cloud Identity アカウントを作って自社ドメインを検証すると、**Google Cloud の organization リソースが自動的に作られる**という点です。organization は resource hierarchy の最上位で、これが無いと組織全体に効くポリシー(organization policy、階層型ファイアウォールポリシー、組織レベルの IAM)を張れません。試験では「全プロジェクトに一括で制約をかけ

たいが organization が無い」という場面が出ます。答えは「Cloud Identity(または Google Workspace)でドメインを検証し、organization リソースを作る」です。

用語	何を指すか	誤答肢との切り分け
Cloud Identity	ユーザー・グループ・デバイスを管理するディレクトリ	IAM ロールはここでは付けない
Google Workspace	Cloud Identity の機能に Gmail 等を足したもの	Google Cloud を使うだけなら Cloud Identity で十分
organization リソース	Google Cloud のリソース階層の最上位	ドメイン検証で自動生成。手作業では作れない
Admin Console(admin.google.com)	Cloud Identity の管理面。ユーザー作成・2SV・SSO 設定	IAM 権限の付与は Google Cloud コンソール側

消費者アカウント(gmail.com など)の扱いも頻出です。従業員が自分の個人 Gmail で Google Cloud を使っていると、退職時にアカウントを止められず、監査もできません。対策は次の順で考えます。

1. Cloud Identity で自社ドメインのマネージド アカウントに移行する
2. 自社ドメインで作られてしまった未管理アカウント(unmanaged accounts)は、**transfer tool for unmanaged users** で組織の管理下に取り込む
3. organization policy の `constraints/iam.allowedPolicyMemberDomains`(**Domain restricted sharing**)で、自社の Cloud Identity 顧客 ID(customer ID)以外のプリンシパルに IAM を付けられないようにする

3 の Domain restricted sharing は「外部の Gmail アカウントにうっかり編集権限を付けてしまう事故」を構造的に止める定番の答えです。似た選択肢として「IAM で個別に削除する」「監査ログで検知する」が並びますが、これらは**事後**であり、**予防**を問われた場合は誤りになります。

1-1-2 Google Cloud Directory Sync(GCDS)と third-party identity provider による single sign-on(SSO) [1.1]

企業には既に Active Directory(AD)や LDAP ディレクトリがあることがほとんどです。ここで使うのが **Google Cloud Directory Sync(GCDS)** です。

GCDS は、オンプレミスの LDAP/AD を**信頼できる情報源(source of truth)**とし、Cloud Identity 側のユーザー・グループ・組織部門(OU)をそれに合わせて**一方向(オンプレ → Cloud Identity)** に同期するツールです。押さえるべき性質は次のとおりです。

- **一方向**である。Cloud Identity 側で手作業で作ったユーザーは、同期のたびに削除対象になり得る
- オンプレ側のネットワーク内で動き、**アウトバウンドの HTTPS のみ**で Google に接続する。受信ポートを開ける必要はない
- **パスワードは既定では同期しない**。ハッシュを同期するオプションはあるが、SSO を使うなら不要
- **simulated sync(シミュレーション モード)** で、実際に反映する前に差分を確認できる。大量削除事故を防ぐため本番前に必ず実行する
- 削除の暴走を防ぐ **deletion limits(削除上限)** を設定できる

GCDS はプロビジョニング(誰が存在するか)、SSO は認証(本人確認をどこでやるか) です。この2つは別物で、両方必要になります。ここを混同させる選択肢が頻出です。

SSO は、third-party identity provider(Microsoft Entra ID、Okta、Active Directory Federation Services など)を **SAML 2.0** の IdP として設定し、Google をサービス プロバイダ(SP)にする形で構成します。Admin Console の「Security → Authentication → SSO with third-party IdP」で、Sign-in page URL、Sign-out page URL、**verification certificate(IdP の署名証明書)** を登録します。

論点	押さえどころ
SSO 設定時の super administrator	SSO 経由でログインできなくなる事故に備え、**super administrator は既定で SSO をバイパス**してパスワードでログインできる
SSO プロファイル	組織全体だけでなく、**組織部門(OU)やグループ単位**で別々の SSO プロファイルを割り当てられる。移行期の段階適用に使う
証明書の期限切れ	IdP の署名証明書が失効すると全ユーザーがログイン不能になる。ローテーション計画を持つ
SSO でも 2-step verification は必要か	認証は IdP 側で行われるため、多要素は IdP 側で強制する。Google 側の 2SV は SSO をバイパスするアカウント(super administrator など)に効かせる

自動プロビジョニング機能を持つ IdP(Entra ID、Okta 等)を使う場合は、GCDS の代わりに **IdP 側の自動プロビジョニング**を使う選択もあります。判断軸は「信頼できる情報源がオンプレの AD なら GCDS、クラウドの IdP なら IdP 側プロビジョニング」です。

1-1-3 super administrator account の管理 [1.1]

super administrator(特権管理者) は Cloud Identity/Workspace の最上位管理者で、全ユーザーのパスワードをリセットでき、他の管理者ロールを付与でき、組織の設定をすべて変えられます。Google Cloud 側の Organization Administrator ロールも自分に付けられるため、**実質的に組織全体を掌握できるアカウント**です。試験ではこの保護がほぼ必ず問われます。

実践	理由
日常業務に使わない	常用すると侵害時の被害が最大化する。日常は最小権限の管理者ロールで
複数名(推奨2〜3名)用意する	1名だとロックアウトで組織が凍る。多すぎると攻撃面が増える
ドメインのメールに紐づく専用アカウントにする	個人アカウントや共有アカウントにしない
security key(ハードウェア キー)による 2-step verification を必須にする	フィッシング耐性が最も高い。SMS は最も弱い
復旧用の電話番号・メールを最新に保つ	ロックアウト時の唯一の経路
super administrator は SSO をバイパスする点を利用する	IdP 障害時にも管理面に入れる「非常口」として残す
使用状況を監査する	Admin Console の監査ログと Cloud Audit Logs を Security Command Center・SIEM へ転送し、super admin の操作にアラートを張る

Organization Administrator ロールとの違いも出ます。super administrator は Cloud Identity 側の管理者、Organization Administrator は Google Cloud の IAM ロール(`roles/resourcemanager.organizationAdmin`)です。組織作成直後は super

administrator に Organization Administrator が付いていますが、運用に入ったら**両者を分離**し、super administrator からは Google Cloud の日常権限を外すのが定石です。これが **separation of duties(職務の分離)** の実装例になります。

1-1-4 user lifecycle management プロセスの自動化 [1.1]

入社・異動・退職(joiner / mover / leaver)の3局面を、手作業ゼロで回す設計が問われます。

- **入社**: 人事システム(HR system)を信頼できる情報源にし、AD → GCDS → Cloud Identity、または HR → IdP(SCIM)→ Cloud Identity の順で自動作成する
- **異動**: 部署変更を **グループのメンバーシップ**に反映させる。IAM は個人ではなくグループに付けているので、グループが変われば権限が自動で変わる
- **退職**: AD 側で無効化すると GCDS が Cloud Identity 側を suspend/削除する。**アカウントの停止(suspend)** は、そのユーザーのセッションと OAuth トークンを無効化する

退職時に見落とされがちなのが次の3点です。試験で「退職者がまだアクセスできる」というシナリオが出たら、ここを疑います。

1. そのユーザーが作った **service account key** が残っている
2. そのユーザーが個人アカウント(gmail.com)で招待されており、Cloud Identity の管理外にある
3. IAM がグループではなく**個人に直接**付いており、削除漏れになっている

