PURESTORAGE®

ROBLOX 導入事例：ロブロックス（Roblox）

# Portworx と HashiCorp Nomad の導入で 7000万人のゲーマーが利用する プラットフォームを構築

## 課題

- 世界のゲーマー 7000万人が利用するプラットフォームの可用性とパフォーマンスの維持および、運用コストの抑制
- 大規模障害発生時のデータ損失の回避
- 人的資源の増大を抑制しつつ、大規模なデータ・プラットフォームのスケーラビリティを確保

## ソリューション

- HashiCorp Nomad の導入により、VMおよびコンテナをベースとしたワークロードを数千のノードにスケーリング
- Portworx Enterprise の導入により、クラウド・ネイティブ・ストレージおよびデータ管理を実現

## 効果

- Portworx により、データベースなどのステートフルなアプリケーションを、耐障害性や冗長性を損なうことなく、ステートレスなサービスと同様に容易に管理できるようになった。
- Portworx のサポートおよびエンジニアリング・チームが専門知識を提供することで、ストレージ・プラットフォームの運用や拡張を行うことなく、膨大な数のユーザーのためのプラットフォームのスケーリングが可能になった。
- サイバー攻撃その他の大規模障害に対する備えとして、システム環境全体を 1 時間以内に容易に再構築できるようになった。

## 主要テクノロジー

- インフラストラクチャ：AWS、Azure、オンプレミス（ベアメタル）
- コンテナ・ランタイム：Docker
- オーケストレーション：HashiCorp Nomad
- ステートフル・サービス：CockroachDB、MongoDB、InfluxDB、ElasticSearch、GitLab、Jenkins、Docker Registry

**熱狂的人気を誇るゲーミング／エンターテインメント企業 Roblox のプラットフォームは、月間 7000万人のアクティブ・プレイヤーに利用されています。Roblox では、Portworx と HashiCorp Nomad の導入により、コンテナを活用したステートフルなアプリケーションの大規模な運用に成功しています。**

https://www.roblox.com/

※この資料では、Roblox の主任 SRE ロブ・キャメロン氏へのインタビュー取材の内容を抜粋してご紹介します。

Roblox
**主任 SRE**
**ロブ・キャメロン（Rob Cameron）氏**

## Roblox の事業はどのようなものですか？

Roblox では、ゲームを通じて世界をひとつにすることをミッションとしています。月間 7000万人以上の世界中の Roblox ファンが、数百万に及ぶ没入感の高いデジタル・ワールドを探索し、仲間と共に楽しんでいます。これらのデジタル・ワールドは、400万人以上のクリエイターで構成される Roblox コミュニティを通じて構築されています。Roblox のエコシステムにはさまざまな側面があり、ソーシャル・インタラクション、ゲーミング・エンターテイメント・システム、STEM 教育の促進、コーディングの学習など、多面的な目的で利用されています。

## どのような業務を担当されていますか？

私は製品開発グループの主任 SRE を務めています。適切なテクノロジーやサービスを利用して Roblox のプラットフォームの安定稼働を保証し、プレイヤーをはじめ、プラットフォームに関わる方々全員にメリットをもたらすことが、私の職務です。2004 年の創業以来、Roblox は飛躍的な成長を続けており、ビジネスの成長にあわせてプラットフォームのモダナイズが継続的に行われています。そのため、従来のインフラや運用方法との兼ね合いという課題が常に生じます。

私は、プラットフォームのオーケストレーションを主に担当するために Roblox に入社しました。400万人のクリエイターが利用するプラットフォームをより使いやすいものにするためには、新機能を迅速にリリースしなければなりません。爆発的な成長にあわせたスケーリングも重要です。これらの要件に欠かせないのがコンテナです。

Roblox では当初、単一のグローバル・プラットフォームを運用しており、このことが最大の課題のひとつでした。Roblox のプラットフォームを介して、中国、オーストラリア、米国など、世界中のプレイヤーが互いにやり取りをしています。一部のゲーミング・プラットフォームでは、共有環境を提供し、地理的な条件に基づいた制限を課していますが、Roblox ではそのような制限がありません。そのことが、よりよいゲーミング・エクスペリエンスにつながると考えています。しかし、技術的な課題を生じさせることにもなります。

## コンテナをどのように利用していますか？

コンテナの利用方法の前に、Roblox における基本的な構成と、これまでの経緯について説明します。Roblox のインフラ・モデルは、主に、データセンターと、POP（Point of Presence）と呼ばれるネットワーク拠点によって構成されています。データセンターは、データベースその他の一元化されたサービスを置く場所です。POP には、インターネット・プロバイダ、Roblox のバックボーン、ゲーム・サーバーとのルーティング上の接続点が含まれます。

