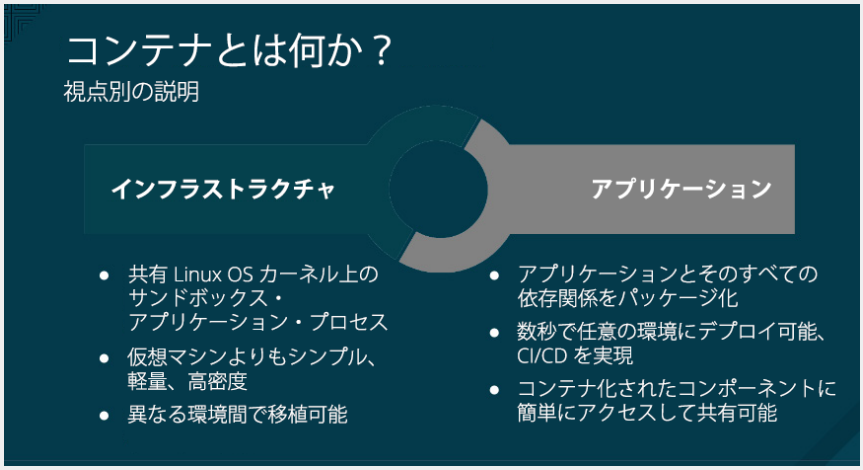
技術詳細

コンテナ・セキュリティの 10 大要素

はじめに

コンテナには幅広い魅力があります。たとえば、コンテナを使用することで、ユーザーはアプリケーションとそのすべての依存関係を単一のイメージに簡単にパッケージ化することができます。そしてそのイメージは、変更を加えることなく、開発からテスト、プロダクションへとプロモートすることが可能です。コンテナを使用すると、物理サーバー、仮想マシン (VM)、プライベートクラウドやパブリッククラウドといったさまざまな環境やデプロイ先で、一貫性を保つことが容易になります。これにより、チームは、ビジネス価値を提供するアプリケーションの開発と管理をより簡単に行うことができます。



企業は強力なセキュリティを必要としています。重要なサービスをコンテナで実行する人は、コンテナが安全であるか、自分たちのアプリケーションに使うコンテナが信頼できるものかを知りたがることでしょう。このホワイトペーパーでは、コンテナ・ソリューション・スタックの各種レイヤーとコンテナのライフサイクルのさまざまな段階におけるセキュリティの 10 大要素について説明します。

facebook.com/redhatjapan
@redhatjapan
linkedin.com/company/red-hat

jp.redhat.com

コンテナのセキュリティ保護：レイヤーとライフサイクル

コンテナのセキュリティ保護は、実行中のプロセスをセキュリティ保護することと似ています。まず、コンテナをデプロイして実行する前に、ソリューション・スタックのレイヤー全体にわたるセキュリティについて考える必要がありますまた、アプリケーションとコンテナのライフサイクル全体におけるセキュリティについても考える必要があります。コンテナ・セキュリティの 10 大要素は以下の通りです。

1. コンテナホストのマルチテナンシー
2. コンテナの中身
3. コンテナレジストリ
4. コンテナのビルド
5. コンテナのデプロイ
6. コンテナ・オーケストレーション
7. ネットワーク分離
8. ストレージ
9. アプリケーション・プログラミング・インタフェース (API) 管理
10. フェデレーテッド・クラスタ

1. コンテナホストのオペレーティング・システムとマルチテナンシー

コンテナにより、開発者がアプリケーションとその依存関係を 1 つのユニットとしてビルドし、提供することが容易になります。また、コンテナは、共有ホストへのマルチテナント・アプリケーションのデプロイを可能にすることで、サーバーを最大限に活用できるようにします。1 つのホストに複数のアプリケーションを簡単にデプロイし、必要に応じて個々のコンテナを起動したりシャットダウンしたりできます。また、従来の仮想化とは異なり、ハイパーバイザーは必要ありません。各 VM のゲスト・オペレーティング・システムの管理も不要です。コンテナは、ハードウェアではなくアプリケーション・プロセスを仮想化します。

コンテナというパッケージ化とデプロイのテクノロジーを最大限に活用するために、運用チームにはコンテナを実行するための適切な環境が必要です。運用チームに必要なのは、コンテナを境界で保護できるオペレーティング・システム (OS) です。これは、コンテナから脱出できないようホストカーネルを保護し、コンテナ同士を互いから保護します。

コンテナとは、分離性とリソース制限機構を備えた Linux® プロセスであり、共有ホストカーネル上でサンドボックス化されたアプリケーションの実行を可能にします。コンテナのセキュリティ保護は、Linux の実行プロセスをセキュリティ保護する場合と同じアプローチで行われます。その際、権限を制限することが重要なベストプラクティスとなります。コンテナを作成する際には、可能な限り最小限の権限を付与することが望ましいでしょう。また、コンテナは root ではなく一般ユーザーとして実行すべきです。次に、Linux が提供する複数のセキュリティを活用します。Red Hat® Enterprise Linux 上で動作するコンテナのセキュリティ保護に使用できるセキュリティ機能として、Linux 名前空間、Security-Enhanced Linux (SELinux)、Cgroups、ケーパビリティ (capabilities)、およびセキュア・コンピューティング・モード (seccomp) が存在します。

