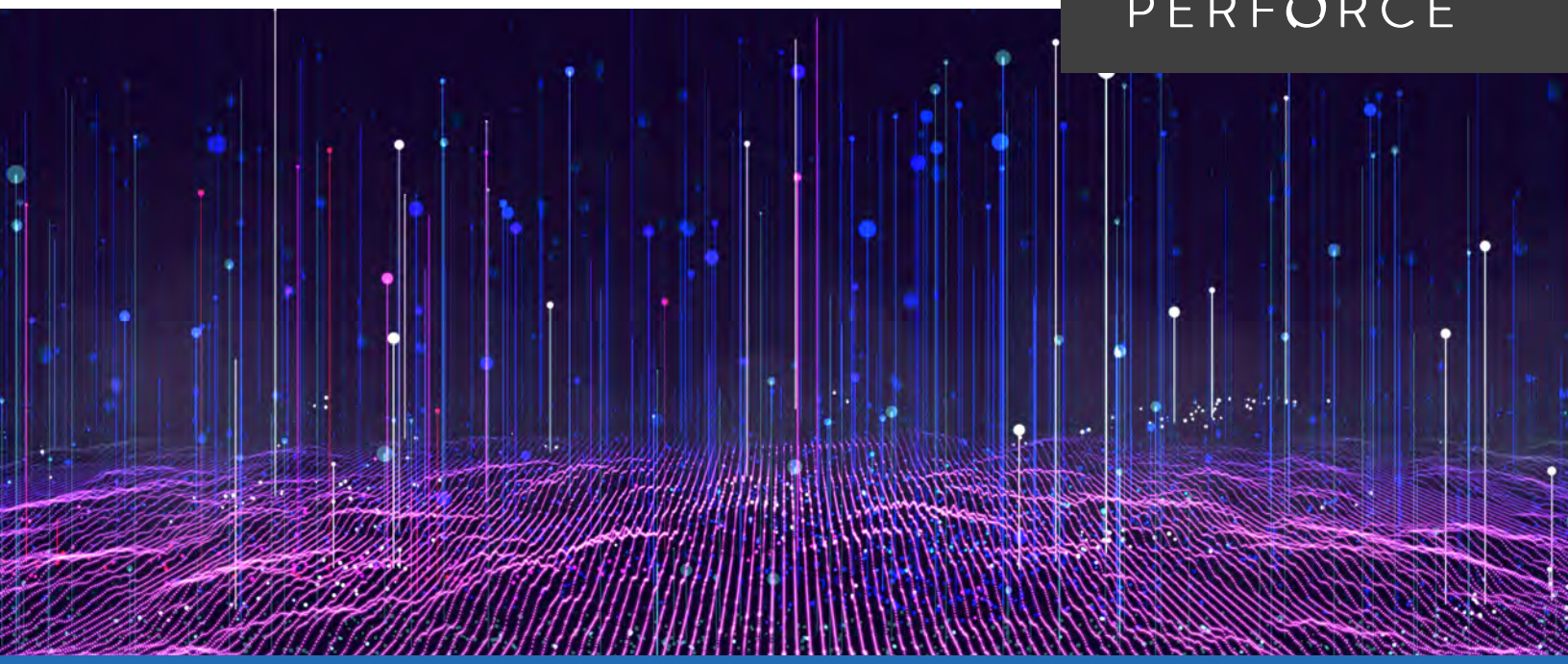
PERFORCE



テクニカルガイド

Helix Core を AWS にデプロイする方法

ゲーム開発チーム向け、ベストプラクティスのご紹介

はじめに

AWS 上には、コンピューティング、ストレージ、ネットワークのリソースが、無限ともいえるほどに溢れています。そして、これらをうまく利用できれば、ゲームの開発サイクルを劇的に改善し、製品リリースまでにかかる時間を大幅に短縮することができます。

本テクニカルガイドは、アーキテクチャ・コンポーネントの観点から、AWS 上でのゲーム開発ワークフローの実装方法や最適化方法についてアドバイスするものです。Perforce Software 社と Amazon Web Services 社（以下、AWS 社）所属のシニアコンサルタントの知識と経験に加え、AWS インフラストラクチャ上で Perforce 製品を実際に利用されているお客様から提供された情報をもとに書かれています。

ご意見・ご感想がありましたら、ぜひ consulting@perforce.com までお送りください。

目次

本テクニカルガイドは、AWS クラウド環境への Helix Core サーバーの標準的なデプロイ方法を説明するものです。例として挙げているトポロジーは、大規模なソフトウェアゲーム開発現場における厳しい要求に対応したものになっていますが、コンセプトや仕組みについては、AWS 上での幅広い Perforce ワークロードで適用が可能です。

本テクニカルガイドは、Perforce 特有の用語（”エッジサーバー”、”レプリカ”等）やAWS 特有の用語（”EC2 インスタンス”、”仮想プライベートクラウド”等）について、ある程度の知識があることを前提として書かれています。また、P.7 で紹介する Perforce Server Deployment Package（SDP）についても言及します。

AWS デプロイメント・トポロジー

ハイブリッド型のトポロジーと AWS のみで構成されたトポロジーの両方について考察します。

ソフトウェアゲーム開発では一般的に、ソースコードに加え、サイズの大きなバイナリファイルを大量にバージョン管理する必要があります。継続的インテグレーションや継続的デプロイメントの実現には、大容量バイナリファイルとその絶え間ない移動が欠かせないため、ハイブリッド型インフラは検討に値しますが、”シンプルさ”の観点でのAWS 単独のトポロジーのメリットも天秤にかけたうえで考える必要があります。

ただ、最終的にどちらを選んだとしても、データの耐久性や可用性を最大化するために、マスターサーバーは AWS 上に置くのがベストと言えるでしょう。

グローバルなデータのやりとりに、（AWS Global Accelerator をはじめとする）AWS の優れた WAN インフラを活用する、というのもトポロジー作成の主な目的のひとつになります。パブリックネットワークの使用は、AWS データセンターから自社オフィスやエンドユーザーへの接続が必要ある時だけに限定するのです。

ハイブリッド型トポロジーについて

ハイブリッド型のデプロイメント・トポロジーの場合、 AWS インフラとオンプレミスインフラ両方の良いところ取りで、可用性、パフォーマンス、システム管理のさまざまなニーズを満たすことができます。

本テクニカルガイドで示す、ハイブリッド型トポロジーは、オンプレミスの既存インフラに多大な投資をしている企業にこそ向いています。

トラフィック全体の98%以上を占めるとも言われる、日常的に発生する read-only トラフィックを、社内 LAN 環境内で処理することによって、AWS のデータ転送料金を最小限に抑えることができます。

「**sync はローカルで、submit は AWS へ**」がハイブリッドトポロジーの考え方です。これを実現するには、Perforce のエッジサーバーを世界中のオンプレミスにあるデータセンターにデプロイし、そのすべてを AWS 上のマスターサーバーに繋ぐ必要があります。つまりは、”クラウドネイティブ”な Perforce のマスターサーバーという位置づけになります。

この役割に最適なのが、パフォーマンス面で最も優れているエッジサーバーです。しかし、エッジサーバーでは、サイトローカルなバックアップや（AWS でマスター用に作成されるチェックポイントに加え）Perforce のチェックポイント作成が必要になります。また、マスターと同じように、問題が発生した際に迅速に復旧できるよう、エッジサーバーもサイトローカルな HA レプリカサーバーマシンと共にデプロイしておくことを推奨しています。

エッジサーバーの代わりに、標準的な転送レプリカや Perforce プロキシを使って、トポロジーを簡素化することもできます。転送レプリカやプロキシであれば、エッジサーバーで必要になるサイトローカルなバックアップ処理を削減したり、完全に無しにすることができます。ただ、ユーザー操作の観点では、エッジサーバーの方が全般的にパフォーマンスが優れているため、エッジサーバーを使ったアプローチが好まれています。

ハイブリッド型トポロジーにおいて、オンプレミスでエッジサーバーを使った場合と、転送レプリカまたはプロキシを使った場合で、Perforce Software のライセンス費用面での違いはありません。