かつて、Roblox のゲーム・サーバーは全て Windows 環境で稼働していました。子供たちが下校する時間帯を境にトラフィックが急上昇し、ゲーム・サーバーの数もそのトラフィックに応じて変動します。Windows 環境でのゲーム・サーバーの運用には、多大なコストがかかります。プラットフォームの人気が高まると同時に、運用コストの増大という長期的なリスクが高まります。私たちは、そのようなコストを、プレイヤーのユーザー・エクスペリエンスの改善やエンジニアの増強、海外展開の促進など、ビジネス成果に直結する活動に割り当てたいと考えました。

そこで、ゲーム・エンジンを Linux に移植することになりました。 2018 年の秋に移植が完了し、Linux 環境での運用管理を支援するために私が採用されました。Linux 環境では、プラットフォームの一部としてのさまざまなサービスを大規模に管理する手段および、デプロイメントの自動化が必要になります。このようなニーズに対応するため、コンテナを導入しました。

まだ Windows 環境で運用しているサービス、コンテナに移行していないサービスがあります。そのため、Windows と Linux コンテナおよび VM で動作する HashiCorp Nomad をオーケストレーション・エンジンとして採用することにしました。

**“まだ Windows 環境で運用しているサービス、コンテナに移行していないサービスがあります。そのため、Windows と Linux コンテナおよび VM で動作する HashiCorp Nomad をオーケストレーション・エンジンとして採用することにしました。”**

当初は、コンテナをゲーム・サーバーにのみ使用していましたが、コンテナへの移行をさらに進めたいと考えるようになり、GraphQL エンドポイントをはじめとする他のマイクロサービスや、サービス・ディスカバリ、動的ロード・バランサーなど、さまざまなインフラ・タスクを移行しました。

ゲーム・サーバーに注力してスチール・スレッドを構築し、コンテナへの移行は成功しました。コンテナを利用するためのインフラの構築も進み、現在では、インメモリ・データベース、コンテナとの親和性の高い他のデータベースなど、プラットフォームのためのマイクロサービスの移行を進めています。

最終的には、極めて高いパフォーマンスを要するごく一部のデータベースを除き、全てをコンテナに移行したいと考えています。ただし、開発チームに対して、合理的でない移行を強要するわけにはいきません。あくまでも業務に最適なツールを提供したいと考えています。チームによっては、コンテナをより積極的に受け入れるケースもあり、全体としては、コンテナの導入は非常にうまくいっていると考えています。

Roblox では現在、Nomad クラスタで約 50 のノードを実行しており、今後は、ゲーム・サーバーだけでも、5000 から 10000 のノードを実行する構想を持っています。プラットフォームを拡張して他のサービスを Nomad へ移行させ、2000 から 3000 の追加ノードを実行する考えです。

**“Roblox では現在、Nomad クラスタで約 50 のノードを実行しており、今後は、ゲーム・サーバーだけでも、5000 から 10000 のノードを実行する構想を持っています。プラットフォームを拡張して他のサービスを Nomad へ移行させ、2000 から 3000 の追加ノードを実行する考えです。”**

### コンテナでデータベースのようなステートフルなサービスを実行する際に、どのような課題がありましたか？

永続ストレージの確保が課題でした。ネイティブなストレージ・エクスペリエンスを提供するコンテナ・スケジューラがなく、このことは重大な問題でした。

以前の職場でも、ストレージ・レイヤーが大きな問題となっていました。ストレージ・レイヤーを使うより、ノードごとにストレージやローカル・ボリュームを割り当てるほうが、コストや柔軟性の面でかえって有利だと考えられていました。ストレージ・レイヤーでもなんとかやっていけるかもしれません。しかし、障害の発生を許容できないデータベースなどのミッション・クリティカルな運用においては、リスクが高すぎます。

また、ストレージ・レイヤーの管理負荷が大きく、期待した柔軟性やコスト削減の効果が相殺されてしまいます。

充実したシャーディングやデータ複製機能を持つ CockroachDB や MongoDB などのデータベースを使用していたとしても、コンテナ環境でタスクが急停止し、ディレクトリがガベージ・コレクトされ、データを失う、といった事態が発生した場合には、有効な対処方法がありません。これは確実に解決しなければならない問題です。

**“充実したシャーディングやデータ複製機能を持つ CockroachDB や MongoDB などのデータベースを使用していたとしても、コンテナ環境でタスクが急停止し、ディレクトリがガベージ・コレクトされ、データを失う、といった事態が発生した場合には、有効な対処方法がありません。これは確実に解決しなければならない問題です。”**

こうした問題を初めて解決しようとしたのは、Nomad で GitLab の実行を試してみたときでした。Roblox では、半年ごとに「ハック・ウィーク」という社内イベントを開催しています。常識的な範囲であれば、誰もが自由にプロジェクトを選んで遂行できる機会です。ゲームを制作する人もいれば、ゲーム用のツールを開発する人もいます。私のようなインフラ・マニアは、インフラに関連したものを作ります。