- **Linux 名前空間**は、コンテナの分離の基礎となります。名前空間を使用すると、名前空間内のプロセスにはグローバルリソースに独自のインスタンスがあるように見えます。名前空間はコンテナ内で専用のオペレーティング・システムを実行しているかのように見せる抽象化を提供します。
- **SELinux** は、コンテナを他のコンテナやホストから隔離した状態に保つための、追加のセキュリティレイヤーを提供します。SELinux では、管理者はすべてのユーザー、アプリケーション、プロセス、およびファイルに対して強制アクセス制御 (MAC) を適用できます。たとえるならば、SELinux は、(偶然または意図的に) 名前空間による抽象化をすり抜けてしまった場合に壁となるものです。

- Cgroups (コントロールグループ) は、一連のプロセスのリソース使用量 (CPU、メモリ、ディスク I/O、ネットワークなど) を制限し、集計し、分離します。Cgroups を使用することで、同じホスト上の別のコンテナによってコンテナがつぶされないようにできます。また、Cgroup は、一般的な攻撃ベクトルである擬似デバイスを制御するためにも使用できます。
- Linux の各種 Capability を使用すると、コンテナの root をロックダウンすることができます。Capability とは、個別に有効化または無効化できるよう分離された部分的な root 権限です。Capability を使用することで、raw IP パケットの送信や、1024 未満のポートへのバインドなどを行うことができます。コンテナを実行する際、複数の Capability を無効化しても、大半のコンテナ化アプリケーションには影響がありません。
- 最後に、セキュア・コンピューティング・モード (seccomp) プロファイルをコンテナに関連付けることにより、使用可能なシステムコールを制限できます。

Red Hat Enterprise Linux Atomic Host など、Linux コンテナの実行に最適化された軽量なオペレーティング・システムにコンテナをデプロイすることで、アプリケーションとインフラストラクチャのセキュリティをさらに強化することができます。Atomic Host は、ホスト環境を最小限に抑えながらコンテナ向けにチューニングすることで、攻撃の対象となる領域を減らします。

Red Hat Enterprise Linux Atomic Host で利用可能なセキュリティ機能の証明として、Red Hat Enterprise Linux 7.1 は最近、Linux Container Framework Support の認定を含む Common Criteria 認証を取得しました。

従来型の仮想化でもマルチテナンシーの実現は可能ですが、その方法はまったく異なります。仮想化はゲスト VM を初期化するハイパーバイザーに依存しています。各 VM に独自の OS があり、実行中のアプリケーションとその依存関係があります。VM の場合、ハイパーバイザーはゲストを互いに、そしてホストから隔離します。ハイパーバイザーにアクセスできるのは少数の個人やプロセスのみであるため、物理サーバーにおいて攻撃対象となる領域が減少します。ただし、脅威への対策として、セキュリティの監視は引き続き必要です。たとえば、あるゲスト VM がハイパーバイザーのバグを利用して別の VM やホストカーネルにアクセスする可能性があります。また、OS にパッチが必要な場合は、その OS を使用するすべてのゲスト VM でパッチを適用する必要があります。

コンテナはゲスト VM 内で実行することが可能で、そうすることが望ましいユースケースもあります。たとえば、従来型アプリケーションをコンテナにデプロイする場合、アプリケーションのクラウド移行を「リフト & シフト」方式で行うためには、コンテナをゲスト VM 内に配置したいと考えるかもしれません。しかし、単一ホスト上でコンテナにより実現するマルチテナンシーのほうが、より軽量で、柔軟性があり、拡張が容易なデプロイ・ソリューションを提供します。これは、特にクラウドネイティブ・アプリケーションに適したデプロイメント・モデルです。

2. コンテナの中身 (信頼できるソースを使用する)

セキュリティに関して考える際、コンテナの中身も重要です。ある程度の期間、アプリケーションとインフラストラクチャの構成には簡単に利用可能なコンポーネントが利用されてきました。これらの多くは、Linux オペレーティング・システム、Apache Web Server、Red Hat JBoss® Enterprise Application Platform、PostgreSQL、Node.js などのオープンソース・パッケージです。現在、これらのパッケージのコンテナ化されたバージョンも簡単に利用できるため、独自のパッケージをビルドする必要はありません。しかし、外部ソースからダウンロードするあらゆるコードと同様に、パッケージの提供元、作成者、そして内部に悪質なコードが含まれていないかを確認する必要があります。以下の事項を確認してください。

- コンテナの内容物は、使用中のインフラストラクチャのセキュリティを脅かさないか？
- アプリケーション・レイヤーに既知の脆弱性が存在しないか？
- ランタイムレイヤーと OS レイヤーは最新のものか？
- コンテナの更新頻度はどの程度で、更新されたことをどのような方法で知ることができるか？

Red Hat は長年にわたり、Red Hat Enterprise Linux と Red Hat のポートフォリオ全体において、信頼ある Linux コンテンツをパッケージ化して提供してきました。そして現在は、同様の信頼あるコンテンツを Linux コンテナとしてパッケージ化して提供しています。これには、Red Hat Enterprise Linux 7 と Red Hat Enterprise Linux 6 のベースイメージが含まれます。さらに、Red Hat Container Catalog を通じて、さまざまな言語ランタイム、ミドルウェア、データベースといった多数の認定済みイメージを提供しています。Red Hat 認定コンテナは、ベアメタルから VM、クラウドに至るまで、Red Hat Enterprise Linux が動作するあらゆる場所で実行できます。認定コンテナは、Red Hat とパートナーによってサポートされています。