自動化の実装手段は、Admin SDK Directory API、Cloud Identity API、Terraform(`google_cloud_identity_group` 等)、および IdP の SCIM プロビジョニングです。**「手動でチェックリストを回す」は Professional 級では常に誤答**と考えてください。

1-1-5 user account と group のプログラム的な管理 [1.1]

API での操作は、次の使い分けを押さえます。

API/ツール	用途	備考
Admin SDK Directory API	ユーザー・グループ・組織部門(OU)・デバイスの CRUD	Workspace/Cloud Identity 全体の管理 API
Cloud Identity API(`cloudidentity.googleapis.com`)	グループ・メンバーシップ、**dynamic groups**、デバイス	Google Cloud 側から扱いやすい。IAM でアクセス制御できる
gcloud identity groups	CLI からのグループ操作	スクリプト化に便利
Terraform	宣言的な管理、drift detection	変更が Git のレビュー対象になる

dynamic groups(動的グループ)は、`user.organizations.exists(org, org.department == "finance")` のようなクエリでメンバーシップが自動決定されるグループです。異動時にメンバー入替の手作業が不要になるため、user lifecycle の自動化とセットでよく出ます。

API 呼び出しの認証には、**domain-wide delegation(ドメイン全体の委任)**を付けたサービスアカウントを使い、管理者ユーザーを impersonate するのが従来の方法です。ただし domain-wide delegation は「そのサービスアカウントが全ユーザーになりすませる」極めて強い権限なので、**必要最小限のスコープだけを許可し、鍵を持たせず、使用を監査する**のが必須です。試験で

「domain-wide delegation を広く許可する」選択肢が並んだら、それはほぼ誤答です。

1-1-6 Workforce Identity Federation の構成 [1.1]

Workforce Identity Federation は、外部 IdP の**人間のユーザー**に、Cloud Identity アカウントを作らずに Google Cloud へアクセスさせる仕組みです。GCDS のような同期が不要になる点が最大の利点です。

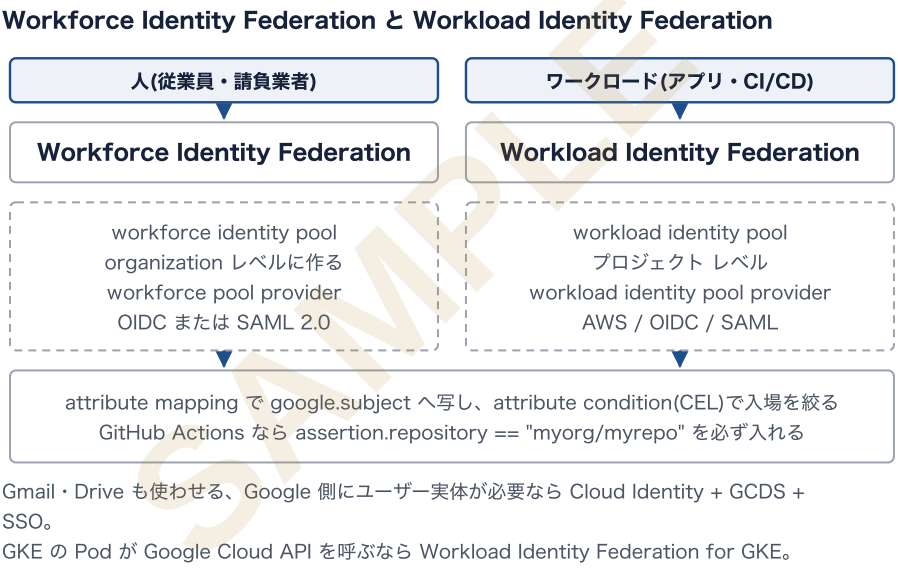


図 1-2　Workforce Identity Federation と Workload Identity Federation

構成要素は次のとおりです。

要素	役割
workforce identity pool	外部ユーザーを収める入れ物。 **organization レベル**に作る

要素	役割
workforce pool provider	外部 IdP との信頼関係。**OIDC** または **SAML 2.0**
attribute mapping(属性マッピング)	IdP のクレームを Google 側の属性に写す。`google.subject` は必須(一意識別子・最大127バイト)。`google.groups` 、`google.display_name` は任意
attribute condition	CEL 式で、条件に合う ID だけ入場させる
プリンシパル識別子	`principal://iam.googleapis.com/locations/global/workforcePools/POOL_ID/subject/SUBJECT` や `principalSet://.../group/GROUP` の形で IAM に直接付与する

カスタムの属性マッピングは最大50個まで定義でき、CEL(Common Expression Language)で記述します。

使い分けの判断軸は次のとおりです。ここが最頻出です。

要件	選ぶもの
外部の従業員・請負業者に、アカウント同期なしで一時的に Google Cloud を使わせたい	**Workforce Identity Federation**
自社のアプリ・CI/CD・AWS/Azure のワークロードが Google Cloud API を呼ぶ	**Workload Identity Federation**
Gmail・Drive など Workspace サービスも使わせたい／Google 側にユーザー実体が必要	**Cloud Identity + GCDS + SSO**

要件	選ぶもの
GKE の Pod が Google Cloud API を呼ぶ	**Workload Identity Federation for GKE**

Workforce Identity Federation は Google Cloud コンソール（`console.cloud.google.com/?organizationId=...` ではなく `console.cloud.google/` の workforce 用の入口）、gcloud、および API から利用できます。VPC Service Controls や Access Context Manager の属性ベース制御と組み合わせると、外部 ID に対しても境界制御が効きます。

1-2 サービスアカウントの管理 [1.2]

1-2-1 service account が必要になる場面の特定 [1.2]

service account は「人ではない主体」の ID です。次の場面で使います。

- Compute Engine の VM、GKE の Pod、Cloud Run のサービス、Cloud Functions などが Google Cloud API を呼ぶとき
- オンプレや他クラウドのアプリが Google Cloud のリソースにアクセスするとき
- CI/CD パイプライン(Cloud Build、GitHub Actions 等)がデプロイするとき
- サービス間で権限を委譲するとき(impersonation)

逆に、**人間が使うべきでない**のがポイントです。「開発者に service account key を配って作業させる」は、監査ログに個人が残らず、鍵が流出しても追跡できないため、常に誤答です。人間には自分のユーザー アカウントを使わせ、必要なら **service account impersonation** を許可します。

service account は**プリンシパル(誰か)であると同時にリソース(何か)**という二面性を持ちます。だから「その service account を使う権限(`roles/iam.serviceAccountUser`)」と「その service account が持つ権限(付与されたロール)」の2階層があり、ここが最も事故が起きる箇所です。

1-2-2 service account の作成・無効化・認可(default service account を含む) [1.2]

作成は `gcloud iam service-accounts create` です。認可は2種類あることを明確に区別します。

種類	意味	例
service account に**ロールを付ける**	その SA が何をできるか	プロジェクトで `roles/storage.objectViewer` を SA に付与
service account **に対する**ロールを付ける	誰がその SA を使えるか	ユーザーに `roles/iam.serviceAccountUser`(その SA としてリソースを動かす)、`roles/iam.serviceAccountTokenCreator`(短期認証情報を発行する)

default service account(既定のサービスアカウント)は特に重要です。

既定 SA	既定の権限	リスクと対策
Compute Engine default service account(`PROJECT_NUMBER-compute@developer.gserviceaccount.com`)	従来はプロジェクトの **Editor** ロール	過剰権限。VM が侵害されるとプロジェクト全体が危険。**専用の最小権限 SA を作って VM にアタッチする**
App Engine default service account	同様に強い権限	同上
Google APIs service agent など	サービスの内部動作用	消さない。消すとサービスが壊れる

