第4章:
ICP Kubernetes環境での基盤運用

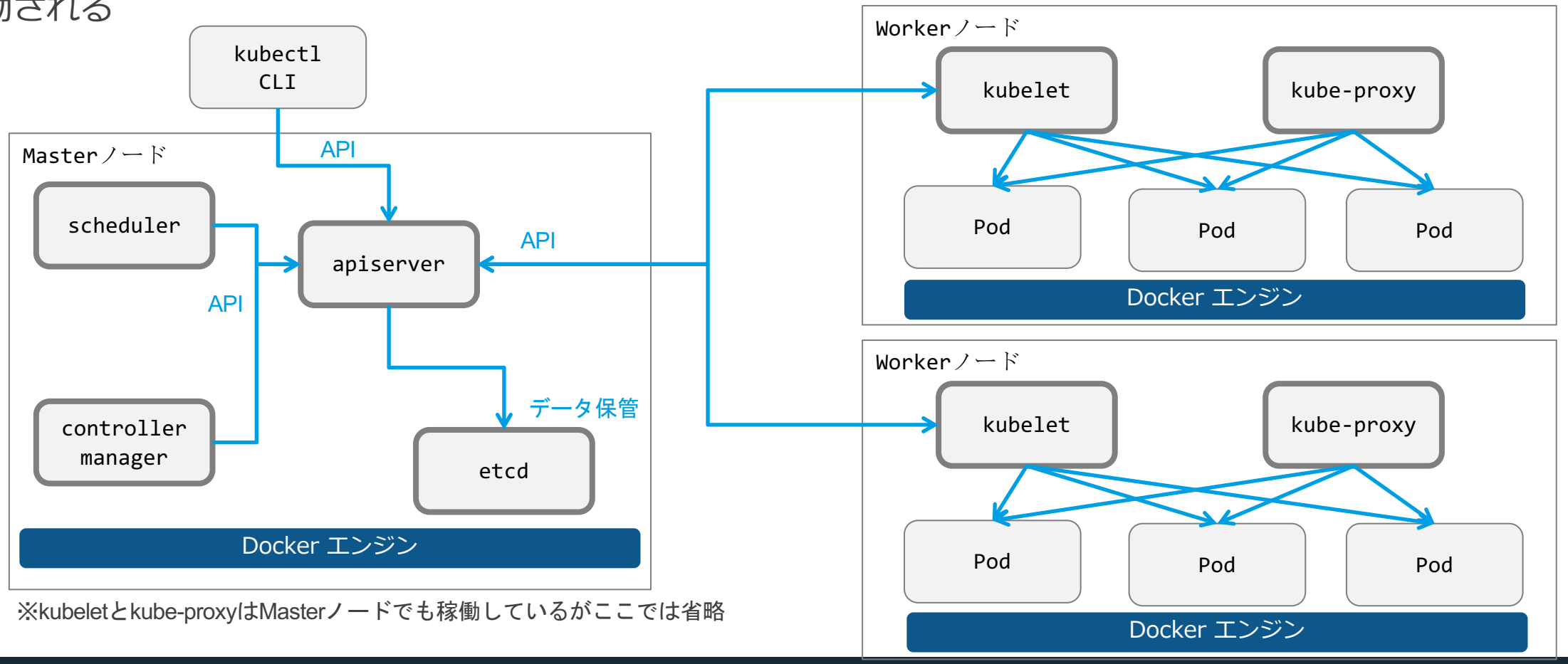
章内目次

第4章: ICP Kubernetes環境での基盤運用

第4章 第1節:
システム起動・停止

システム起動・停止（Knowledge Center記載の手順）

- IBM Cloud Private製品のKnowledge Center記述では**kubeletサービスおよびdockerサービスの停止、起動**による再起動手順が示されている
  - 参考情報: IBM Knowledge Center - クラスターの再始動
  - 各ノードではDockerエンジン及び kubelet が systemdとして稼働し、他のコンポーネントは Podとして起動される



※kubeletとkube-proxyはMasterノードでも稼働しているがここでは省略

システム起動・停止（ワーカー・ノード停止に伴う考慮点）

■ ワーカー・ノード停止時は以下の2点を考慮する（Knowledge Center および「Kubernetes.io」記述に基づく）

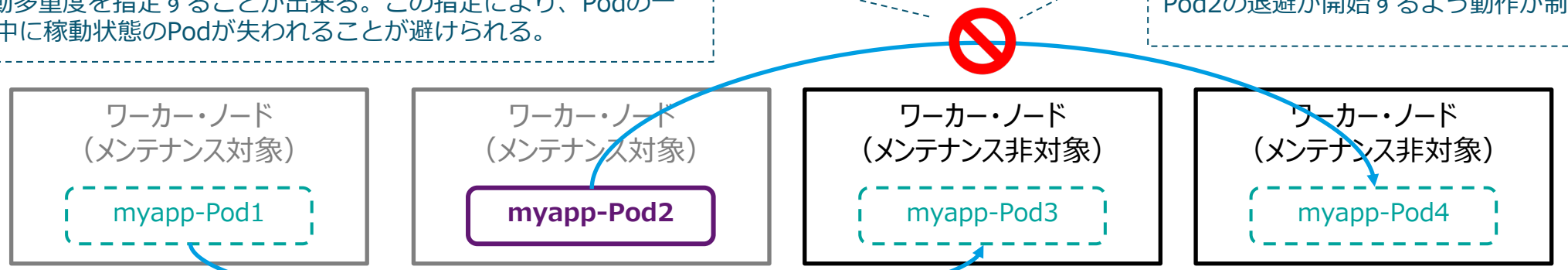
–“kubectl drain”コマンドによる、停止するノードからのPodの事前退避の要否を検討する