AWS とオンプレミスの接続

AWS上にあるインフラとオンプレミスのトポロジーを繋ぎ合わせようと考えた場合、一番簡単なのは VPN の利用です。これで、Amazon VPC（仮想プライベートクラウド）が、オンプレミスのネットワークやデータセンターに繋がります。接続は、Amazon 側は仮想プライベートゲートウェイで、オンプレミス側はカスタマーゲートウェイで行われます。カスタマーゲートウェイは、物理的なデバイスかソフトウェアから選ぶことになります。Checkpoint、Cisco、Dell、Fortinet、Juniper、Palo Alto などのベンダーのデバイスについては、既に AWS 社でテスト済みです。また、数種類の Windows サーバーマシンでも、カスタマーゲートウェイが動作することが確認されています。

インターネット接続の帯域幅や起りうる遅延が、期待するスループット要件を満たさない場合は、オンプレミス環境からAWS 間の接続に専用回線を利用する ”Direct Connect”という種類の AWS サービスを利用することもできます。

AWS 単独型トポロジーについて

ゲームソフトウェア開発の高まるニーズに応えるためにインフラのアップグレードを検討中であったり、オンプレミスのサーバー管理に手間を割きたくないと考えているのであれば、AWS のみで完結するトポロジーがおすすめです。

原則、ユーザーは AWS のマスターサーバーに直接接続することになるため、（デスクトップ PC やノート PC、ワークステーションなどの）ユーザー端末以外のインフラは、オンプレミスでは必要がなくなります。

また、AWS 単独のインフラは「管理のしやすさ」という点でも優れています。すでにオンプレミスのインフラが候補となっている場合であっても、クラウドのみの構成でもベンチマークテストを実施してみる価値はあります。オンプレミスのインフラと AWS のみのインフラとの間でパフォーマンスを比較した場合、実は AWS 単独の方が高パフォーマンスなどという驚きの結果が出ることもあるからです。

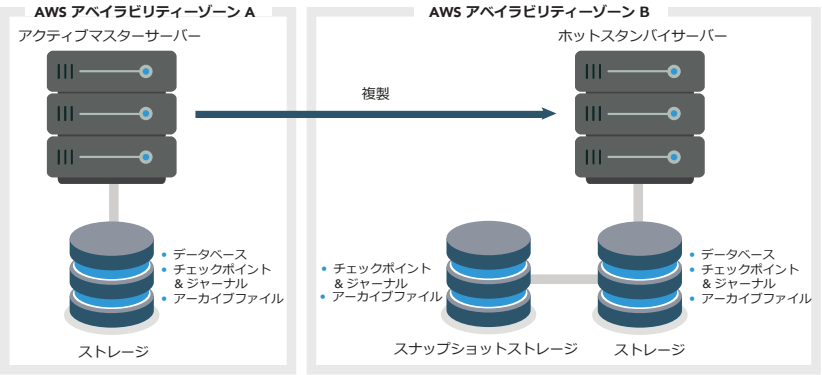


図 1: AWS 単独のトポロジー

AWS 単独のトポロジーであっても、エッジサーバーやスタンバイサーバー、転送レプリカ、プロキシが利用されることはあります。この点は、グローバル展開のオンプレミストポロジーと変わりません。

AWS 単体のトポロジーには、管理面やパフォーマンス面以外での強みもあります。例えば、セキュリティ対策や災害対策、各種 AWS プラットフォームサービス（DNS やディレクトリサービス等）との連携、CI/CD などの他のワークロードのサポート面でも優れています。ハイブリッド型と比較して、よりシンプルなキャッシュ方式であるため AWS データ転送料金を節約できる点や、AWS だけを扱えば済むという「シンプルさ」も魅力です。これら機能のデプロイは、AWS/Perforce であればさらに簡単で、支払いも「本当に使うもの」に対してのみに絞り込むことができます。

ビルドサーバー

ビルドサーバーは、AWS 構成に追加するワークロードとして非常に適しています。AWS のクラスタープレイスメントグループ内のマスターの隣にビルドサーバーを設置すれば、マスターとビルドサーバー間におけるネットワーク遅延を抑え、高ネットワークスループットを実現できることから、高速な CI/CD 環境の構築にも役立ちます。

定期的もしくは予想できない頻度で CI/CD リソースに対するニーズがあるのであれば、EC2 インスタンスやコンテナを使って、ビルドマシンを追加で立てることで、自動的なリソース調整が可能になります。 例えば、Jenkins にはビルドサーバーの負荷に応じて、インスタンスを自動的に起動し、トラフィックを流し込むことができるプラグインが存在します。そして、このインスタンスは、トラフィックが元のレベルに戻ったら、自動的に停止することもできます。

AWS 単独によるセキュリティ上の強み

AWS 単体のインフラ利用には、AWS IAM（ユーザーアクセスと暗号化キーの管理）やセキュリティグループ、NACL（ネットワークアクセス・コントロールリスト）を用いて、セキュリティを強化できるというメリットもあります。Helix Core に備わっているプロテクション・アーキテクチャと組み合わせることで、物理的なセキュリティから、ネットワーク、システム、アプリケーション、データレイヤー保護までをカバーする、強固かつ綿密な防御網を築くことができます。

- 詳細なアクセス管理
- 多要素認証
- HSM ベースの暗号キー保管
- サーバー側での暗号化　　など

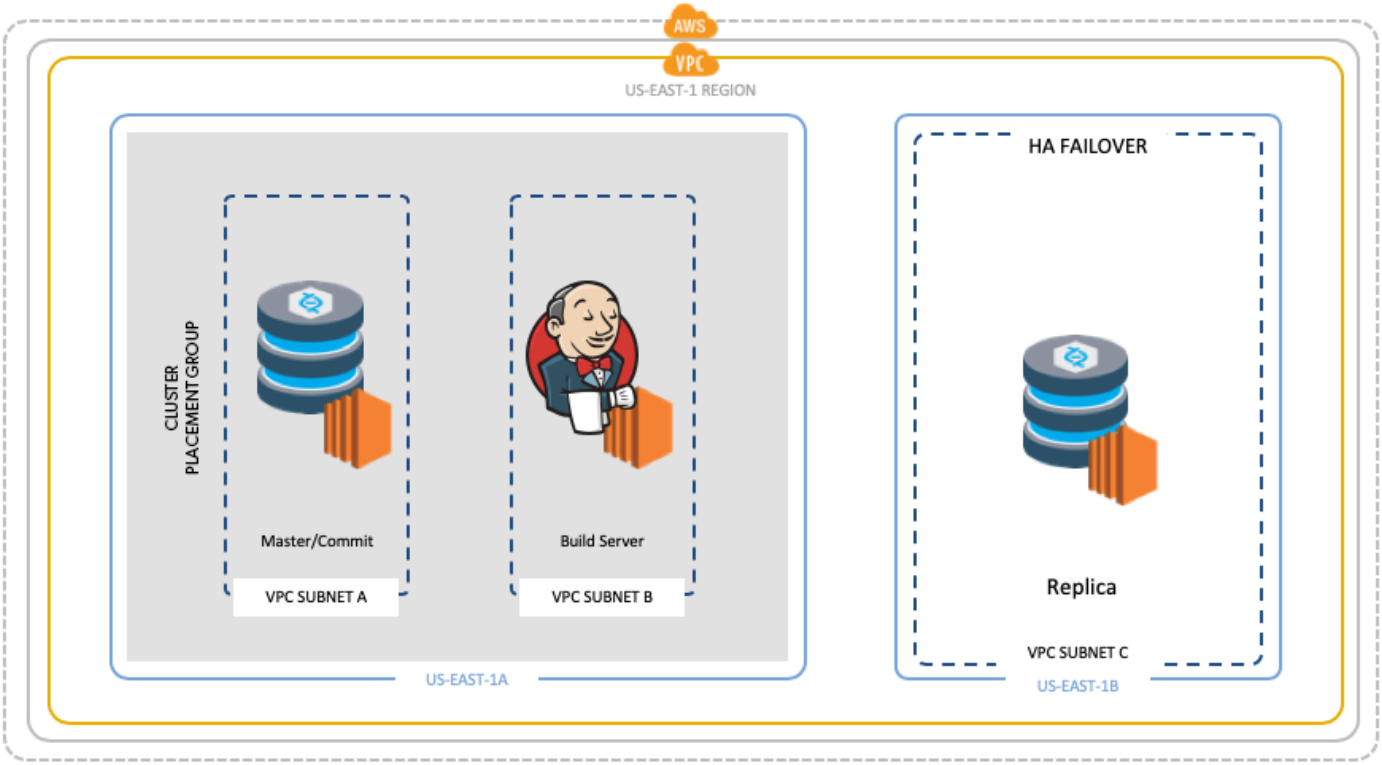


図 2: プレイスメントグループ内のビルドサーバー

多くの AWS サービスの実装と同様、これらのセキュリティ機能の実装は、同様の機能をオンプレミスのシステムで実現しようと考えた場合に比べると、費用を安く抑えることができるケースが多く、システム管理者の手間も少なく済みます。