対策の定番は organization policy による予防です。

- `constraints/iam.automaticIamGrantsForDefaultServiceAccounts` を **enforce** すると、既定 SA への Editor 自動付与が止まる
- `constraints/compute.requireShieldedVm` 、`constraints/compute.vmExternalIpAccess` と併せて VM の攻撃面を絞る
- VM 作成時に **access scopes** を「Allow full access to all Cloud APIs」にしない。ただし access scopes は古い仕組みで、**IAM ロールで絞るのが正**。両者が併存する場合は**より制限の強い方が勝つ**

無効化(disable)と削除(delete) の違いも問われます。無効化は `gcloud iam service-accounts disable` で即座にトークン発行を止められ、元に戻せます。削除は30日以内なら `undelete` できますが、同名で作り直しても**内部 ID(unique ID) が変わる**ため、IAM バインディングが `deleted:serviceAccount:...?uid=...` の形で残り、復旧が面倒です。したがって「使っていない SA を止めたい。影響が読めない」場合の第一手は**削除ではなく無効化**です。

未使用 SA の洗い出しには、**IAM の service account insights(Policy Intelligence の一部)** と `activity analyzer`(最終認証時刻)を使います。

1-2-3 service account の保護 [1.2]

リスク	対策
過剰権限	用途ごとに SA を分ける(1ワークロード 1SA)。Editor/Owner を付けない。Recommender の提案で絞る
SA が別の強い SA を偽装できる	`roles/iam.serviceAccountTokenCreator` の付与先を厳格に管理する。これは**権限昇格の経路**そのもの
キーの流出	キーを作らない。作るなら短命に
SA の作成・鍵作成が野放し	organization policy `constraints/iam.disableServiceAccountCreation`、`constraints/iam.disableServiceAccountKeyCreation`、`constraints/iam.disableServiceAccountKeyUpload`
SA がプロジェクト外から使われる	`constraints/iam.disableCrossProjectServiceAccountUsage` の解除を慎重に扱う
攻撃者による横展開	VPC Service Controls の境界内に閉じ込め、Access Context Manager で接続元を絞る

1-2-4 service account key の保護・監査・利用の抑制 [1.2]

user-managed service account key(ユーザー管理の鍵) は、有効期限のない秘密鍵(JSON)です。これが漏れると、恒久的にそのアイデンティティを乗っ取られます。GitHub への誤コミットが典型的な事故です。

図 1-3　service account key を避ける優先順位

優先順位は次のとおりで、試験ではこの順で「より上の手段が取れるならそれが正解」になります。

1. **リソースに SA をアタッチする**(VM/Cloud Run/GKE)。鍵は不要。メタデータ サーバーから短期トークンが取れる
2. **Workload Identity Federation**(外部・他クラウド・GitHub Actions 等)。鍵は不要

3. **service account impersonation**(人間・別サービスから)。鍵は不要
4. どうしても無理な場合のみ **service account key**。ただし厳格に管理する

鍵を使わざるを得ない場合の管理策です。

- organization policy でキー作成を既定禁止にし、例外のフォルダ/プロジェクトだけ許可する
- `constraints/iam.serviceAccountKeyExpiryHours` で**キーの有効期間の上限**を強制する
- 鍵は **Secret Manager** に置き、コードやイメージに埋め込まない
- **定期的にローテーション**する(新しい鍵を作る → 配布 → 古い鍵を削除、の順で無停止に)
- 監査: Cloud Asset Inventory と Policy Analyzer で鍵の一覧を取り、Security Command Center の **Security Health Analytics** の検出項目(`SERVICE_ACCOUNT_KEY_NOT_ROTATED`、`USER_MANAGED_SERVICE_ACCOUNT_KEY` 等)で常時検知する
- **Cloud Audit Logs** の `google.iam.admin.v1.CreateServiceAccountKey` にアラートを張る
- 漏洩時の初動は **鍵の削除**(SA の削除ではない)。鍵を消せばそのクレデンシャルは即座に無効になる

なお **Google-managed keys(Google 管理の鍵)** は、SA に自動で紐づき、Google が自動ローテーションします。こちらはダウンロードできないため、事故の対象になりません。「鍵がローテーションされていない」という指摘の対象は常に user-managed key です。

1-2-5 short-lived credentials(短期認証情報)の管理と作成 [1.2]

short-lived credentials(短期認証情報) は、有効期限が数分〜数時間のトークンです。鍵を配らない設計の中心です。

種類	何に使うか	既定の寿命
OAuth 2.0 access token	Google Cloud API の呼び出し	既定1時間(最大12時間まで組織ポリシーで延長可)
ID token(OIDC)	Cloud Run/Cloud Functions/IAP で保護されたエンドポイントへの認証	1時間
self-signed JWT	一部 API への直接認証	短時間
signed URL / signed policy document	Cloud Storage への一時的な直接アクセス	指定した期間(最大7日、鍵の種類による)

発行は **IAM Service Account Credentials API**(`iamcredentials.googleapis.com`)の `generateAccessToken`、`generateIdToken`、`signBlob`、`signJwt` を使います。呼び出し側には `roles/iam.serviceAccountTokenCreator` が必要です。

`constraints/iam.allowServiceAccountCredentialLifetimeExtension` は、アクセストークンの寿命を12時間まで延ばすことを許す組織ポリシーです。長時間バッチのために使いますが、**長くするほど漏洩時の被害が伸びる**ため、既定では設定しません。

1-2-6 Workload Identity Federation の構成 [1.2]

Workload Identity Federation は、外部の ID プロバイダが発行したトークン(OIDC/SAML)を Google Cloud の短期トークンに交換する仕組みです。

AWS、Azure(Entra ID)、GitHub Actions、GitLab、Kubernetes、オンプレの OIDC IdP などから、鍵なしで Google Cloud にアクセスできます。

構成要素は次のとおりです。

要素	説明
workload identity pool	外部ワークロード ID の入れ物。プロジェクト レベル
workload identity pool provider	外部 IdP との信頼。AWS / OIDC / SAML
attribute mapping	外部トークンのクレーム → `google.subject` 、`attribute.xxx`
attribute condition	CEL で入場条件を絞る。例: GitHub の特定リポジトリ・特定ブランチのみ
付与方法	① `principalSet://` を直接リソースに付与(**direct resource access**)、②service account を impersonate させる