コンテナイメージの中身は、既知のソースコードからパッケージ化されたものです。また、Red Hat はセキュリティ監視も提供します。Red Hat は、新たな Container Health Index で各コンテナイメージの「グレード」を公開し、プロダクション環境におけるシステムのニーズを満たすためにコンテナイメージを選定、使用、評価する方法を詳しく説明しています。コンテナのグレードの評価基準の一部には、未適用のセキュリティエラータの古さと、それがコンテナの全コンポーネントに及ぼす影響の度合いが含まれています。ここでは、コンテナの安全性に関する総合的な評価が、セキュリティの専門家にもそれ以外の人にも理解できる形式で提供されます。

Red Hat がセキュリティ・アップデート (glibc、Drown、Dirty Cow のフィックスなど) をリリースする際は、コンテナイメージの再ビルドと公開レジストリへの push も行っています。Red Hat セキュリティ・アドバイザリーは、認定コンテナイメージで新しく発見された問題を警告し、最新のイメージにユーザーを誘導します。それによりユーザーは、そのイメージを使用するアプリケーションを更新することができます。

もちろん、Red Hat が提供していないコンテンツが必要なときもあるでしょう。他のソースからのコンテナイメージを使用する際には、継続的に更新される脆弱性データベースを利用するコンテナ・スキャン・ツールを使用して、既知の脆弱性に関する最新情報を常に取得することをお勧めします。既知の脆弱性に関するリストは絶えず更新されるため、コンテナイメージをダウンロードした時点でコンテナイメージの中身をチェックし、さらに 、Red Hat が Red Hat コンテナイメージに関して行っているのと同じように、承認されデプロイされたすべてのイメージの脆弱性ステータスを追跡し続ける必要があります。

Red Hat は、プラガブルな API を Red Hat Enterprise Linux で提供することにより、OpenSCAP、Black Duck Hub、JFrog Xray、Twistlock など複数のスキャナーをサポートしています。Red Hat CloudForms を OpenSCAP と共に使用して、セキュリティ問題がないかコンテナイメージをスキャンすることもできます。また、Red Hat OpenShift では、継続的インテグレーションおよび継続的デリバリー (CI/CD) プロセスでスキャナーを使用することができます。これに関する詳細は後述します。

3. コンテナレジストリ (コンテナイメージへの安全なアクセス)

コンテナを構築する際、ダウンロードしたパブリック・コンテナ・イメージの上にコンテンツレイヤーを重ねることは一般的です。他の種類のバイナリを管理するのと同様に、ダウンロードしたコンテナイメージと社内でビルドしたイメージへのアクセスと提供も管理する必要があります。コンテナイメージの保存をサポートするさまざまなプライベート・レジストリが存在します。プライベート・レジストリ選定の際は、レジストリに保存されているコンテナイメージについて、使用ポリシーを自動化できるものを選ぶことをお勧めします。

Red Hat OpenShift には、コンテナイメージの管理に使用できるプライベート・レジストリが含まれています。OpenShift レジストリではロールベースのアクセス制御が提供され、誰が特定のコンテナイメージの pull や push を実行可能かを管理できます。また、OpenShift は、JFrog's Artifactory や Docker Trusted Registry など、すでに使用している他のプライベート・レジストリとの統合もサポートしています。

既知の脆弱性に関するリストは絶えず更新されているため、デプロイされたコンテナイメージと新しくダウンロードされたイメージの中身を継続的に追跡する必要があります。レジストリは、コンテナに関するメタデータ (既知の脆弱性を含む) に基づいてコンテンツを管理するための機能を備えているべきです。たとえば、Red Hat CloudForms SmartState 分析を使用して、レジストリ内の脆弱なイメージにフラグを立てることができます。フラグが立てられると、OpenShift はそれ以降、そのイメージが実行されることを防ぎます。

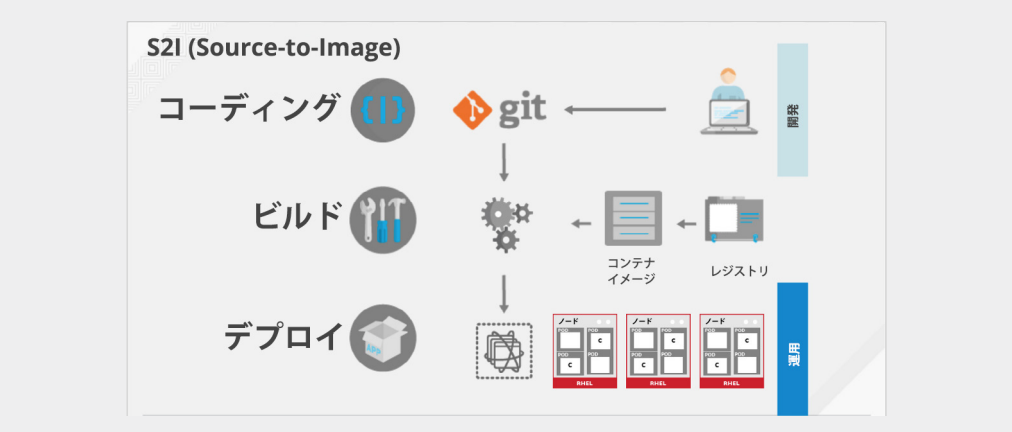
OpenShift がコンテナの中身に関するセキュリティポリシーの自動化をサポートするその他の方法については、次の数セクションでも紹介していきます。

4. セキュリティとビルドプロセス

コンテナ化された環境において、ソフトウェアのビルドプロセスとは、アプリケーション・コードが必要なランタイムライブラリと統合されるライフサイクルの段階です。このビルドプロセスの管理が、ソフトウェアスタックを保護するための鍵となります。「一度のビルドで、あらゆる場所にデプロイ」という思想に則れば、ビルドプロセスの成果物と完全に同一のものがプロダクション環境にデプロイされます。また、コンテナの不変性 (イミュータビリティ) を維持することも重要です。つまり、実行中のコンテナにパッチを当てるのではなく、コンテナを再ビルドして再デプロイします。