トポロジーのモニタリング機能

AWS 単独のトポロジーには、モニタリングに役立つサービスが最初から組み込まれています。例えば、AWS CloudWatch はトポロジーの全体的な健康状態について、実用的な情報を提供してくれます。ログやメトリクス、イベントから AWS上で使用しているリソース、アプリケーション、サービスを確認することができます。さらには、オンプレミスで使用しているサービスのモニタリングにも利用可能です。CloudWatch は、定義した範囲外のメトリクスが出た際に（テキストやメールで）通知ができるSNS（Simple Notification Service）という、別の AWSサービスとも連携します。CloudWatch は、特定のインスタンスが応答不能になった場合に、リブートなどの修正サイクルや、サーバーのフェイルオーバープロセスの起動にも使うことができます。

ハイブリッド型 VS. AWS 単独型

総合的に見て、ハイブリッド型は「AWS 単独型トポロジー」+「Perforce フェデレーションアーキテクチャで実現する、インテリジェントなローカル・キャッシュ・オンプレミス」と考えるべきでしょう。AWS 単独のトポロジーと比べ、ハイブリッド型の方が AWS のデータ転送料金を大幅に抑えることができるのは確かです。けれども、パフォーマンスのベンチマークテストを実施した場合に、オンプレミスのハードウェアを利用することで、エンドユーザー側において AWS 単独と比べて大きなパフォーマンスの差が出るかと言うと、思ったほどの差は出ないことが予想されます。常に一定の品質を保ち、継続的な改善が行われているAWS インフラと比べると品質に大きな差が出るオンプレミスのハードウェアをはじめ、さまざまな要素が関わることで、この結果は変わってきます。

AWS をバックアップ目的のみで使う

現在、オンプレミスのインフラのみを利用しており、AWS の利用経験がほとんどない場合、今のやり方への影響を最小限に抑えた形でも AWS を使い始めることは可能です。

最初に、ディザスタリカバリ（DR）用の Helix Core レプリカを AWS 上に作ります。オンプレミスのインフラには、何も変更は加えません。まず、IT チームが AWS とはどのようなものなのか体験できる場を作るのです。ユーザーにはまったく関係のない場所での作業ですので、何の影響も生じません。

DR レプリカをさらに活用するには、S3 のマルチ AZ の耐久性が最初から保証される、EBS スナップショットを利用する必要があります。クロスリージョン S3 レプリケーションで、効果を拡大することもできます。どちらの方法でも、大切なアセットを今まで以上に確実に保護できるようになります。

S3 でライフサイクルポリシーを設定することで、低コストでバックアップやログファイルを"永久的"に保存しておくこともできます。 各種 SLA の条件や（データ取得に求める頻度と速度に応じた）価格の異なるストレージオプションが用意されています。永久的な保存は必要ないのであれば、ポリシーの設定で、削除のタイミングを決めることも可能です。

テストや実験に使う

AWS の利用を開始する良い方法として、新しいインフラツールやワークフロー、ゲームエンジンや外部コードのテスト環境として、追加のレプリカをいくつか AWS 上にデプロイしてみるという手もあります。

ただ、この方法にも良い点はあるのですが、さらにクラウド移行を進めない限り十分に得ることができないメリットが存在していることも事実です。なお、Helix Core のクラウドデプロイメントでも、この方法で実現できる範囲であれば、同じ様に実現可能であることは確認済みです。

Perforce Consulting サービスの専門スタッフは、AWS の世界へと皆様が思い切ってさらに踏み出すことができるよう、お手伝いをしています。

エッジサーバーの導入

すでに Helix Core のエッジサーバーをご利用の方にとって、以下の説明はすでに分かりきった話だと思います。しかし、ハイブリッド型トポロジーでのクラウド移行を検討されている方の中には、Helix Core のエッジサーバーを初めて使うことになる方もいるかもしれません。

基本的に、新しいエッジサーバーをデプロイする際に考えなくてはならないのは、トポロジー内での最初のエッジサーバーの導入をどうするかです。転送レプリカやプロキシの時とは違い、ユーザーに関係のないところでできる作業ではありません。ユーザーには、現在使っているワークスペースから、エッジサーバーに繋がる新しいワークスペースへの引越しを一度にまとめて行ってもらう必要があるからです。通常、退去するワークスペース内の使用中ファイルを保留する、サブミットする、または編集前の状態に戻すという作業が発生します。保留中にしたファイルはエッジサーバー上の新しいワークスペースで再度開くことができます。
この"引越し"作業には、エンドユーザーとの調整が必要になるため、ユーザー権限のやり取りが成功のカギを握ります。

その他のエコシステムについて

本テクニカルガイドでは、（Active Directory や LDAP のような）ID 管理システム、ワークフロー管理、不具合追跡システム、CI/CD ソリューションなどの、Helix Core エコシステムの他の構成要素については触れないこととします。

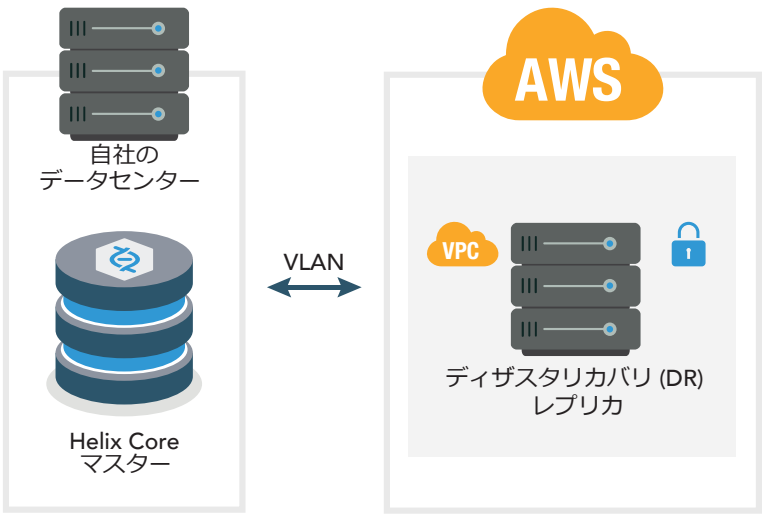


図 3: AWS をバックアップ目的でのみ使用するケース

クラウドへの移行を行う際には、エコシステムの構成要素それぞれへの影響を考慮する必要があります。ただ、今までに多くのクラウド移行を行った経験から言わせてもらうと、その影響は大抵の場合は小さなものです。よくあるケースは、AWS のマスターサーバーがオンプレミスの AD サーバーから認証を行えるように、もともと厳密に設定されていたネットワークファイアウォールルールの緩和が必要になる等です。AWS には、Microsoft の Active Directory をベースとする、AWS ディレクトリサービスというサービスが存在しており、オンプレミスに Microsoft Active Directory を実装していれば、その各種機能とも互換性を持ち、オンプレミスのサーバーへのリンクがダウンしたとしても、ユーザー認証を行うことができるようになっています。