そのハック・ウィークの期間中に、私たちはチームを編成し、複数のシステムやツールを実行する際の大規模なソース・コントロール管理の問題解決に取り組むことにしました。

スケーリングを容易にするためのツールが欲しいという考えから、GitLab や GitHub Enterprise を検討しました。しかし、それらを実行する場所がないこと、専用の VM がないことがわかり、コンテナの Nomad での実行を試してみることになりました。

**“スケーリングを容易にするためのツールが欲しいという考えから、GitLab や GitHub Enterprise を検討しました。しかし、それらを実行する場所がないこと、専用の VM がないことがわかり、コンテナの Nomad での実行を試してみることになりました。”**

しかし、Nomad その他のコンテナ・ソリューションを使用する際に、私たちが当時、一貫したストレージ・レイヤーを必ずしも持っていなかったことが問題になりました。ローカル・ノード・ストレージを利用することは可能でしたが、それらの管理負荷を考慮しなければなりません。

そこで、NFS 共有を行う GlusterFS や Ceph などのソリューションをいくつか検討しましたが、ストレージ管理の負荷が発生するため、採用しないことにしました。

データ・レイヤーの採用には、データを引き継ぐ組織にどれだけの責任を委ねるかという問題が伴います。Gluster や Ceph などのソリューションは、さまざまなディストリビューションの一部として広く普及しており、一見使いやすいように思えます。

しかし、リスクがあります。データ消失やシステム障害が発生した場合に、どう対処すればよいでしょうか。管理をサポートを担う十分な規模のチームが社内にあるでしょうか。バックアップ、リストアをはじめ、運用ライフサイクル全体の管理のための十分な人員を確保できるでしょうか。それらが可能であれば、選択肢となり得ます。

しかし、急成長企業である Roblox には余剰人員は存在しません。 これらのソリューションを採用したとして、障害が発生した場合にはどう対処すればよいでしょうか。誰を頼りにすればよいでしょうか。

検討の結果、Portworx のような企業との連携が望ましいという考えに至ります。よくあるベンダーと顧客の関係ではなく、共に成長するパートナーとして互いに助け合い、データの可用性を維持するのです。問題が判明した場合には、Portworx から必要な数のエンジニアが割り当てられて、安全性・セキュリティが確保され、私たちにとって最も重要なプレイヤーのデータの消失を防いでくれます。

**急成長企業である Roblox には余剰人員は存在しません。これらのソリューションを採用したとして、障害が発生した場合にはどう対処すればよいでしょうか。誰を頼りにすればよいでしょうか。**

**私には、Portworx のような企業との連携が望ましく思えました。よくあるベンダーと顧客の関係ではなく、共に成長するパートナーとして互いに助け合い、データの可用性を維持するのです。**

**問題が判明した場合には、Portworx から必要な数のエンジニアが割り当てられて、安全性・セキュリティが確保され、私たちにとって最も重要なプレイヤーのデータの消失を防いでくれます。 ”**

他のソリューションでは、こうはいきません。VM 上で実行するワークロードに最適化されているため、私たちとは目指すところが異なります。Roblox のユースケースでは、コンテナベースのサービスを念頭に置いており、この方針は今後も変わりません。Roblox の目的に最も合致するのが Portworx でした。

Portworx を購入して最初に展開したユースケースは、それぞれの POP を独立した状態で実行し、Roblox のリージョナル・データセンターの中央サーバーで管理する Nomad クライアントを配置するというものでした。

さらに、テレメトリとアラート用に TICK スタック、あらゆるゲーム・インスタンスの中央ログ用に ELK スタックを、それぞれインストールしています。 コンテナ化環境には最適な構成です。なぜなら、ゲーム・サーバーにおいては、新しいインスタンスで更新しつつ、ログは保持する必要があるためです。

サイバー攻撃や、POP の侵害があった場合には、全ゲーム・サーバーを含む環境全体を容易に再利用し、1 時間以内に再構築することが可能です。

Portworx を導入したことで、ステートフルなアプリケーションをステートレスな環境と同様に実行できるようになったと同時に、耐障害性と冗長性を備えたプラットフォームをユーザーに提供できるようになりました。