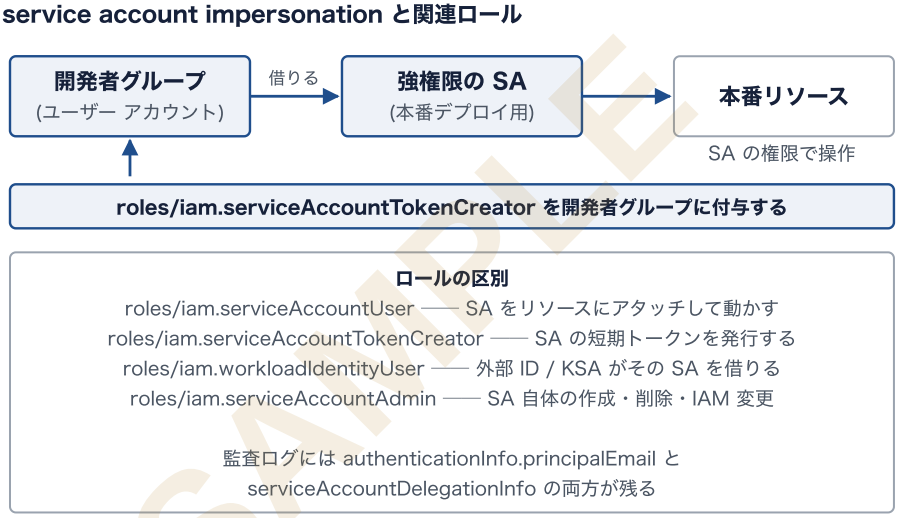
`attribute condition` の設計が最重要です。たとえば GitHub Actions からのアクセスを受ける場合、`assertion.repository == "myorg/myrepo"` のような条件を必ず入れます。これが無いと、**GitHub 上の任意のリポジトリ**から自社の Google Cloud に入れてしまいます。試験の「設定は動いているが安全でない」型の問題は、ここを問うものが多いです。

Workload Identity Federation for GKE は、GKE の Kubernetes ServiceAccount(KSA)を Google の service account または直接 IAM プリンシパルに紐づける機能です。従来のノードのメタデータ サーバー経由で**ノードの SA を Pod が使い回す**構成は、Pod 間の権限分離ができないため誤りです。クラスタとノードプールで Workload Identity を有効化し、KSA にアノテ

ーションを付けるのが正解です。GKE の metadata server 保護(**GKE metadata server**)も併せて有効になります。

1-2-7 service account impersonation の管理 [1.2]

impersonation(なりすまし・権限借用) は、あるプリンシパルが service account の権限で操作する仕組みです。



impersonation chain は指定できるが、経路が長いほど追跡が難しいため1段に留める。

図 1-4　service account impersonation と関連ロール

```
gcloud storage ls --impersonate-service-account=deploy-
sa@PROJECT.iam.gserviceaccount.com
```

利点は次のとおりです。

- 鍵が不要

- 監査ログに「**誰が**その SA を借りたか」が残る（`authenticationInfo.principalEmail` と `serviceAccountDelegationInfo` の両方）
- 権限を一時的に借りるだけなので、常時強権限を持たせずに済む

関連ロールの区別が頻出です。

ロール	できること
`roles/iam.serviceAccountUser`	その SA を**リソースにアタッチ**して動かす（VM 作成時に SA を指定する等）
`roles/iam.serviceAccountTokenCreator`	その SA の**短期トークンを発行**する（直接の impersonation）
`roles/iam.workloadIdentityUser`	外部 ID / KSA がその SA を借りる
`roles/iam.serviceAccountAdmin`	SA 自体の作成・削除・IAM 変更

「開発者が本番へデプロイできるようにしたいが、本番の強い権限を常時持たせたくない」という要件には、**強権限の SA を作り、開発者グループに `serviceAccountTokenCreator` を付けて impersonation させる**のが定番の答えです。さらに一時性を強めたい場合は、次節の Privileged Access Manager を組み合わせます。

impersonation chain（委譲チェーン）も可能ですが、経路が長いほど追跡が難しくなります。`--impersonate-service-account` にカンマ区切りで委譲チェーンを指定できますが、設計上は1段に留めるのが安全です。

1-3 認証の管理 [1.3]

1-3-1 user account の password and session management policy [1.3]

Admin Console で設定する項目です。Google Cloud の IAM ではなく **Cloud Identity 側**の設定である点を押さえます。

設定	内容	注意点
password policy	最小/最大長(8〜100文字)、強度の強制(enforce strong password)、再利用の禁止、有効期限	SSO を使っている場合、Google 側のパスワードポリシーは**SSO をバイパスするアカウントにしか効かない**
パスワードの再設定強制	次回ログイン時に変更を要求	侵害疑い時の初動
Google Cloud session length(セッション長)	Google Cloud コンソール・gcloud・API へのセッションの再認証間隔。1時間〜24時間、または無期限	機密プロジェクトほど短く。**Google Cloud session control** として Admin Console の Security から設定
web and mobile app session length	Workspace アプリのセッション	Google Cloud のセッションとは別設定
組織部門(OU)単位の適用	部門ごとに異なる強度を適用できる	全社一律にせず、リスクに応じて分ける

「開発者がコンソールに開きっぱなしで放置している」という場面の答えは、**Google Cloud session length を短くする**です。IAM ロールの見直しや 2SV は別の問題を解く手段であり、この設問では誤答になります。

1-3-2 Security Assertion Markup Language(SAML)と OAuth の設定 [1.3]

プロトコル	何のためのものか	Google Cloud での使い所
SAML 2.0	企業向けの**認証(SSO)**。XML ベースのアサーションを IdP が署名して SP に渡す	third-party IdP との SSO、Workforce/Workload Identity Federation の SAML プロバイダ
OAuth 2.0	**認可の委譲**。アプリがユーザーに代わってリソースにアクセスするためのトークンを得る	アプリからの API アクセス、service account の認証、OAuth client の作成
OpenID Connect(OIDC)	OAuth 2.0 の上に構築された**認証**の層。ID token を返す	Workload Identity Federation、IAP、Cloud Run の認証

SAML 設定の要素は、IdP の **Sign-in page URL**、**Sign-out page URL**、**change password URL**、**verification certificate(署名検証用の公開鍵証明書)**、そして SP 側の **ACS URL** と **Entity ID** です。IdP 側では NameID を Google のプライマリ メールアドレスに一致させる必要があります。「SSO でログインできるが、ユーザーが見つからないと言われる」という不具合の原因は、この NameID と Cloud Identity 上のメールアドレスの不一致か、GCDS によるプロビジョニング漏れです。

OAuth の管理では次を押さえます。

- **OAuth consent screen(同意画面)** を internal(組織内のみ)にすると、外部ユーザーは認可できない
- **API controls(app access control)** で、サードパーティ アプリが Workspace/Cloud Identity データへアクセスする範囲を管理者が許可

制にできる(**trusted / limited / blocked**)

- `constraints/iam.restrictCrossProjectServiceAccountLienRemoval` などとは別に、**OAuth client の作成を組織ポリシーで抑制**する運用もある
- **domain-wide delegation** は OAuth スコープ単位で許可する。広いスコープを与えない

1-3-3 2-step verification(2SV)の構成と強制 [1.3]

2-step verification(2段階認証)は Admin Console で組織部門・グループ単位に強制します。方式の強度は次の順です。

方式	フィッシング耐性	備考
security key(FIDO/WebAuthn、Titan Security Key)	最も高い	特権アカウントには必須。**Advanced Protection Program** で更に強化
Google prompt	高い	端末に通知。番号照合あり
Google Authenticator 等の TOTP	中	オフラインでも使える
backup codes	中	紛失時の非常口。安全に保管
SMS / 音声通話	低い	SIM スワップに弱い。特権アカウントでは使わない

強制の実務では次の順で進めます。

1. まず enforcement を **オプション**にして登録を促す(登録猶予期間 = **enrollment period** を設定)
2. 例外グループ(サービス用アカウント等)を明示する