ハイブリッド型クラウドトポロジーの例

本テクニカルガイドで示すトポロジーの例では、AWS のプライマリ・リージョン内（一般的に、ユーザーが最も多く集中する場所の近く）にマスターサーバーを設置しています。Helix Core の "スタンバイ" レプリカは、高可用性とリアルタイムでのデータ保護を実現するため、同 AWS リージョン内の別のアベイラビリティーゾーンに構築されています。

（ローカルのデータセンターがある場合）ユーザーは、データセンター内にオンプレミスでデプロイされたエッジサーバーを介して、Helix Core にアクセスします。エッジサーバーは、いざという時のフェイルオーバーのために、それぞれがサイトローカルなスタンバイレプリカサーバーを備えることで、同サイトにおける高可用性を確保しています。

開発チームが世界中に分散して存在しているようなケースでは、"転送レプリカ" も AWS内のユーザーが多く集中している場所に近接するリージョンに追加しておくのがいいでしょう。前述の AWS Global Accelerator サービスは、ネットワークの観点から、このような構成を簡単にしてくれます。オンプレミスのエッジサーバーは、最も近いリージョンにある転送レプリカまたはマスターの内、近い方に接続されます。このアプローチは、大陸間であったり、海を超えるような長距離でのデータ移動が必要になるケースにおいて、AWS インフラのパフォーマンスを向上させ、パブリックネットワークを使った場合に比べてコストを抑えることを目指したものになっています。

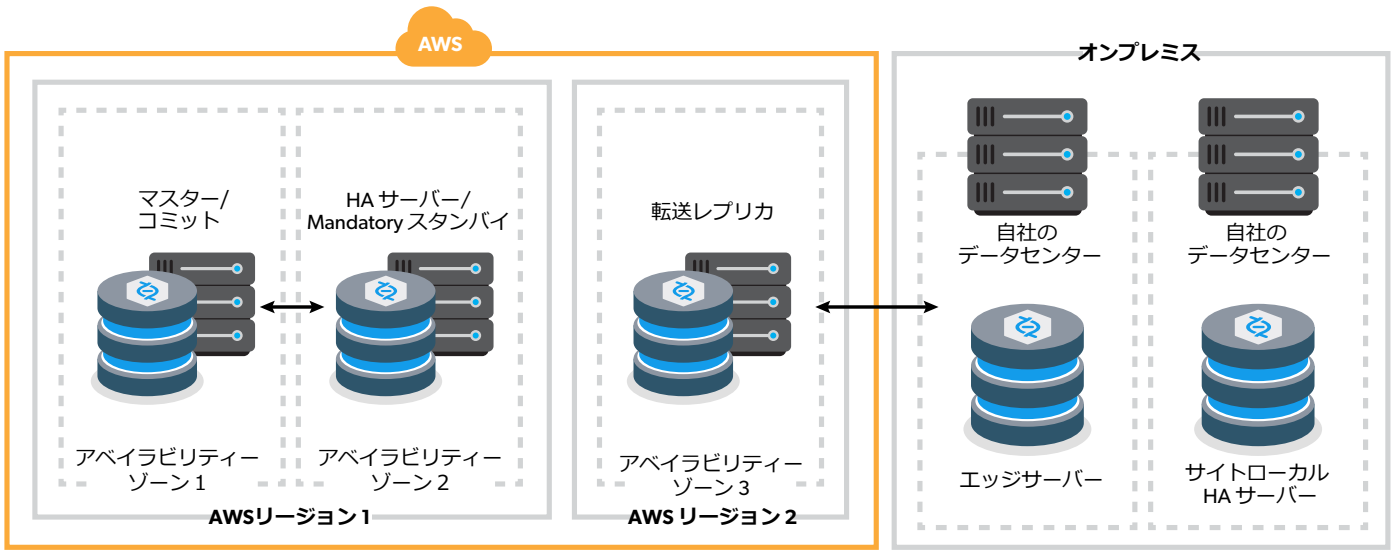


図 4: ハイブリッドトポロジーの例

次のページから、トポロジー例についてもう少し詳細に説明します。この例では、ハイブリッド型クラウドアプローチを用いており、オンプレミスのインフラを活用することで、AWS のデータ転送料金を最小限に抑えつつ、マスターサーバーにおけるデータの安全性と可用性を最大限まで高めています。

右の表に示すのは、ホスト名と Perforce ServerIDです。
注:

- ホストの命名規則（p4-から始まる）から、ホストは Helix Core インフラ専用になっていることは分かりますが、その役割までは分からないようになっています。これは、役割は変更される可能性があり、フェイルオーバー時には逆転する可能性もあるからです。また、ホスト名には末尾に数字が付きます。
- Helix Core ServerID は、マシン上で Helix Core サービスが現在担っている役割を定義しています。（本テクニカルガイドのトポロジー例では、データセットはひとつしか存在しないものと仮定します）。
- ServerID の値は、SDP の mkrep.sh スクリプトで決められるルールに従っており、変更は推奨されません。
- ホストの命名規則は、サンプルのベストプラクティスとして記載していますが、各社で決めている命名規則を適用しても構いません。ただし、フェイルオーバーの際に余計な混乱が生じないように、ホスト名は個別の役割と紐づけない方が良いとされています。
- エンドユーザーは、サーバーのホスト名を知る必要はありません。代わりに、ホスト名から末尾のダッシュと数字を外した、ホストのエイリアス（p4awsnva や p4syd）を使うことになります。

EC2 ホスト名	ServerID	説明
p4-awsnva-01	master.1	マスター/コミットサーバー。.1 は、SDP インスタンス名で、データセット識別子。
p4-awsnva-02	p4d-ha-awsnva	マスター向けの Mandatory スタンバイレプリカとして構築された、高可用性 (HA) サーバー。
p4-awssyd-01	p4d-fr-awssyd	オーストラリアのシドニーに設置された、マスター向け転送レプリカ。
p4-syd-01	p4d-edge-syd	オーストラリアのシドニーにオンプレミスで設置されたエッジサーバー、AWS 上の転送レプリカ向け。
p4-syd-02	p4d-fs-edge-syd	シドニーのエッジサーバー向けにスタンバイサーバーとして構築された、サイトローカルな HA サーバー。

Helix Core HA レプリカについて

Helix Core のマスター/コミットサーバーは、p4-awsnva-01※1（-01は、任意の整数）という名前のホスト上に設置されます。このホスト名は、ハードウェアが新しい OS や新しいインスタンスタイプに更新されたり、本体が交換されたりすることで、時々変更になることがあります。

マスター/コミットサーバーは、専用の高可用性（HA）サーバーを持ちます。本テクニカルガイドでは、HA レプリカを以下の様に定義します。

- マスターサーバーと同じ AWS リージョン内に存在する。
- 割り当てられているマスターと同じ AWS リージョン内の別のアベイラビリティーゾーン内に存在する。
- サーバースペックの命名規則に従った、**サーバースペック**名が付けられている。例えば、アメリカのノースバージニア州にある us-east-1 リージョンの HA サーバーであれば、p4d_ha_awsnva となる。
- サーバースペックの Services: フィールドの値は、standby である。
- サーバースペックの Options: フィールドの値は、mandatory である。
- Helix Core (P4D) のバージョンが、standby タイプと forwarding-standby タイプのレプリカを使う上で必要になる機能を備え、p4 failover コマンドをサポートしている 2018.2 以降になっている。
- db.replication 設定が readonly である。
- lbr.replication 設定が readonly あり、メタデータとアーカイブの完全なレプリカの状態になっている。
- rpl.journalcopy.location 設定が 1 で、ジャーナルストレージが最適化されている。
- フィルタは何もかかっていない状態。レプリカで設定された "pull" の開始コマンドで '-T' フラグの使用がなく、サーバースペックで 各種 *DataFilter 値の使用もない状態。

Helix Core DR レプリカについて

HA 用のソリューションだけでなく、ディザスタリカバリ用のソリューションも用意しておく必要があります。本テクニカルガイドのトポロジーの例においては、シドニーの AWSデータセンターにある転送レプリカが DR 機能を持ち、長距離 WAN トラフィックが最適な経路を通れるように AWS インフラを使って管理しています。