**Portworx を導入したことで、ステートフルなアプリケーションをステートレスな環境と同様に実行できるようになったと同時に、耐障害性と冗長性を備えたプラットフォームをユーザーに提供できるようになりました。”**

Portworx 導入時のセットアップはわずか 2 時間で完了しました。開発クラスタの実行はもちろん、多くの異なるステートフルなサービスも迅速に起動させることができました。テストには、GitLab と Jenkins や CockroachDB などのツールを使用しました。さまざまなテストを実施した結果、極めて容易に使用できるソリューションであることが確認できました。セットアップが容易で、高い信頼性を提供してくれます。ボリュームの 3 つの複製を使用してパフォーマンス・テストを実施し、1 つのノードがオフラインの場合と、2 つのノードがオフラインの場合で、サービスが自動的にフェイルオーバーし、その後も稼働し続けるかどうかを確認しました。このテストでは、全体に問題なしという結果が得られました。

良好なテスト結果を受けて、必ずしも必要になるとは限らないストレージを大量に購入するのではなく、ディスクの有無に関わらず、さまざまなクラスのサーバーを利用できる Portworx 型の環境に移行し、Portworx と Nomad でワークロードを実行するべきであると考えました。 また、各コンテナ・ボリュームを個別にチューニングできる Portworx の仕組みも高評価でした。 この仕組みにより、データベースのワークロードを実行する場合に、データベースには ReadWriteOnce でパフォーマンスをベースとした 1 種類のボリュームを使用し、他の環境では共有ボリュームがあるケースに対応できます。

プレイヤーのデータには、しばしば機密性の高い情報が含まれるため、暗号化ボリュームが使用可能であることは極めて重要です。スナップショットが容易に作成できること、Portworx Lighthouse ユーザー・インターフェースによる一元化されたレポートおよび管理機能も、Portworx の優れた特長です。

Portworxは、私たちのインフラをシンプルにするために欠かせない、重要な役割を担っています。

**“プレイヤーのデータには、しばしば機密性の高い情報が含まれるため、暗号化ボリュームが使用可能であることは極めて重要です。スナップショットが容易に作成できること、Portworx Lighthouse ユーザー・インターフェースによる一元化されたレポートおよび管理機能も、Portworx の優れた特長です。**

**Portworxは、私たちのインフラをシンプルにするために欠かせない、重要な役割を担っています。 ”**

## ステートフルなサービスおよびコンテナの構築・運用についてアドバイスはありますか？

コンテナでのサービスの実行を評価する際には、当該サービスの実運用でどのような状況を目指すのかをまず明らかにすることが大切です。例えば Nginx のようなケースは容易に評価できます。しかし、さらに複雑なサービス、特にステートフルなサービスではそうはいきません。コンテナで複雑なサービスを実行する場合は、一定の場所で長時間存続する VM での実行とは要件が異なります。順調に実行している場合と、障害が発生した場合について、それぞれ想定するサービスの状態を明確にしておく必要があります。コンテナ化されたサービスの運用について、想定する内容を文書化しておくことをお薦めします。コンテナがどのように動作すべきか、どのような状況を期待するかという点で、人や役割によって解釈が異なる場合があるためです。

サービスからどのような結果を得るべきかについても、想定する基準を定義することと、テストを実施することが重要です。時間を要しますが、その価値は十分にあります。

ステートフル・サービスについては、次の点を考慮します。

1. ボリュームをどのようにプロビジョニングするか
2. サーバーやディスクに障害が発生した場合でも、運用を継続するにはどうすべきか
3. バックアップ、リストアをどのように行うか、どのようなツールを使用すべきか

“

**ステートフル・サービスについては、次の点を考慮します。**

1. **ボリュームをどのようにプロビジョニングするか**
2. **サーバーやディスクに障害が発生した場合でも、運用を継続するにはどうすべきか**
3. **バックアップ、リストアをどのように行うか、どのようなツールを使用すべきか** ”

また、ピュア・ストレージもしくは販売パートナーの担当者に問い合わせて、Portworx のベスト・プラクティスに関する情報を入手し、参考にすることをお薦めします。ユニークだと思われるユースケースでも、同様の事例が既に存在することが多いのです。実在のユースケースに基づいた情報は、大規模なサービスの実運用に向けた準備に確実に役立ちます。

## Portworx について

ピュア・ストレージの Portworx は、Kubernetes のための包括的なデータ・サービス・プラットフォームとして、永続ストレージ、データ保護、ディザスタ・リカバリ、データ移行、容量管理のための Kubernetes ネイティブな統合ソリューションを提供します。単一のデータ管理レイヤーが、実行環境を問わず、データベースをはじめとするあらゆるステートフル・サービスに対応し、複数のクラウドやオンプレミスのハイブリッド環境におけるコンテナ化されたアプリケーションの迅速なデプロイメントを可能にします。Portworx は、多くのエンタープライズにおいて、デプロイメントのコストおよび工数の削減に貢献しています。

ご相談・お問い合わせをお待ちしております。

PURESTORAGE®

ピュア・ストレージ・ジャパン株式会社
お問い合わせ：03-4563-7443（代表）
https://www.purestorage.com/jp/contact.html