3. 段階的に **enforce** へ切り替える。特権 OU から先に強制する
4. super administrator には security key を必須にする
5. **新規ユーザーの猶予期間**を短くする

「一部のユーザーが 2SV を回避している」原因の定番は、**SSO を使っており認証が IdP 側で完結しているか、enforcement が該当 OU に適用されていない**かの2つです。前者の場合の正解は「IdP 側で MFA を強制する」で、「Google 側の 2SV を強制する」だけでは解決しません。

1-4 認可制御の管理と実装 [1.4]

1-4-1 IAM の基本構造 [1.4]

Google Cloud の IAM は「**誰が(principal)** に **どのロール(role)** を **どのリソースで(resource)**」という allow policy(旧称: IAM policy)の集合です。

概念	内容
principal	user、group、serviceAccount、domain、`principalSet://` (federated)、allUsers、allAuthenticatedUsers
basic roles	Owner / Editor / Viewer。**広すぎるため本番では使わない**
predefined roles	Google が用意した製品別の適正粒度のロール
custom roles	権限を個別に選ぶ。組織またはプロジェクトに定義。ローンチ ステージ (ALPHA/BETA/GA)を持つ
allow policy	リソースに付く「許可」の定義。**継承される**

概念	内容
deny policy	「拒否」の定義。allow より**強い**。組織・フォルダ・プロジェクトに付ける

`allUsers`(誰でも)と `allAuthenticatedUsers`(Google アカウントを持つ誰でも)は公開を意味します。Cloud Storage バケットへの `allUsers` 付与は情報漏洩の典型なので、`constraints/storage.publicAccessPrevention` と `constraints/iam.allowedPolicyMemberDomains` で塞ぐのが定石です。

1-4-2 privileged roles と separation of duties の管理 [1.4]

separation of duties(職務の分離) は、1人が「実行」と「承認/監査」の両方を持たない設計です。Google Cloud での実装例を挙げます。

分離したいもの	実装
ログの生成者と閲覧者	ログ用プロジェクトを分け、`roles/logging.viewer` は監査チームのみ。ワークロード側には**書き込みのみ**
鍵の管理者と鍵の利用者	`roles/cloudkms.admin`(鍵を作る)と `roles/cloudkms.cryptoKeyEncrypterDecrypter`(鍵を使う)を別チームに。管理者に暗号化操作権限を与えない
課金の管理者とリソース作成者	`roles/billing.admin` と `roles/resourcemanager.projectCreator` を分ける
組織ポリシーの設定者とプロジェクト管理者	`roles/orgpolicy.policyAdmin` を専任のガバナンス チームへ

分離したいもの	実装
ネットワーク設計者とワークロード開発者	Shared VPC のホスト プロジェクトに `roles/compute.networkAdmin`、サービス プロジェクトには `roles/compute.networkUser` のみ

`roles/cloudkms.admin` が鍵の暗号化・復号を**できない**ことは頻出の細部です。同様に `roles/iam.securityReviewer` は IAM ポリシーを**読むだけ**で変更できません。

privileged role(Owner、Organization Administrator、Security Admin、Billing Admin、Folder Admin など)は、次のように扱います。

- 個人ではなく**グループ**に付与し、メンバーシップを人事プロセスで管理する
- 常時付与ではなく **Privileged Access Manager** による一時付与に置き換える
- **Cloud Audit Logs** で付与操作(`SetIamPolicy`)にアラートを張る
- **Security Command Center** の検出項目で過剰権限を継続的に洗い出す

1-4-3 IAM と access control list(ACL)の権限管理 [1.4]

Cloud Storage には歴史的に **ACL(アクセス制御リスト)** があり、IAM と二重にアクセスを制御できてしまいます。ACL はオブジェクト単位で付き、IAM とは別経路なので、**IAM だけを見ても実際のアクセス権が分からない**という監査上の問題を生みます。

正解は **uniform bucket-level access(均一なバケットレベルのアクセス)** を有効化して ACL を無効化し、IAM に一本化することです。組織ポリシー

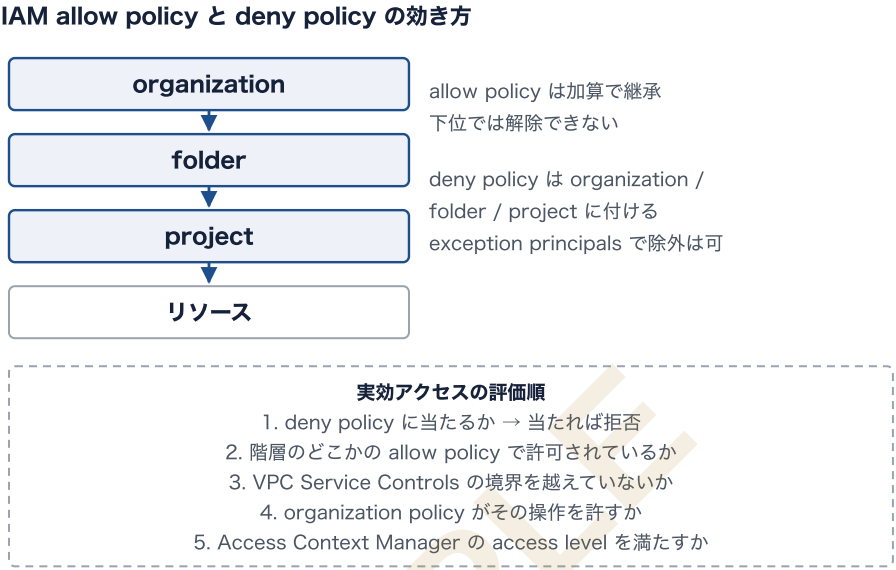
`constraints/storage.uniformBucketLevelAccess` で全社に強制できます。

方式	使いどころ	注意
uniform bucket-level access(IAM のみ)	既定にすべき。監査が単純になる	有効化から90日以内なら元に戻せる。以降は不可
fine-grained(ACL 併用)	オブジェクト単位で異なる相手に配る古い要件	権限の可視性が下がる。**signed URL** で代替できることが多い

BigQuery でも、データセット/テーブル/**行レベル・列レベル**のアクセス制御があり、列レベルは **policy tag(Data Catalog / Dataplex Universal Catalog のタグ)** と `roles/datacatalog.categoryFineGrainedReader` の組み合わせで実装します。**Cloud Storage の ACL と BigQuery の policy tag を混同させる選択肢**が出るので、製品ごとに整理しておきます。

1-4-4 IAM conditions と IAM deny policies による付与 [1.4]

IAM conditions(条件付きロール付与) は、CEL 式が真のときだけロールを有効にします。



「Owner を持っているのに操作できない」の候補は、VPC Service Controls の境界、organization policy の制約、deny policy、access level 不一致、API 自体が無効、の5つ。

図 1-5　IAM allow policy と deny policy の効き方

条件属性	例	用途
`request.time`	`request.time < timestamp("2026-12-31T00:00:00Z")`	**期限付きアクセス**(請負業者、緊急対応)
`resource.name` / `resource.type`	`resource.name.startsWith("projects/_/buckets/prod-logs")`	特定リソースだけに絞る
`resource.service`	`resource.service == "compute.googleapis.com"`	サービス単位