Server Deployment Package

Perforce Server Deployment Package (SDP) は、Perforce Software 社が提供するオープンソースソフトウェアです。さまざまな Helix Core トポロジーのコンポーネント管理に使うことができ、AWS 上の EC2 インスタンス上にデプロイされたものから、オンプレミスにデプロイされたものまで広くサポートします。

SDP は、Helix Core と共に進化を続けており、（ゼロダウンタイムチェックポイント等の）ルーティンなメンテナンス作業から、パフォーマンスの最適化をはじめとする多種多様なベストプラクティスの実装についての最新情報を提供します。

本テクニカルガイドでは、SDP の最適なストレージレイアウトについて理解していただきたいと思います。SDP には、（OS のルートボリュームの他に）以下の 3 つのストレージボリュームが Helix Core 用に用意されています。

- /hxdepots ― アーカイブファイルのコンテンツに加え、番号付きジャーナルやチェックポイントを格納。データ量が膨大になる可能性あり。データ量の多い readや write アセスが多い。バックアップと暗号化が必須。
- /hxmetadata ― メタデータのデータベースのみを格納。ランダム I/O パターンでのアクセスを受けることが多い。直接的な バックアップは不可、書き込みは常に行われる。
- /hxlogs ― ライブジャーナルとサーバーログ、その他のログを格納。バックアップは不要。

EBS ストレージの設定

EC2 インスタンスの設定を行う前に、EBS※2ストレージボリュームを後述の表に記すとおりに作成・設定する必要があります。これらストレージボリュームは、マスターと HA レプリカの EC2 インスタンスの両方に必要になります。

注：下記の説明において gp2 は AWS の汎用 SSD ストレージを、 st1 は AWS のスループット最適化 HD ストレージを意味します。理由は以下のとおりです。

- gp2 のパフォーマンス向上に最も効果がある（例：メタデータボリュームで求められる遅延レベルを実現できる等）ところで使用。
- gp2 はログボリュームで使用。中のファイルは頻繁に rotate されるため、サイズはさほど大きくならない。
- パフォーマンス的に問題が無いところ（つまりは、デポボリュームで多く発生する、データ量の多い readや write 操作）においては、より安価な st1 ボリュームを活用。

ストレージのベースサイズは、プロダクション・ゲーム開発サーバー用として見積もったものであり、実サイズとは異なります。実サイズは、もっと大きくなる可能性がかあり、特に、/hxdepots ボリュームのサイズは、開発するソフトウェアのスケールによっては、簡単に数テラバイトを超える可能性もあります。

オンプレミスのエッジサーバー用ストレージの設定

オンプレミスのエッジサーバーマシン（つまり、エッジサーバーとサイトローカルな HA サーバー）用のストレージを設定する場合は、次ページに記載の Amazon EBS 設定にできる限り近い設定を使うようにしてください。

EC2 インスタンスの設定

Helix Core サーバー用の EC2 インスタンスを作成する際には、以下の情報が役に立つでしょう。ヘッダーは、AWS コンソールの EC2 セクションで ”インスタンスを起動” を選択した際に表示されるものと、完全一致ではないことはご承知ください。

最初のインスタンスを作成しさえすれば、後のインスタンス作成は簡単になります。AWS コンソールから既存のインスタンス（例：p4-awsnva-01 ホスト）を選んで、テンプレートとして使用し、[Launch more like this] アクションでの追加ができます。

注：インスタンス起動前には、VPC およびにセキュリティグループの作成が必要です。

EBS ネームタグ	マウント	ベースサイズ	EBS ストレージ注記
p4-awsnva-01-hxdepots	/hxdepots	1TB	st1、暗号化あり
p4-awsnva-01-hxlogs	/hxlogs	128G	gp2、暗号化なし
p4-awsnva-01-hxmetadata	/hxmetadata	64G	gp2、暗号化なし
p4-awsnva-02-hxdepots	/hxdepots	1TB	st1、暗号化あり
p4-awsnva-02-hxlogs	/hxlogs	128G	gp2、暗号化なし
p4-awsnva-02d-hxmetadata	/hxmetadata	64G	gp2、暗号化なし

AMI の選択：AMAZON Linux AMI を選ぶ

最新の Amazon Linux AMI を選んで使ってください。Amazon Linux AMI は、AWS インフラを最大限に活用できるように調整されています。選択できる Amazon EC2 のインスタンスタイプも多く、RHEL/CentOS の標準的な管理方法で容易にメンテナンスできるようになっています。

インスタンスタイプの選択： C5 を選ぶ

Amazon EC2 のインスタンスタイプを選ぶ際は、以下を考慮してください。

- **コンピュート最適化**されたインスタンス（**c5**インスタンスタイプ）の中から選択。メモリ最適化インスタンスでも問題なく動作はするかもしれませんが、バランス的に考えて、コンピュート最適化されたものの方が、ゲーム開発における Helix Core パフォーマンスが全体的に良くなると期待できます。また、高速なプロセッサを使うことで、大きなデジタルアセットを扱う際の大量の圧縮/解凍処理による作業の遅延を回避しやすくなります。
- EBS 最適化済みのインスタンスタイプのみを選択。
- ”ネットワークパフォーマンス” が ”**10 ギガバイトまで**” またはそれ以上のインスタンスタイプのみを選択。
- 十分な RAM のあるインスタンスを選択。例えば、合計アセットサイズが 2 TB 程度になるゲームを開発している場合、32 GB の RAM が使える **c5.4xlarge** インスタンスタイプをお勧めしています。

インスタンスのオプション設定

インスタンスのオプション設定を行う場合、Helix Core VPC と紐づけて考える必要があります。マスターと HA サーバーが、同じリージョン内の別のアベイラビリティーゾーン（サブセット）に存在するように、適切なサブセットを選んでください。

以下の設定は無視します。

- プレイスメントグループ [Placement groups]
- キャパシティー予約 [Capacity reservation]

下記のオプションにはチェックを入れます。

- 意図しない停止に対する保護 [Protect against accidental termination]
- CloudWatch の詳細モニタリングを有効 [Enable CloudWatch detailed monitoring]

テナント属性については、共有を推奨します。"dedicated" の利用お勧めしません。この属性は、厳密な規定に基づいたデータ分離が求められるワークロードに対して適用されるもので、基本的に、ソフトウェアゲーム開発向けではありません。AWS テナント属性における ”dedicated” という言葉は、高パフォーマンスを意味するものではありません。本件については、AWS ユーザーが実施した詳細なベンチマークテストでも裏付けられています。

以下、追加情報です。

- CPU負荷においては僅かな差はあるものの、ほとんど無視できる範囲です。コンピュート最適化タイプのインスタンスを選択していれば、完全に吸収できる程度の差です。
- ネットワークのパフォーマンスや遅延についての影響はありません。
- メモリアクセスに関連する影響も報告されていません。

最後になりますが、Helix Core を Elastic Network Interface と一緒に使うことは避けてください。

ストレージのオプション設定

インスタンス作成時には、OSを持つ必要のあるインスタンスルートボリュームだけ作成する必要があります。その他の、Helix Core 関連の /hx* ボリューム用の EBS ストレージボリュームについては、上記に記載がある通り、別途作成します。

タグの追加

対象マシンのホスト名に対応したネームタグ（例えば、p4-awsnval-01）を追加します。

リザーブドインスタンス

EC2 インスタンス作成時の標準オプションは、オンデマンド設定で、好きな時にインスタンスの開始、停止、終了ができるようになっています。構成が確定し、どのサーバーを長期的に使うのか決まった時点で、オンデマンドのインスタンスは、リザーブドインスタンスに切り替えることをお勧めします。リザーブドインスタンスの場合、料金は前払いで期間契約となり、同じ構成でのオンデマンド料金に比べ、利用料金を大幅に下げることができます。

アベイラビリティーゾーン

アベイラビリティーゾーンとは、リージョン内の分離された場所を指します。各リージョンは、複数のアベイラビリティーゾーンで構成されていますが、各アベイラビリティーゾーンが複数のリージョンに属することはありません。AZ (アベイラビリティーゾーンの AWS の専門家内での通称) は、それぞれが個々に分離された状態で存在しますが、低遅延のリンクで繋がっています。