Red Hat OpenShift は、ビルド管理とセキュリティのための多くの機能を提供します。

- Source-to-image (S2I) は、ソースコードとベースイメージを結合するための、オープンソースのフレームワークです。S2I が提供する再現可能なビルド環境上では、開発チームと運用チームが簡単にコラボレーションすることができます。開発者が git でコードをコミットすると、S2I により OpenShift で以下のことが可能になります。
  - (コードリポジトリ上の webhooks または他の自動化された継続的インテグレーション (CI) プロセスを介して) 利用可能なアーティファクト (S2I ベースイメージ、新たにコミットされたコードなど) から新しいイメージの自動ビルドをトリガーできる
  - 新たにビルドされたイメージをテスト用に自動的にデプロイする
  - テストされたイメージをプロダクション・ステータスにプロモーションさせ、CI プロセスを通じて新しいイメージを自動的にデプロイできる



Red Hat OpenShift には、CI 実現のために統合された Jenkins インスタンスが含まれています。OpenShift には機能豊富な RESTful API も含まれており、独自のビルドツールや CI ツール、および JFrog's Artifactory などのプライベート・イメージ・レジストリの統合に利用できます。

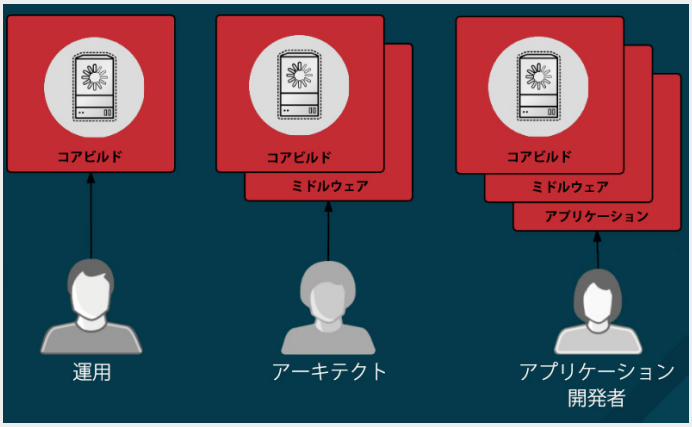
アプリケーションをセキュリティ保護するためのベストプラクティスは、自動化されたセキュリティテストをビルドプロセスや CI プロセスに組み込むことです。たとえば、以下のものを統合します。

- HP Fortify や IBM AppScan などの、静的アプリケーション・セキュリティ・テスト (SAST) ツールおよび動的アプリケーション・セキュリティ・テスト (DAST) ツール
- Black Duck Hub や JFrog Xray など、既知の脆弱性のリアルタイムチェックができるスキャナー。これらのツールは、コンテナ内のオープンソース・パッケージをカタログ化し、既知の脆弱性を通知し、過去にスキャンされたパッケージで新たな脆弱性が発見されたときに最新情報を知らせます。

さらに、CI プロセスには、セキュリティスキャンで発見された問題があるビルドにフラグを立てるポリシーが含まれているべきです。これにより、チームは問題に対処するための適切な措置を後回しにするのではなくすぐに講じることができます。

高度に規制された業界で働いている場合でも、単にチームの努力を最適化したい場合でも、コンテナイメージ管理とビルドプロセスを設計する際には、コンテナレイヤーを活用して職掌の分離を実装することを推奨します。

- 運用チーム：ベースイメージを管理
- アーキテクト：ミドルウェア、ランタイム、データベースなどのソリューションを管理
- 開発者：アプリケーション・レイヤーに集中し、コードの記述に専念



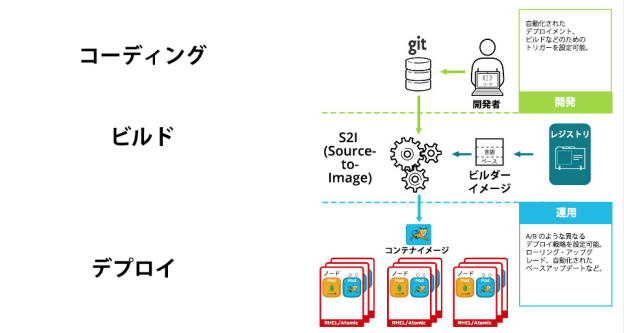
最後の点として、ビルドとデプロイの間にコンテナが改竄されていないことを確認できるよう、カスタムビルドのコンテナには署名することをお勧めします。

5. デプロイメント：クラスタ内にデプロイできるものを制御する

イメージのデプロイ後にビルドプロセス中の取りこぼしや脆弱性が発見された場合は、さらにもう 1 つセキュリティレイヤーが欲しいと思われるでしょう。つまり、自動化されたポリシーベースのデプロイを実現するためのツールです。

例として、コア・レイヤー、ミドルウェア・レイヤー、およびアプリケーション・レイヤーという 3 つのコンテナ・イメージ・レイヤーを使用してビルドされたアプリケーションを検討しましょう。コアイメージで問題が発見され、イメージが再ビルドされます。ビルドが完了すると、イメージは OpenShift のレジストリに push されます。OpenShift はイメージの変更を検出できます。このイメージに依存し、トリガーが定義されているビルドがあれば、OpenShift は修正されたライブラリを組み込んで、アプリケーション・イメージを自動的に再ビルドします。