条件属性	例	用途
`request.auth.access_levels`	Access Context Manager の access level	**接続元の IP・デバイス条件**で絞る

条件をサポートするのは一部のサービスに限られる点、そして **基本ロール(Owner/Editor/Viewer)には条件を付けられない**点が引っかけです。

IAM deny policies(拒否ポリシー) は、allow より優先して権限を**拒否**します。

- 組織・フォルダ・プロジェクトに付ける(リソース単位ではない)
- **denied principals**、**denied permissions**、**exception principals**、および CEL の **denial condition** を持つ
- 評価順は「**deny が1つでも当たれば拒否**」。上位で継承した allow を打ち消せる
- 権限は `service.googleapis.com/resource.action` の形式で書く

典型的な使い所は「組織全体で `iam.serviceAccountKeys.create` を誰も実行できないようにするが、鍵管理チームだけ例外にする」です。**allow policy を消して回るのでは、上位からの継承や新規プロジェクトに追随できない**ため、deny policy が正解になります。

Cloud Storage の **bucket IAM に加えて deny policy** を効かせることで、「バケットは公開設定にできるが、この組織では誰も公開できない」といった不変条件を作れます。

1-4-5 organization / folder / project / resource レベルでの least privilege [1.4]

principle of least privilege(最小権限の原則) の実装は、階層のどこに付けるかの判断です。

付与レベル	適する場面	危険な例
organization	全社共通の監査・セキュリティチーム（`roles/iam.securityReviewer`、`roles/logging.viewer`）	ここに Editor を付ける
folder	部門・環境(本番/検証)単位の共通権限	本番フォルダに開発者グループを付ける
project	ほとんどのワークロード権限はここ	個人に直接付ける
resource(バケット、鍵、トピック、Pub/Sub、BigQuery データセット等)	特定リソースだけの限定アクセス	上位で広く付けて resource で絞ろうとする(**絞れない**)

継承は加算のみで、下位で減算はできません。これが最重要の性質です。「フォルダで Editor を継承しているプロジェクトで、特定の人だけ読み取りに落としたい」場合、プロジェクトの allow policy を触っても無意味で、**deny policy** か**上位の付与そのものを外す**しかありません。

設計手順としては、①ワークロード単位でプロジェクトを分ける ②権限はグループに付ける ③predefined role で足りるか確認し、足りなければ custom role ④条件付き付与で期限や範囲を絞る ⑤Recommender で過剰権限を継続的に削る、の順です。

1-4-6 Access Context Manager の構成 [1.4]

Access Context Manager は、**access level(アクセスレベル)** という「接続元の条件」を定義するサービスです。定義した access level は、**VPC Service Controls の ingress/egress ルール**、**IAP のコンテキスト アウェア アクセス**、**IAM conditions** から参照されます。

access level の種類	内容
basic access level	条件を AND/OR で組む。IP subnetwork(CIDR)、**geographic region(国)**、必要な **device policy**、principals(ユーザー・SA)
custom access level	CEL 式で細かく記述する

device policy では、**screen lock の要求**、**ストレージの暗号化**、**OS の最小バージョン**、**企業所有デバイス**、**endpoint verification による承認済みデバイス** などを条件にできます。これが Google の **BeyondCorp(zero trust)** の実装です。

「社内 VPN を必須にせず、社給端末からのみ管理コンソールに入れたい」という要件は、VPN(誤答: Cloud VPN を張る)ではなく、**Access Context Manager の device policy + IAP** で解きます。ここは Professional 級で頻出の設計判断です。

access level は organization 配下の **access policy** に属します。組織全体の access policy は1つ(スコープ付きポリシーで例外を作れる)で、`roles/accesscontextmanager.policyAdmin` を持つ管理者が管理します。

1-4-7 Policy Intelligence の適用 [1.4]

Policy Intelligence は、IAM の実態を可視化・最適化するツール群です。それぞれ何を答えるためのものかを区別します。

いきなり適用する、手作業で全ログを読む、はいずれも誤答になる。

図 1-6　Policy Intelligence —— どのツールが何に答えるか

ツール	答える問い
Policy Troubleshooter	「なぜこの人はこのリソースにアクセスできる(できない)のか」。当たっている allow/deny を辿る
Policy Analyzer(IAM policy analysis)	「この権限を持っているのは誰か」「この人は何にアクセスできるか」を組織全体で検索する
IAM Recommender(role recommendations)	過去90日の実使用に基づき、**過剰なロールを絞る提案**を出す

ツール	答える問い
Policy Simulator	ロール変更を**適用する前に**、過去のアクセスが壊れないかを検証する
Lateral Movement Insights	プロジェクト間で SA を悪用した横展開が可能な経路を検出する
Activity Analyzer	SA・鍵の**最終利用日時**を見る。未使用の棚卸しに使う

実務の流れは、Recommender の提案 → Policy Simulator で影響確認 → 適用 → Policy Analyzer で残存を確認、です。**「いきなり適用する」「手作業で全ログを読む」は誤答**になります。

1-4-8 group を通じた権限管理 [1.4]

IAM は**個人ではなくグループに付ける**のが原則です。理由は3つあります。

1. 異動・退職時にグループのメンバーシップだけ変えればよい(IAM を触らない)
2. 監査時に「この権限を持つのは誰か」がグループ定義で読める
3. IAM ポリシーのバインディング数の上限(1500 プリンシパル程度)に達しにくい

設計の型は「**職務ごとのグループ**を作り、グループにロールを付ける」です。例: `gcp-prod-network-admins@example.com` に本番フォルダの `roles/compute.networkAdmin`。グループの入れ子(nested groups)も使えますが、深くすると実効権限が読みにくくなるため2段程度に留めます。

注意点として、**グループのオーナーがメンバーを自由に追加できる**設定だと、権限昇格の穴になります。Admin Console でグループの **access settings** を

管理者のみに制限するか、AD/HR 由来の同期グループだけを IAM に使う、という運用が安全です。

1-4-9 Privileged Access Manager(PAM)のユースケースと構成 [1.4]

Privileged Access Manager(PAM) は、**just-in-time(必要なときだけ)の一時的な権限昇格**を提供するサービスです。常時強権限(standing privilege)をなくすための道具です。

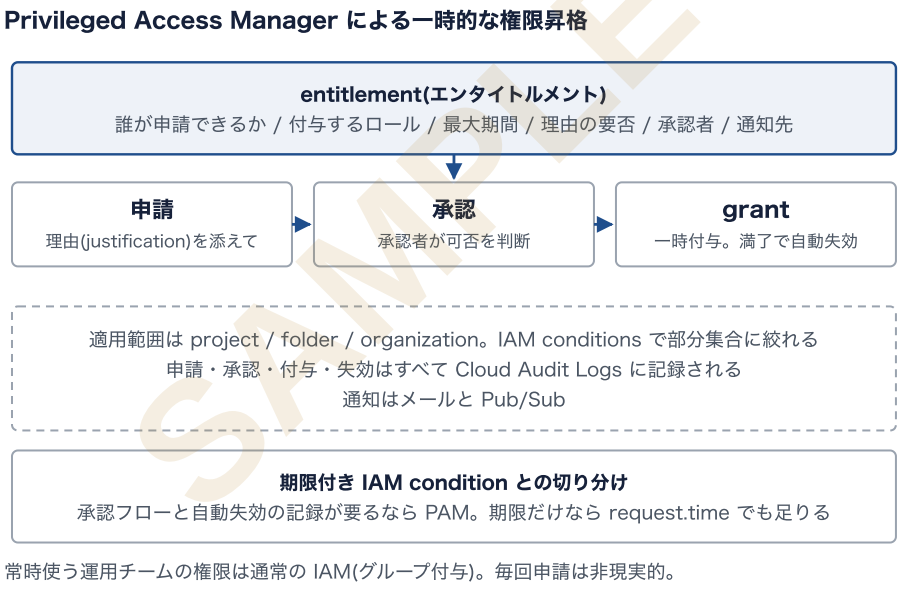


図 1-7　Privileged Access Manager による一時的な権限昇格

概念	内容
entitlement(エンタイトルメント)	誰が申請できるか、付与するロール、最大期間、**理由(justification)の要否**、承認者、通知先を定義する