冗長構成を最適化するためには、マスターとスタンバイサーバーは、別々の EC2 アベイラビリティーゾーンにある必要があります。

仮想プライベートクラウド（VPC）と Perforce ライセンスファイル

Perforce コンポーネントでの使用に限る、仮想プライベートクラウドを、名前に perforce と入れたタグを付けて定義します。

そして、Perforce ライセンスファイルをリクエストする際には、AWS インスタンスの"プライベート" IP アドレスを Perforce sales (sales@perforce.com) にご連絡ください。

Server Deployment Package (SDP)

サーバーストレージとデポの標準仕様

すでに SDP を使っている Perforce ユーザーに関しては、ストレージの標準仕様を満たしているため、調整作業などは必要ありません。AWSへの移行を行う場合は、重要なデータがきちんとバックアップされ、正しい暗号化もされた状態で EBS ボリューム上にすべて存在し、コスト面においても最適なストレージソリューションになっているようにするためには、以下の基準を満たしておく必要があります。

- server.depot.root は、SDP の標準仕様に従って設定（例： /hxdepots ボリューム上の/p4/1/depots の値を使用）。
- デポの Map: フィールドには、デフォルトの DepotName/... 以外は設定しない。
- AWS 環境内およびにオンプレミス環境にあるエッジサーバーに対して、サーバーマシンは必ず、server.depot. root で参照される、デポに対する論理ストレージボリュームをひとつだけ持つように設定しなくてはならない。また、このボリュームは、すべてのアーカイブファイルを格納できる程度に、大きいものである必要がある。必要に応じて、拡張できるものが理想的。

AWS 上では、EBS ストレージのサイズは任意の大きさに調整可能なため、上記の最後の点については気にする必要がありません。ここで気にすべきは、ハードウェア面での物理的な制限がある場合にありがちな、オンプレミスのエッジサーバーがデポ単位でのシンボリックリンクで特異な分離状態に陥ってしまうのを避けることです。最初は大きな問題にはならなそうだとしても、そのような特異な分離は後々になって、余計な複雑さを持ち込み、何らかの不具合が発生して、復旧が必要になった時に問題を引き起こしがちです。デポが、間違ったボリュームに保存されてしまい、バックアップが行われなかったり、P4ROOT の貴重なディスクスペースが奪われてしまうことにもなりかねません。

Helix Core の server. depot.root は、常に（SDP の標準仕様にあるように） /p4/N/depots に設定されている必要があります。また、そのように設定されたデポは、すべて常に /p4/N/depots/DepotName パス（N には SDP インスタンス名が入る）経由でアクセス可能な状態になっている必要があります。

SDP を使った複製

エッジサーバーとレプリカは、SDP の mkrep.sh スクリプトを使ってセットアップされます。このスクリプトによって、レプリカは、タイプごとに適切なベストプラクティスの設定で作られるようになっています。

mkrep.sh を使う前に、/p4/common/config/SiteTags. cfg. ファイルで地理的なサイトタグの設定が必要になります。

各レプリカの P4TARGET の値が、フェイルオーバーがあっても生き残るシンボリックエイリアスに設定されていることを確認してください。エンドユーザーが、フェイルオーバーがあっても無くなることのないホストエイリアスを使って、ローカルのプライマリ・エッジサーバーやマスターサーバーを参照すべきであるように、レプリカも同じようにすべきなのです。そのため、シドニーにあるエッジサーバーの P4TARGET 値であれば、（p4awsnva-**01**:1666 や p4awsnva-**02**:1666 ではなく）p4awsnva:1666 のようになります。フェイルオーバーの際には、p4awsnva エイリアスは -01 から -02 ボックスにリダイレクトされます。

SDP を使ったエッジサーバーのセットアップ

以下のサンプルコマンドは説明用です。実際のデプロイメント時には、実際のコマンドやサイトごとに異なる操作が必要になります。

本テクニカルガイドで例として挙げているトポロジーでは、SDP はインストール済みで、なおかつ SDP インスタンス名は（デフォルトの最初のインスタンス）1 に設定されているとします。この場合のエッジサーバーは、下記のコマンドで設定され、**perforce@p4-awsnva-01** として動作します。

```
cd /p4/common/bin
./mkrep.sh -i 1 -t edge -s syd -r p4-
awssyd-01
```

そして、そのエッジサーバーのサイトローカルな HA レプリカコマンドは以下の様になります。

```
./mkrep.sh -i 1 -t fa -s syd -r p4-awssyd-02
```

これらのコマンドは両方とも、他の設定に先立って、マスターサーバーから実行されます。

次に、エッジサーバの基になるチェックポイントは、以下の様なコマンドで作られます。

```
./edge_dump.sh 1 p4d_edge_syd
```

/p4/1/checkpoints の中に、エッジサーバーの基になるチェックポイントが作られたら、対象のエッジサーバーに移動させます。インスタンス用に SDP の設定も必要になります。そして、以下のコマンドを **perforce@p4-syd-01** として実行することで、エッジサーバーの基になるチェックポイントをリストアします。

```
./recover_edge.sh 1
/p4/1/checkpoints/p4_1.edge_syd.seed.
ckp.1234.gz
```

コマンド内の 1234 は、上記で生成されたエッジサーバーの基になるチェックポイントのチェックポイント番号に置き換えてください。

次に、生成されたエッジサーバーの基になるチェックポイントを、エッジサーバーと、そのサイトローカルなレプリカに移動させて、それぞれのエッジサーバー上でチェックポイントをリストアします。

AWS EC2 スナップショット用に SDP を調整

AWS 上にデプロイした際に、SDPの backup_functions. sh には、少し調整を加える必要があります。これは、/hxdepots ボリュームの EC2 スナップショットが、最適なタイミングで（Helix Core のチェックポイント処理の完了直後に）実施されるようにするためです。これにより、復旧用のデジタルアセットが作成されてから、より安全な場所へと退避させられるまでの間の時間をできる限り短くすることができるのです。

この実現には、aws ec2 create-snapshot 呼び出しを行う、AWS　コマンドラインツールを使う必要があります。呼び出しは、例えばですが、以下の様になります。

perforce@p4-awsnval-01:/home/perforce **aws ec2 create-snapshot --description “Backup of /hxdepots on $(date).” --volume-id vol-aabb6c5e5a1c6cd8c**

この例では、 volume-id の部分は、任意の EC2 インスタンス上に /hxdepots ボリュームとしてマウントされている EBS ボリュームのボリューム ID が入ります。同様の考え方で、ルートボリュームもスナップショットする必要があります。/hxmetadata ボリュームのスナップショットは不要です。アクティブな P4JOURNAL ファイル内にある最も重要なデータは複製されているため、/hxlogs ボリュームのスナップショットは通常はとられません。サーバーログなど、それ以外のデータは、より一時的な扱いになります。

作成した EC2 スナップショットは、まずは Amazon S3 バケットへ格納し、その後、コスト削減のために Amazon Glacier に移します。前述した S3ライフサイクルポリシーによって、「いつ」「ストレージのどのTierへ」アセットを移動させるかは自動化できるようになっています。

パフォーマンス、セキュリティ、安全性に関連する設定などの微調整

SDP には、パフォーマンスやセキュリティ、データの安全性など、さまざまな設定を最適な値に調整できるようにサポートするスクリプトも含まれています。このスクリプトは、新規の SDP インスタンス上で実行される前提で作られてはいますが、既存のデータセットの設定においても参考にすることができます。

各設定については、p4 configure show *SettingName* コマンドを使って確認します。それぞれの設定値は、必要に応じて、右の表を参考に、p4 configure set *SettingName SettingValue* を使い、調整してください。

最新情報については、SDP 内の Best Practices Settings Guidance を参照してください。

Helix 管理システム (HMS)