ビルドが完了すると、イメージは OpenShift の内部レジストリに push されます。OpenShift は、内部レジストリ内のイメージへの変更を即座に検出し、トリガーが定義されているアプリケーションがあれば、更新されたイメージを自動的にデプロイし、プロダクションで実行されているコードが常に最新のイメージと同一であることを保証します。これらの機能すべてが連携して、CI/CD プロセスとパイプラインにセキュリティ機能が組み込まれます。



強力なセキュリティには自動化されたポリシーが含まれます。これらのポリシーを使用して、セキュリティの観点からコンテナのデプロイメントを管理することができます。OpenShift には、セキュリティ・コンテキストによる制約 (SCC) があらかじめ組み込まれています。これは、システムに受け入れられるために Pod (コンテナの集合体) が従う必要のある一連の条件を定義するために使用できます。Pod Security Policy として Kubernetes に貢献してきた OpenShift のセキュリティ・コンテキストによる制約 (SCC) により、管理者は以下を制御することができます。

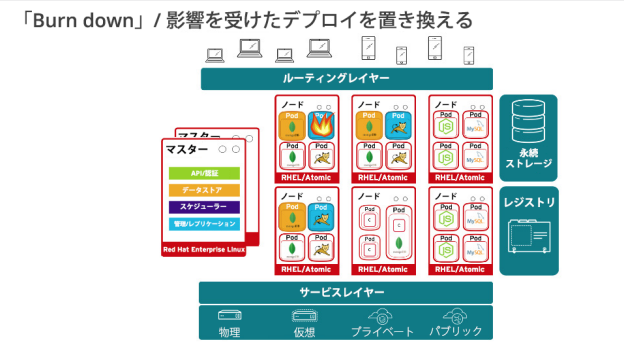
- 特権コンテナの実行
- コンテナが追加を要求できる機能
- ボリュームとしてホストディレクトリを使用すること
- コンテナの SELinux コンテキスト
- seccomp プロファイルの作成
- コンテナユーザー ID

必要な権限を有するユーザーは、希望する場合にはデフォルトの SCC ポリシーを調整して制約を緩めることができます。また、CI ツールと同様に、ユーザーは希望する場合、自分の CD ツールとコンテナ・プラットフォームを統合することができます。

CI/CD を OpenShift と統合することにより、最新の修正を組み込むためにアプリケーションを再ビルドし、テストし、そして環境内のすべての場所に確実にデプロイするプロセス全体を、完全に自動化することができます。

6. コンテナ・オーケストレーション：コンテナ・プラットフォームのセキュリティ保護

アプリケーションが単一のコンテナで提供されることは稀です。単純なアプリケーションにも、通常はフロントエンド、バックエンド、およびデータベースがあります。また、コンテナ化された最新のマイクロサービス・ベースのアプリケーションをデプロイすることは、複数のコンテナをデプロイすることを意味します。この図が示すように、デプロイ先は同じホストである場合も、複数のホストやノードに分散されている場合もあります。



大規模環境においてコンテナのデプロイメントを管理する場合、次の点を考慮する必要があります。

- どのホストにどのコンテナをデプロイするか
- どのホストの容量が大きいか
- どのコンテナが互いにアクセスする必要があるか、どのように互いの存在を検出するか
- ネットワークやストレージなどの共有リソースへのアクセスとその管理をどのように制御するか
- コンテナの正常性をどのように監視するか
- 需要に対応するため、アプリケーション容量の自動的なスケーリングをどのように行うか
- セキュリティ要件を満たしながら開発者向けのセルフサービス機能を有効化する方法

独自のコンテナ管理環境を構築することもできます。しかし、組み込みの管理機能とセキュリティ機能を備えたコンテナ・プラットフォームを導入すれば、チームはビジネス価値を提供するアプリケーションの構築に注力でき、ツールの構築に時間を費やす必要もありません。

Red Hat OpenShift Container Platform は、オープンソースの Kubernetes プロジェクトを組み込んで拡張することにより、コンテナ・オーケストレーション、スケジューリングの自動化、および物理または仮想マシンのクラスタ上でのアプリケーション・コンテナの実行を実現します。Google によって開始されたオープンソース・プロジェクトである Kubernetes は、コンテナ・クラスタ・オーケストレーションの複雑さを管理するために「マスター」を使用しています。OpenShift には Red Hat CloudForms も含まれます。これは、プライベート・レジストリ内のコンテナの正常性を監視し、新たに検出された脆弱性を有するコンテナのデプロイを防止するためなどに使用できます。

このコンテナ・プラットフォームは開発と運用の両方に役立つ機能を豊富に備えているため、強力なロールベースのアクセス制御が重要な要素となります。たとえば、オーケストレーション・マスターはアクセスの中心点となるので、最高レベルのセキュリティ監視を受ける必要があります。API は、大規模なコンテナ管理を自動化するための鍵となります。API は、Pod、サービス、および複製コントローラーに関するデータの検証および設定、リクエストに応じたプロジェクト検証の実行、他の主要なシステム・コンポーネントのトリガーの呼び出しに使用されます。

API アクセス制御 (認証と認可) は、コンテナ・プラットフォームを保護するために重要です。OpenShift の**マスター**には、組み込みの**OAuth サーバー**が含まれています。開発者と管理者は、**OAuth アクセストークン**を取得して API に認証されます。管理者は、Lightweight Directory Access Protocol (LDAP) ディレクトリなどの任意の **ID プロバイダー**を使用して認証するように OAuth を設定できます。