概念	内容
grant(グラント)	実際の一時付与。期間満了で**自動的に失効**する
適用範囲	**project / folder / organization**。IAM conditions を併用してリソースの部分集合に絞れる
承認	承認者を指定できる。多段承認や service account を承認者にする拡張もある
監査	すべての申請・承認・付与・失効が **Cloud Audit Logs** に記録される
通知	メールと **Pub/Sub**

ユースケースは、緊急対応(break-glass)、機密リソースへの限定的な作業、請負業者への期間限定アクセス、内部不正の抑止(多者承認)です。

似た選択肢との切り分けが試験の勘所です。

要件	正解	誤答になりやすい選択肢
障害対応時だけ本番の強権限を、承認と記録付きで一時的に	**PAM の entitlement**	期限付き IAM condition(承認フローと自動失効の記録が無い)
期限が決まっている請負業者に読み取りを	IAM conditions の `request.time` でも可	PAM でも可。要件に「承認」があるなら PAM
常時使う運用チームの権限	通常の IAM(グループ付与)	PAM(毎回申請は非現実的)

PAM は内部的に**時間条件付きの IAM role binding** を作ります。そのため、PAM の管理外で該当バインディングを手で編集すると整合が崩れます。

Terraform で管理する場合は非権威的(non-authoritative)なリソースを使う、という注意点があります。

1-5 リソース階層の定義 [1.5]

1-5-1 folder と project を大規模に管理する [1.5]

Google Cloud の **resource hierarchy(リソース階層)** は次の構造です。

```
organization → folder(最大10階層) → project → リソース
```

大規模運用の設計原則は次のとおりです。

原則	内容
フォルダは**組織構造ではなく統制単位**で切る	部署名で切ると組織改編のたびに作り直しになる。環境(prod/nonprod)・機密度・規制対象で切ると安定する
プロジェクトは**分離の単位**	課金、IAM、割り当て、ネットワーク、ログの境界。ワークロード×環境で1プロジェクトが基本
共通サービス用のプロジェクトを分ける	Shared VPC ホスト、ログ集約、Cloud KMS の鍵、Artifact Registry など
作成を自動化する	**Terraform** と **project factory**、Cloud Build/Cloud Deploy によるパイプライン。手作業で作らない
lien(リーエン) を付ける	`gcloud resource-manager liens create` で誤削除を防ぐ

原則	内容
プロジェクトの上限・命名規約	命名規則(例: `env-team-workload`)と**labels**(cost center、data classification、owner)を必須にする

Cloud Foundation Toolkit や Terraform モジュールで landing zone(基盤)を組む、という答えが Professional 級では期待されます。

1-5-2 pre-built / custom organization policies の管理 [1.5]

Organization Policy Service は、リソースの**構成そのもの**を制限します。IAM が「誰ができるか」なのに対し、organization policy は「**何が作れるか**」を縛ります。この違いが最頻出です。

種類	内容
boolean constraint	有効/無効。例: `constraints/compute.disableSerialPortAccess`、`constraints/sql.restrictPublicIp`、`constraints/storage.publicAccessPrevention`
list constraint	許可/拒否のリスト。例: `constraints/gcp.resourceLocations`(データの所在地制限)、`constraints/compute.vmExternalIpAccess`、`constraints/iam.allowedPolicyMemberDomains`

種類	内容
custom organization policy	CEL でリソースのフィールドに条件を書く独自制約。例:「特定のマシンタイプ以外の VM 作成を拒否」「CMEK の無い Cloud SQL を拒否」

セキュリティで頻出の pre-built constraint を整理します。

制約	効果
`constraints/iam.disableServiceAccountKeyCreation`	SA キーの新規作成を禁止
`constraints/iam.allowedPolicyMemberDomains`	自社ドメイン外へ IAM を付与できなくする (Domain restricted sharing)
`constraints/compute.vmExternalIpAccess`	外部 IP を持つ VM を制限
`constraints/compute.restrictVpcPeering`	VPC peering の相手を制限
`constraints/compute.requireOsLogin`	SSH を **OS Login** 経由に強制(メタデータの SSH 鍵を使わせない)
`constraints/compute.requireShieldedVm`	Shield VM を強制
`constraints/gcp.resourceLocations`	リージョンを限定(データ所在地の規制対応)
`constraints/gcp.restrictCmekCryptoKeyProjects`	CMEK に使える鍵プロジェクトを限定
`constraints/storage.uniformBucketLevelAccess`	ACL を無効化して IAM に一本化

制約	効果
`constraints/sql.restrictPublicIp`	Cloud SQL のパブリック IP を禁止
`constraints/essentialcontacts.allowedContactDomains`	通知先ドメインを限定

継承と例外の挙動が重要です。ポリシーは階層で**継承**され、下位で `inheritFromParent` を false にして上書きしたり、list constraint に例外値を足したりできます。上書きを許すかどうか自体を管理するため、`roles/orgpolicy.policyAdmin` の付与先は厳しく絞ります。

dry-run モード(`--dry-run` の適用)で、実際にブロックする前に「もし適用したら何が落ちるか」を監査ログで確認できます。本番導入前にこれを回すのが正しい手順です。

1-5-3 リソース階層によるアクセス制御と permissions inheritance [1.5]

継承の性質をまとめます。

対象	継承の向き	下位で解除できるか
IAM allow policy	上位 → 下位に**加算**	できない
IAM deny policy	上位 → 下位に適用	できない(例外プリンシパルで除外は可能)
organization policy	上位 → 下位	制約と設定次第で上書き可
階層型ファイアウォールポリシー(hierarchical firewall policy)	organization/folder → VPC	`goto_next` で下位に判断を委ねられる

対象	継承の向き	下位で解除できるか
ラベル・タグ(**tags**)	tag は階層で継承される	下位で別の値を付けて上書き可

tags(タグ) は、`key:value` を組織で定義し、フォルダ・プロジェクト・一部リソースに付けて、**IAM conditions・organization policy の条件・ファイアウォールポリシー**から参照できる仕組みです。labels(課金・整理用の自由なキー値)とは別物である点が引っかけです。tag は IAM で管理され(`roles/resourcemanager.tagUser` 等)、アクセス制御に使える一方、label はアクセス制御に使えません。

実効アクセスの読み解き方は、次の順で評価すると考えます。

1. **deny policy** に当たるか → 当たれば拒否
2. 階層のどこかの **allow policy** で許可されているか → されていれば許可
3. **VPC Service Controls** の境界を越えていないか → 越えていれば拒否
4. **organization policy** がその操作(リソース作成等)を許すか
5. **Access Context Manager** の access level 条件を満たすか