Helix 管理システムは、SDP の最新版と共に提供され、高度なグローバル・ハイブリッドトポロジーの管理には欠かせない機能も備えています。

SDP と拡張機能をしっかりと管理

p4hms.p4demo.com という Bastion（踏み台）ホスト上で動く、小さな Helix Core 用インスタンスが存在します。その SDP インスタンス名が、hms です。Helix Core (P4D) マスターサーバーをはじめ、レプリカ、エッジサーバー、プロキシ、ブローカー、グラフィックツール用の Perforce プラグイン（P4GT）などの、Helix Core トポロジーコンテンツが存在する全てのホスト上にある SDP は、このインスタンスによって管理されています。

設定名	推奨値	カテゴリー
run.users.authorize	1	セキュリティ
filesys.P4ROOT.min	5G	安全性
filesys.depot.min	5G	安全性
filesys.P4JOURNAL.min	5G	安全性
server	4	ログ
monitor	1 (または 2)	モニタリング
db.reorg.disable	1	パフォーマンス
net.tcpsize	0	パフォーマンス
net.autotune	1	パフォーマンス
db.monitor.shared	4096	パフォーマンス
net.backlog	2048	パフォーマンス
lbr.autocompress	1	安全性 パフォーマンス
lbr.bufsize	1M	パフォーマンス
filesys.bufsize	1M	パフォーマンス
server.start.unlicensed	1	ライセンス
rejectList	P4EXP,version=2014.2, Operating System	セキュリティ/サイバー防御、安全性
server.global.client.views	1	エッジ機能性
server.locks.global	1	エッジ機能性
auth.id	p4_SDPInstanceName	機能性
rpl.forward.login	1	機能性
dm.shelve.promote	1	Swarm
dm.keys.hide	2	Swarm
filetype.bypasslock	1	Swarm

Helix Core によるファイルのデプロイや検証で、SDP スクリプトと設定ファイルは、すべてのホスト上で、きちんと最新の状態かつ同期のとれた状態で管理されます。

HELIX トポロジーの定義

すべての SDP インスタンスとホストは、ひとつの Helix トポロジー設定ファイルで定義されています。また、どのトポロジーコンポーネントが、どのホスト上で ”通常” 時に ”post-failover” モードで動いているのかもこのファイルで分かるようになっています。詳細は、この Sample Helix Topology file を参照してください。

一元管理

Helix トポロジーファイルは、SDP インスタンス名と（Helix トポロジー設定ファイルで定義される）コンポーネント名を組み合わせる形で、SDP インスタンスに関連するコンポーネント名を決めるという命名規則を持っています。これにより、以下のようなコマンドを使うことで HMS サーバーから、どのインスタンスの P4D （または、他のコンポーネント）であっても、停止、開始、ステータスのチェックが可能になります。

```
hms status 1:master
hms stop 1:p4d_edge_syd
```

フェイルオーバー策の定義

HMS では、フェイルオーバーの際の対応について、計画や準備を事前に定義しておくことを強く求めています。万が一、フェイルオーバーが必要になった場合も、定義した計画通りに迅速に実施できるからです。計画の中では、起こりうるさまざまな故障のシナリオ（最もよくあるのは、マスターやエッジサーバーマシンの故障）からの復旧について考えておく必要があります。

フェイルオーバーが必要であると判断する状況別に、次のようなフェイルオーバー策が選択肢として挙がってきます。

シンプルなフェイルオーバーの実行

ローカルにあるオフライン DBS へのフェイルオーバー

”Local” と名前に入っているフェイルオーバー策では、指定のサーバーマシン上に存在する、使用中の（破損してしまったと思われる）データベースが、スペアのデータベースに置き換えられます。なお、このスペアデータベースは、offline_ db フォルダ内の同じホスト上にある SDP によって管理されています。このローカル・フェイルオーバー策を採用した場合は、ユーザーやネットワークトラフィックのリダイレクトのための変更作業は発生しません。

このフェイルオーバーは、例えば下記の様なコマンドで実行が可能です。

```
hms failover local i:1 u
```

AWS マスターの HA フェイルオーバー

”HA” と名前に入っているフェイルオーバー策では、使用中かつリアルタイムに複製が作られている、データベースやアーカイブファイルを持つスタンバイサーバーが、新しいマスターとして格上げされます。この HA フェイルオーバー策を採用した場合は、ユーザーやトラフィックのリダイレクトが必要になります。

このフェイルオーバーは、例えば下記のようなコマンドで実行が可能です。

```
hms failover ha i:1 u
```

上記のコマンドの裏では、hms スクリプトがフェイルオーバーマシンをマスターマシンへ格上げするのに必要となる Perforce コマンド（例えば、p4 failover コマンド）を実行します。

フェイルオーバー策	ホストの切り替え	使用例
Master Local	p4awsnva-01 (切り替えなし)	p4awsnva-01　ホストは問題ないが、データベースが（例えば、突然の停電や人的管理エラーによって）壊れてしまった場合に使用。データベースを同ホスト上で復旧。
Master HA	p4awsnva-01→ p4awsnva-02	p4awsnva-01 ホストが使用不可能または（EC2 インスタンスの RAM 更新のため等で）オフラインになる必要がある場合に使用。スタンバイレプリカで復旧。
Sydney Edge Local	p4-syd-01 (切り替えなし)	p4-syd-01 ホストは問題ないが、データベースが（例えば、突然の停電や人的管理エラーによって）壊れてしまった場合に使用。データベースを同ホスト上で復旧。
Sydney Edge HA	p4-syd-01→ p4-syd-02	p4-syd-01 ホストが使用不可能または（RAM更新のため等で）オフラインになる必要がある場合に使用。サイトローカルなスタンバイレプリカで復旧。

フェイルオーバーを完了させるためには、hms スクリプトの実行後に、トラフィックの通り道が p4awsnva-01 の代わりに p4awsnva-02 になるように、社内ネットワーク DNS で p4awsnva ホストエイリアスを変更する必要があります。

AWS 上の HA サーバーは *mandatory* のスタンバイレプリカとして設定されているため、グローバルダウンストリームのエッジサーバーやレプリカは、フェイルオーバー後に作られた複製データで、故障発生前の作業を続けることができます。

SYDNEY EDGE の HA フェイルオーバー

Sydney edge の HA フェイルオーバーは、例えば以下の様なコマンドで実行することができます。

```
hms failover syd-edge-ha i:1 u
```

このコマンドの実行後に、社内ネットワーク DNS を変更して、今まで Sydney edge サーバーを参照していた p4syd DNS エイリアスを p4- syd-01 から p4d-syd-02 に変える必要があります。

AWS セキュリティグループの設定

AWS セキュリティグループを使う場合には、ネットワークファイアウォールのルールの検討が必要になります。まず、セキュリティグループは、Perforce Helix の名前のタグを付けて作成してください。また、有効にする際には、下記に示すように、ポートを開いておく必要があります。

もしも、コードレビューやレポジトリの閲覧に Helix Swarm を使っている場合は、「HTTPS では 443 番ポートを使う」といったルールを Perforce Helix セキュリティグループに追加してください。

この後に示す例では、p4d プロセスは 1999 番ポートで、p4broker プロセスは 1666 番ポート上で動きます。p4broker プロセスの使用は必須ではなく、使用しない場合は、p4d プロセスが代わりに 1666 番ポートを使用することになります。ブローカーを用いることで、Helix Core の管理者は、さまざまな機能が使えるようになりますが、同時に管理や更新しなくてならないトポロジーコンテンツが増えることにもなります。

/hxdepots ボリューム（任意）に EFS を使用する場合は、EFS と名前を付け、ソースをセキュリティグループ自体の ID とした NFS インバウンドルールを追加してください。

HMS ハブ・アンド・スポーク

HMS サーバーとは、それひとつで他すべてをコントロールする Helix Core サーバーを指します。例えば、バックアップスクリプトやカスタムトリガースクリプト、さらには Helix Core 関連マシン上の全インスタンス用の crontab コマンドまで、AWS の内/外に関係なく、まとめて管理しています。そのため、アクセス制御の観点から、このサーバーは bastion ホストとして、最高レベルのセキュリティで構築しておく必要があります。オンプレミスにデプロイすることもできますが、可能であれば、AWS上にゲーム開発データ用のサーバーインスタンスとは別の EC2 インスタンスとして、デプロイするのが理想的です。