コンテナ・プラットフォームの重要な価値の 1 つは、開発者用のセルフサービス機能を実現することです。開発チームは承認されたレイヤー上でビルドされたアプリケーションを簡単かつ迅速に提供できるようになります。マルチテナンシー・セキュリティは、チームが許可なくお互いの環境にアクセスしないようにするためにプラットフォーム自体にとって必須です。セキュリティを実現しながらコラボレーションを促進するために、チームに十分な制御性を与えるセルフサービス・ポータルが必要です。OpenShift は Kubernetes にいくつかのコンポーネントを追加して、安全なマルチテナント・マスターを維持し、以下のことを保証します。

- マスターへのすべてのアクセスは、トランスポート・レイヤー・セキュリティ (TLS) 経由で実行
- API サーバーへのアクセスは、X.509 証明書またはトークンベース
- 不正なトークンが与えるダメージを抑えるためにプロジェクトクォータを使用
- Etcd はクラスタに直接公開されない

OpenShift 3.5 は、改善された秘密情報と証明書管理を含む、**拡張されたクラスタ管理**を提供します。

7. ネットワークの分離

コンテナ化された最新のマイクロサービス・ベースのアプリケーションをデプロイすることは、しばしば複数のノードに複数のコンテナを分散させてデプロイすることを意味します。ネットワーク防御を念頭におくと、クラスタ内でアプリケーションを互いに分離する方法が必要です。

Google Container Engine (GKE)、Azure Container Services、Amazon Web Services (AWS) Container Service などの一般的なパブリッククラウド・コンテナ・サービスは、シングル・テナント・サービスです。これらのサービスでは、各テナントの VM クラスタ上でコンテナを実行できます。安全なコンテナ・マルチテナンシーを実現するには、単一クラスタでトラフィックを分割し、そのクラスタ内の異なるユーザー、チーム、アプリケーション、および環境を分離できるコンテナ・プラットフォームが必要です。

ネットワーク名前空間では、コンテナの各集合体 (「Pod」) は、バインド先となる独自の IP とポート範囲を取得し、それによりノード上の Pod ネットワークは互いから分離されます。異なる名前空間 (プロジェクト) の Pod は、デフォルトでは別のプロジェクトの Pod やサービスにパケットを送信したり、そこから受信したりできません。ただし、後で説明する例外オプションがあります。これらの機能を使用することで、同じクラスタ内で開発、テスト、プロダクションの環境をそれぞれ分離することができます。

しかし、このような IP アドレスとポートの急増は、ネットワークの複雑化を招きます。加えて、コンテナはその性質上、常に入れ替わります。したがって、この複雑さに対処するツールに投資することをお勧めします。ソフトウェア・デファインド・ネットワーク (SDN) を使用することで、統一されたクラスタネットワークを提供し、クラスタ全体におけるコンテナ間通信を可能にするコンテナ・プラットフォームが望ましいです。

また、ルーターまたはファイアウォール方式のいずれかで IP ホワイトリストを使用した出力トラフィック制御が可能なコンテナ・プラットフォームが望ましいです。例えばデータベースアクセスなどを制御できます。

ネットワーク名前空間に加えて、SDN は、ovs-multitenant プラグインでマスター (オーケストレーション) 名前空間の分離を提供することにより、セキュリティを強化します。ovs-multitenant プラグインを有効にすると、名前空間内の Pod トラフィックは、デフォルトで、別の名前空間 (プロジェクト) 内の Pod トラフィックから隔離されます。ovs-multitenant プラグインの例外を提供するために、Red Hat OpenShift Container Platform 3.1 に「oadm pod-network」機能が導入されました。これにより、2 つのプロジェクトが互いのサービスにアクセスしたり、すべてのプロジェクトがクラスタ内のすべての Pod やサービスにアクセスしたりすることを許可できます。これにはプロジェクト全体のレベルで動作するという制限があり、許可されるトラフィックは常に双方向です。つまり、あるプロジェクトのあるサービスにアクセスできるユーザーは、そのプロジェクトのすべてのサービスにアクセスできます。そして、自分のプロジェクトのすべてのサービスへのアクセス許可も必然的に与えているということです。この権限は双方向的であるため、クラスタ管理者のみが設定できます。

Red Hat OpenShift Container Platform 3.5 のテクノロジー・プレビューとして導入された新しいネットワーク・ポリシー・プラグイン (ovs-networkpolicy) は、ovs-multitenant プラグインを使用して Pod 間の許容トラフィックを設定する方法を改善するよう設計されています。ネットワークポリシーを使用すると、個々の Pod レベルで分離ポリシーを設定できます。ネットワークポリシーは双方向的ではなく、プロジェクト管理者の制御下で Pod の入力トラフィックにのみ適用されるため、クラスタ管理者権限も必要ありません。

ネットワーク・スキャナーを導入したい場合には、簡単にコンテナ化でき「スーパー特権コンテナ」として実行できます。

8. ストレージ

コンテナは、ステートレス・アプリケーションとステートフル・アプリケーションの両方において役立ちます。接続されたストレージの保護は、ステートフル・サービスのセキュリティ保護において重要な要素です。Red Hat OpenShift Container Platform は、ネットワーク・ファイル・システム (NFS)、AWS Elastic Block Stores (EBS)、GCE Persistent Disks、GlusterFS、iSCSI、RADOS (Ceph)、Cinder などさまざまなストレージのプラグインを提供しています。