この5段の順序を頭に入れておくと、「IAM で Owner を持っているのに操作できない」という定番のトラブルシューティング問題に即答できます。答えの候補は、VPC Service Controls の境界、organization policy の制約、deny policy、access level 不一致、そして API 自体が無効、の5つです。

1-6 ドメイン1の設計判断まとめと誤答の切り方 [1.4]

ここまでの内容を、試験本番で使える形に圧縮します。ドメイン1の設問は「アイデンティティ」「認証」「認可」「階層」の4層のどこで解くかを見分けるものが

大半です。

1-6-1 identities(アイデンティティの種類)ごとの permissions の granting [1.4]

granting permissions to different types of identities(さまざまな種類のアイデンティティへの権限付与) は、相手の種類によって使う道具が変わります。

identities の種類	表現	付与の考え方	注意点
社内の従業員	`user:...` / `group:...`	**group に付ける**。個人には付けない	退職時の削除漏れが最大のリスク
社外の協力会社(短期)	`user:` + **IAM conditions** の `request.time`、または **Privileged Access Manager**	期限を必ず入れる	ドメイン外なら `iam.allowedPolicyMemberDomains` に例外が要る
外部 IdP の人間	`principal://` / `principalSet://` (**Workforce Identity Federation**)	pool と provider を作り、attribute condition で絞る	属性マッピングの取り違えで想定外の相手が入る
自社ワークロード	`serviceAccount:`	リソースにアタッチする。鍵を作らない	default service account の過剰権限
外部・他クラウドのワークロード	`principalSet://` (**Workload Identity Federation**)	attribute condition で発行元を限定	条件が緩いと第三者のリポジトリから入れる

identities の種類	表現	付与の考え方	注意点
誰でも(公開)	`allUsers` / `allAuthenticatedUsers`	原則使わない	`publicAccessPrevention` と domain restricted sharing で塞ぐ
GKE の Pod	KSA + **Workload Identity Federation for GKE**	Pod 単位で分離	ノードの SA を使い回さない

deny を使うべき場面の見分け方も再掲します。「上位から継承した allow を打ち消したい」「組織全体で絶対に許さない操作がある」という要件は、allow policy の編集では解けません。**IAM deny policy** が答えです。

1-6-2 service account の creating、disabling、authorizing の運用手順 [1.2]

creating, disabling, and authorizing service accounts(サービスアカウントの作成・無効化・認可) を、運用の順序として整理します。

1. **creating(作成)**: ワークロード1つにつき1つ。命名で用途が分かるようにする(`svc-<system>-<env>-<role>`)。作成権限は `roles/iam.serviceAccountAdmin` を持つ基盤チームに限定し、開発者には `constraints/iam.disableServiceAccountCreation` を効かせる
2. **authorizing(認可)**: 必要な predefined role だけを、必要なリソース レベルで付与する。Owner/Editor は付けない。他者にその SA を使わせるときは `roles/iam.serviceAccountUser` か `roles/iam.serviceAccountTokenCreator` を明示的に付ける

3. **運用**: 利用状況を **activity analyzer** で確認する。90日使われていない SA は候補として洗い出す
4. **disabling(無効化)**: 削除の前に必ず `gcloud iam service-accounts disable` で止め、影響が出ないことを一定期間観測する。問題が出たら `enable` で即座に戻す
5. **削除**: 影響が無いと確認できてから削除する。削除後30日以内は `undelete` 可能だが、IAM バインディングの復旧は自動ではない

「不要な SA を整理したい」と問われたら、**いきなり削除するのではなく、まず disabling して観測する**が正解です。復旧可能性の差がそのまま採点基準になります。

1-6-3 service account key の usage の securing、auditing、mitigating [1.2]

securing, auditing, and mitigating the usage of service account keys(サービスアカウント キーの利用の保護・監査・低減) を3語に沿って整理します。

securing(保護)

- 作成を組織ポリシーで既定禁止にし、例外を申請制にする
- 作成できる場合も `constraints/iam.serviceAccountKeyExpiryHours` で寿命の上限を強制する
- 保管は **Secret Manager**。リポジトリ・イメージ・CI の環境変数に平文で置かない
- アップロード鍵(`disableServiceAccountKeyUpload`)も統制対象にする

auditing(監査)

- **Cloud Asset Inventory** で組織全体の鍵を棚卸しする
- **Security Health Analytics** の検出項目(user-managed key の存在、未ローテーション)を常時監視する
- **Cloud Audit Logs** の `CreateServiceAccountKey` / `DeleteServiceAccountKey` にアラートを張る
- **activity analyzer** で鍵の最終使用日時を確認し、使われていない鍵を消す

mitigating(利用の低減)

- 鍵を使っている経路を洗い出し、①アタッチ ②**Workload Identity Federation** ③**impersonation** のどれに置き換えられるかを判定する
- 置き換えができない外部システムには、寿命の短い鍵と厳格なローテーションを課す
- 鍵の流出が疑われたら、**鍵の削除**を最優先で行う。SA そのものの削除は依存関係を壊すため後回しにする

1-6-4 2-step verification の configuring と enforcing の段取り [1.3]

configuring and enforcing 2-step verification(2段階認証の構成と強制) は、順序を誤ると業務が止まります。

1. **configuring**: 許可する方式を決める(security key のみ、または security key + Google prompt)。SMS は特権 OU では許可しない
2. 例外の洗い出し: 共有アカウント、システム連携用のアカウント。これらは本来 service account に置き換えるべきものとして棚卸しする
3. **登録期間(enrollment period)** を設けて、ユーザーに登録を促す

4. **enforcing**: 特権 OU から順に enforce へ切り替える。全社一斉にしない
5. 監視: 未登録ユーザーのレポートを Admin Console で確認し、期限までに解消する
6. 復旧手順: backup codes の配布方法と、管理者による一時的な enforcement 解除の手順を用意する

1-6-5 folders と projects を使ったガバナンスの型 [1.5]

folders(フォルダ) の切り方は、統制の効きやすさを決めます。代表的な型を挙げます。

型	構造	向く組織
環境優先	`org / prod / nonprod` の下に部門	本番と非本番で統制を大きく変えたい
部門優先	`org / 部門 / prod・nonprod`	部門ごとに独立性が高い
規制優先	`org / regulated / general`	PCI DSS や医療データなど、規制対象を明確に分けたい
ブートストラップ + ワークロード	`org / common(共通サービス) / workloads / bootstrap`	Terraform ベースの landing zone。Google の推奨に近い

いずれの型でも、**common(共通)フォルダ**に次のプロジェクトを置くのが定石です。

- Shared VPC の host project(ネットワーク)
- ログ集約プロジェクト(logging)
- 鍵管理プロジェクト(Cloud KMS)

- Security Command Center・監視プロジェクト
- Artifact Registry・CI/CD プロジェクト
- 組織ポリシー・Terraform state を管理する bootstrap プロジェクト

folders への組織ポリシー適用は、`--folder=FOLDER_ID` で行います。組織で厳しい既定を敷き、例外が必要な folders でだけ上書きする、という構成が管理しやすくなります。上書きを無制限に許すと統制が骨抜きになるため、`roles/orgpolicy.policyAdmin` の付与先はガバナンス チームに限定します。

サンプルはここまでです

続き(第2章〜巻末)は、GCP PSE 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 PSE 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 Security 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