HMS サーバー (p4hms.p4demo.com) は、以下のようなトラフィックを許可する必要があります。

- HMS が管理する全 p4d サーバーからの、7467番 ポートを通るインバウンドトラフィック（常時 SSL 暗号化通信）
- HMS が管理する全 p4d サーバーからの、7468番 ポートを通るインバウンドトラフィック（常時 SSL 暗号化通信）

オンプレミスエッジサーバー向けインバウンドルール

オンプレミスのエッジサーバーは、以下のポートを通るインバウンドアクセスを必要とします。

- 1666 番ポートを通る社内ネットワークからのアクセス
- 1999 番ポートを通る社内ネットワークからのアクセス
- 22 番ポート（SSH）を通る HMS サーバー（Linux bastion ホスト）からのアクセス

AWS サーバー向けインバウンドルール

AWS サーバーは、以下のインバウンドアクセスを必要とします。

- VA1 エッジサーバーから 1999 番 ポート（SSL 暗号化任意）上の VPC へのアクセス
- HMS サーバーから 1666 番ポート（SSL 暗号化任意）上の VPC へのアクセス
- HMS サーバーと Linux bastion ホストから 22 番ポート（SSH）へのアクセス

全サーバー向けアウトバウンドルール

アウトバウンドルールに関しては、各社のポリシー従い、制限の有無を決めることができます。制限をかけるのであれば、セキュリティと管理のしやすさのバランスをとるためにも、以下のガイドラインの参照をお勧めします。

Host and Port Access List for Perforce Helix Servers

PERFORCE の SSL 暗号化

Perforce は、ネットワーク上でやりとりするデータをより厳重に保護するため、SSL 暗号化機能を提供しています。（また、EBS ストレージにも、Helix Core サーバー上の重要なデータを暗号化する機能が備わっています。）

SSL を使うことで、セキュリティを強化することができます。また、ベンチマークテストの結果から、SSL の使用にかかるコストは非常に小さいと分かっています。

AWS 内、オンプレミス、さらには AWS とオンプレミスインフラ間において、しっかりとした境界セキュリティ対策済みの場合は、SSL は必要ないかもしれません。

しかし、SSL の利用で発生する "複雑さ" もあります。

- OpenSSL 技術を使う場合、証明書の発行と管理が必要になります。ただ、Helix Core サーバーの場合は、独自の証明書発行が可能なため、証明書の期限切れだけ注意していれば済みます。
- SSL を利用した場合、フェイルオーバーが少し複雑になり、ユーザーにも影響が出てきます。これは、フェイルオーバーの目的（ユーザーが接続しているサーバーをこっそり切り替える）と、SSL の目的（アクセスしているサーバーが変わった際にユーザーに周知する）が相反関係にあるからです。ユーザー全員が、今まで存在を知らなかったフェイルオーバー用のサーバーを信頼できるか、という問題もあります。そのため、SSL を利用している Helix Core ユーザーは、フェイルオーバー後にユーザーが新しいサーバーの trust をやり直す必要があったとしても、受け入れる傾向があります。また、ロボットユーザーの再設定を行い、trust とログインロジックを Helix Core との連携に係るコードに組み込んでおくという、ベストプラクティスの採用も必要になってくるかもしれません。
- 大きな負荷がかかった Helix Core サーバーの接続が、SSL タイムアウトによって切れることもあるかもしれません。この場合、本来であれば、一時停止の後で問題なく処理されるはずのものが、ユーザーエラーになってしまいます。また、有益な Helix Core サーバーログが、役に立たない SSL タイムアウトエラーに置き換えられてしまうこともあります。ただ、これらの問題は、現在までのところ、SSL の利用を止めるほどには大きな問題にはなっていません。
- HMS サーバーが bastion ホスト上にデプロイされる場合、HMS インスタンスは常に SSL 設定されます。これは、他の "real data" インスタンスへのアクセスに悪用されかねない、センシティブな情報がこのサーバーには格納されている可能性があるからです。

3.11.6 その他ルール

Perforce が他にどのようなシステムと連携しているかによっては、追加のネットワークファイアウォールのルールが必要になってくるかもしれません。例えば、Helix Core LDAP 連携が使用されている場合、AWS マスターサーバーからオンプレミスの社内 Active Directory サーバーへのアクセスを（389 番ポートか 636 番ポート上で）許可するルールが必要になる可能性があります。

今後の展開

本テクニカルガイドの今後のこれから

AWS 社とそのパートナー会社のよる継続的な技術開発により、AWS インフラも AWS を使って提供されるサービスも進化し続けています。本テクニカルガイドも、ゲーム開発のために、Helix Core を AWS にデプロイしようと考えている皆様に最新の役立つ情報を提供できるよう、常にアップデートしていきます。

例えば、以下の情報の追加を検討しています。

- EFS をベストプラクティスに追加。アクティブ接続の制限など、いくつか検討すべき点もありますが、EFS はすでに、何社かのお客様環境で、/hxdepots ボリュームに対して効果的に使われています。
- AWS CloudFormation の活用。
- SDP や HMS の改良による AWS 上での操作の簡易化（微調整を不要に）。
- AWS Route 53 についての考察。
- CloudFormation を使ったリファレンス実装。EC2 スナップショットのバックアップや Glacier への移行の解説を含む。
- クラウドホスト型の仮想マシンを使った、高度なグローバルトポロジーのオンプレミス環境をシミュレーションできる Perforce Battle School トレーニングクラスの AWS への移行の検討。

- HA や DR ソリューションのさらなる改良。
- Perforce サーバーのデプロイメントへの適用性も踏まえた上での、Kubernetes などのコンテナ技術のメリットやコスト感についての考察。
- コーディングベストプラクティスとして、一般的なインフラに従うのに加えて、CloudFormation の使用は、新しいスタジオの立ち上げという、ゲーム開発業界の特定のユースケースでは適用可能かもしれません。マスターサーバーから離れた場所で、新しいゲーム開発スタジオを素早く立ち上げたいというのは、よくあるニーズです。パラメーター化された CloudFormation テンプレートを使うことで、デポのサブセット付きエッジサーバーを立ち上げて、新スタジオとの限られた範囲でのコラボレーションを実現できる可能性があります。

SDP CLOUD DEPLOY

SDP Cloud Deploy プロジェクトは、初期のプロトタイプ実装で、シンプルなテンプレートファイルを使ってサンプルの Helix Core 環境の構築（AWS アカウントの作成や、ゼロからのレプリカ構築を含む）ができることを実証するものです。

注記

本テクニカルガイドの見解には、以下も反映されています。

※1Helix Core サーバーのスペック名とホスト名には、一貫したサイトタグ（例：アメリカの北バージニア州の AWS us-east-1 リージョンのタグは、awsnva）を利用することが理想的です。

※2 Amazon EFS（AWS 上の NFS ストレージのさらに優れたもの）は、/hxdepots ボリューム向けだと考えられてきました。確かに、EFS はこの用途に適してはいますが、何点か欠点があります。その最たるものは、EBS ボリュームに対する rsync をしない限り、スナップショットがとれないという欠点です（ただし、EFS から EFS へのバックアップは可能）。これはつまり、 EBS ボリュームにバックアップを行う場合、キャパシティー予約の回避ができないということになります。とは言え、EFS には無視できない長所があることも事実であり、/hxdepots ボリュームと組み合わせた利用に　ついて、今後、改めて説明する予定です。直近でのリリースが予定されている EFS IA（Infrequent Access）は、一般的な Helix Core ワークロードに非常に適した、コストの最適化を可能にします。

しかし、本テクニカルガイドの執筆時においては、EBS ボリュームの方が実績があり、（スナップショット機能等の）理想的な機能を備え、最も高いパフォーマンス（EFS よりも低遅延）を実現できると言えます。EFS ボリュームは、必要とされるデータよりも自身を大きくするために、人工的な “garbage data injection” や “burst credits”の最適化を必要とします。これは、EFS のパフォーマンスは、サイズによって決まるからです。つまり、EFS としては大きければ、大きいほど良いということになります。