永続ボリューム (PV) は、リソースプロバイダーがサポートしている方法であればどの方法でもホストにマウントできます。プロバイダーはそれぞれ異なる機能を持ち、各 PV のアクセスモードはその特定のボリュームでサポートされている特定のモードに設定されます。たとえば、NFS は複数の読み取り/書き込みクライアントをサポートできますが、特定の NFS PV はサーバー上で読み取り専用でエクスポートされることがあります。各 PV は、特定の PV の機能を説明するアクセスモードのセットを取得します。たとえば、ReadWriteOnce、ReadOnlyMany、ReadWriteMany などです。

共有ストレージ (NFS、Ceph、Gluster など) の場合、共有ストレージ PV にそのグループ ID (gid) を PV リソースのアノテーションとして登録させます。PV が Pod により要求されると、アノテーションとして登録された gid が Pod の従属グループに追加され、その Pod に共有ストレージのコンテンツへのアクセス権が与えられます。

ブロックストレージ (EBS、GCE Persistent Disks、iSCSI など) の場合、コンテナ・プラットフォームは SELinux 機能を使用して非特権 Pod 用にマウントされたボリュームの root を保護し、マウントされているボリュームは関連付けられたコンテナに所有され、そのコンテナからのみ見えるようにします。

転送中のデータは、相互に通信するすべてのコンテナ・プラットフォーム・コンポーネントで https を使用して暗号化する必要があります。

また、選択したストレージ・ソリューションで利用可能なセキュリティ機能も併せて活用しましょう。

9. API 管理 / エンド・ポイント・セキュリティとシングルサインオン (SSO)

アプリケーションのセキュリティ保護には、アプリケーションの管理に加えて、API 認証と認可が含まれます。

Web SSO 機能は現代的アプリケーションの重要要素です。コンテナ・プラットフォームにはアプリケーション構築に開発者が使用できるコンテナ化されたサービスが多数用意されています。たとえば、Red Hat SSO (RH-SSO)、すぐに利用でき完全サポートされている SAML 2.0 ベースまたは OpenID Connect ベースの認証、Web シングルサインオン、アップストリーム Keycloak プロジェクトをベースにしたフェデレーション・サービスなどがあります。RH-SSO 7.1 には、Red Hat JBoss Fuse および Red Hat JBoss Enterprise Application Platform (JBoss EAP) 用のクライアントアダプタが搭載されています。RH-SSO 7.1 には新しい Node.js クライアントアダプタがあります。これにより、Node.js アプリケーションの認証と Web シングルサインオンが可能になります。RH-SSO は、Microsoft Active Directory や Red Hat Enterprise Linux Identity Management などの LDAP ベースのディレクトリサービスと統合することができます。RH-SSO は、Facebook、Google、Twitter などのソーシャル・ログイン・プロバイダーとも統合されています。

API は、マイクロサービスで構成されているアプリケーションにとって重要です。これらのアプリケーションには複数の独立した API サービスがあるので、サービス・エンドポイントの急増につながります。そのため、ガバナンスのための追加ツールが必要になります。加えて、API 管理ツールを使用することをお勧めします。3Scale by Red Hat は、API 認証とセキュリティに関するさまざまな標準オプションを提供します。これらのオプションを単独または組み合わせて使用することで、資格情報の発行やアクセス制御を行えます。これらのオプションには、標準 API キー、アプリケーション ID とキーのペア、OAuth 2.0 などが含まれます。

3Scale のアクセス制御機能は、基本的なセキュリティと認証以上のものを提供します。アプリケーション・プランとアカウントプランでは、特定のエンドポイント、メソッド、およびサービスへのアクセスを制限し、ユーザーグループにアクセスポリシーを適用できます。アプリケーション・プランでは、API

使用率のレート制限を設定し、開発者グループのトラフィックフローを制御できます。インフラストラクチャを保護し、トラフィックがスムーズに流れるようにするためには、期間あたりの受信 API コールを制限します。また、レート制限に達するか超過したアプリケーションの超過アラートを自動的にトリガーし、制限を超過したアプリケーションに対する動作を定義します。

10. クラスタ・フェデレーションにおけるロールとアクセス管理

2016 年 7 月、Kubernetes 1.3 は初めて Kubernetes Federated Clusters を導入しました。これは Kubernetes のアップストリームで進化している画期的な新機能の 1 つで、現在、Kubernetes 1.6 ではベータ版です。フェデレーションは、パブリッククラウドまたはエンタープライズ・データセンターの複数クラスタにまたがるアプリケーション・サービスのデプロイおよびアクセスに役立ちます。複数クラスタは、複数のアベイラビリティー・ゾーンにわたるアプリケーションの高可用性を実現したり、AWS、Google Cloud、Azure などの複数のクラウドプロバイダーにまたがるデプロイや移行の共通管理を可能にしたりするために役立ちます。

フェデレーテッド・クラスタを管理する際は、使用しているオーケストレーション・ツールが、異なるデプロイメント・プラットフォーム・インスタンス全体に必要なセキュリティを提供できることを確認する必要があります。ここでも、認証と認可が重要な要素となります。加えて、実行されている場所に関わらず、アプリケーションにデータを安全に渡し、クラスタ全体でアプリケーションのマルチテナンシーを管理できる能力も重要です。Kubernetes は Cluster Federation を拡張しており、Federated Secrets、Federated Namespaces、Ingress オブジェクトにも対応するようになりました。

Federated Secrets は、フェデレーション内のすべてのクラスタにわたりシークレットを自動的に作成して管理し、オリジナルの更新が適用されたときに一部のクラスタがオフラインであったとしても、全体において一貫性と最新の状態が保たれるようにします。

Federated Namespaces は、従来の Kubernetes 名前空間に似ており、同じ機能を提供します。フェデレーション・コントロール・プレーンで名前空間を作成すると、フェデレーション内のすべてのクラスタ間で名前空間が同期されます。

Red Hat はコミュニティと協働しながら、これらの機能が成熟するにつれて、OpenShift に追加していく予定です。

まとめ

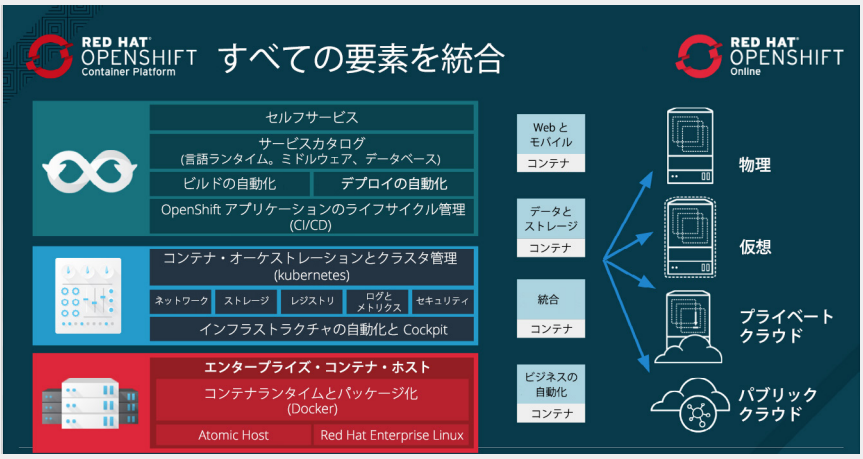
当然ながら、セキュリティ以外にも考慮すべき事項が存在します。コンテナ・プラットフォームは、開発チームや運用チームに適したエクスペリエンスを提供する必要があります。必要なのは、各チームが必要とする機能を損なうことなく、開発と運用の両者が運用効率とインフラストラクチャの利用効率を向上できる、安全なエンタープライズ・グレードのコンテナ化アプリケーション・プラットフォームです。

OpenShift 3 は標準のポータブル Linux コンテナのコア上に構築されています。OpenShift には以下のセキュリティ機能を提供します。

- エンタープライズ認証システムとの統合による強力なロールベースのアクセス制御
- Google Kubernetes による強力な Web スケールのコンテナ・オーケストレーションと管理
- 統合された Red Hat Enterprise Linux 7 と Red Hat Enterprise Linux Atomic Host (大規模環境でのコンテナ実行に最適化されており、SELinux による強力な分離が可能)
- パブリックレジストリおよびプライベート・レジストリとの統合
- 統合された CI/CD ツールによる安全な DevOps プラクティスの実現支援
- 新しいコンテナ・ネットワーク・モデル
- リモート・ストレージ・ボリュームのサポート

さらに OpenShift は、幅広いプログラミング言語、フレームワーク、サービスをサポートしており、フェデレーテッド・クラスタへの対応もロードマップに含まれています。

OpenShift は、OpenStack、VMware、AWS、GCP、Azure、および Red Hat Enterprise Linux 7 を提供するあらゆるプラットフォーム上で実行できます。さらに、Red Hat ではパブリッククラウド・サービスとして OpenShift Dedicated と OpenShift Online も提供しています。



Red Hat は、世界有数のオープンソースソリューション・プロバイダーとして過去 15 年以上にわたり、優れた信頼性と安全性を誇る完全オープンソースのソリューションを法人のお客様に提供してきました。Red Hat Enterprise Linux や Red Hat OpenShift Container Platform などのソリューション、そしてコンテナ対応の Red Hat 製品ポートフォリオ全体を通じて、Red Hat は同水準の信頼性とセキュリティをコンテナに提供します。

RED HAT について

オープンソースソリューションのプロバイダーとして世界をリードする Red Hat は、コミュニティとの協業により高い信頼性と性能を備えるクラウド、Linux、ミドルウェア、ストレージおよび仮想化テクノロジーを提供、さらにサポート、トレーニング、コンサルティングサービスも提供しています。Red Hat は、お客様、パートナーおよびオープンソースコミュニティのグローバルネットワークの中核として、成長のためにリソースを解放し、ITの将来に向けた 革新的なテクノロジーの創出を支援しています。

アジア太平洋 +65 6490 4200
オーストラリア 1 800 733 428
ブルネイ / カンボジア 800 862 6691
インド +91 22 3987 8888
インドネシア 001 803 440224
日本 03 5798 8510
韓国 080 708 0880
マレーシア 1 800 812 678
ニュージーランド 0800 450 503
フィリピン 800 1441 0229
シンガポール 800 448 1430
タイ 001 800 441 6039
ベトナム 800 862 6691
中国 800 810 2100
香港 852 3002 1362
台湾 0800 666 052

facebook.com/redhatjapan
@redhatjapan
linkedin.com/company/red-hat

jp.redhat.com
#f7530_0517

Copyright © 2018 Red Hat, Inc. Red Hat、Red Hat Enterprise Linux、Shadowman ロゴ、および JBoss は、米国およびその他の国における Red Hat, Inc. またはその子会社の登録商標です。Linux® は、米国およびその他の国における Linus Torvalds 氏の登録商標です。OpenStack® のワードマークと OpenStack のロゴは、米国とその他の国における OpenStack Foundation の登録商標/サービスマークまたは商標/サービスマークのどちらかであり、OpenStack Foundation の許諾の下に使用されています。Red Hat は、OpenStack Foundation と OpenStack コミュニティのいずれにも所属しておらず、公認や出資も受けていません。