–“PodDisruptionBudget”オブジェクトを用いて、ノードからのPod退避時にPodの稼動多重度を制御することを検討する

- 参考情報: Knowledge Center – ノードの保守
- 参考情報: Safely Drain a Node while Respecting Application SLOs – Kubernetes
- 参考情報: Disruptions – Kubernetes

“PodDisruptionBudget” オブジェクトを用いたPod稼動多重度の制御:
“PodDisruptionBudget”オブジェクトを用いると、Pod退避時に確保すべきPodの稼動多重度を指定することが出来る。この指定により、Podの一斉退避の最中に稼動状態のPodが失われることが避けられる。

“PodDisruptionBudget”の機能により、Pod退避時におけるPodの稼動多重度を制御することが出来る（”myapp-Pod1”の退避が完了した後にmyapp-Pod2の退避が開始するよう動作が制御される）。



“kubectl drain” コマンドによるPodの退避:
“kubectl drain”コマンドを用いると、コマンドの対象として指定したノードからPodを退避することが出来る。

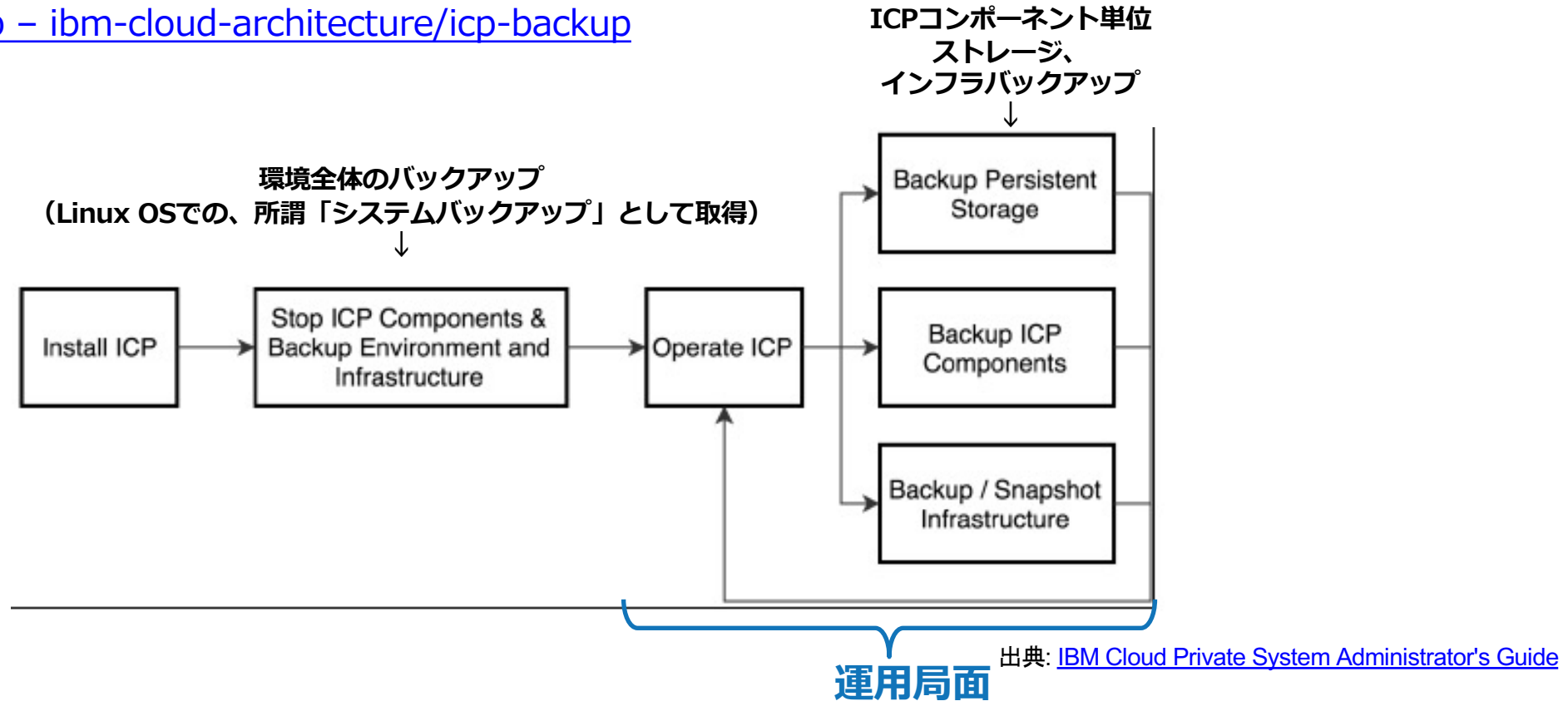
システム起動・停止（冗長度喪失に伴う影響）

- システム停止運用についてはマスター・ノード、およびプロキシー・ノードの停止がもたらすシステム影響について把握し、十分な実機検証の対象とされることが望ましい
  - マスター・ノード のクラスターにおいて許容される停止ノード数は（N-1)/2 ノードである
  - 例えば、マスター・ノード 3台のクラスターにおいて許容される停止ノード数は (3-1)/2 = 1 ノード となる
  - システム停止運用の際は、同時に２ノードを停止することのないよう手順上の考慮が必要
    - 参考情報: 高可用性 IBM Cloud Private クラスター

第4章 第2節:
バックアップ・リストア

バックアップ・リストア（IBM Redbookに基づくバックアップ戦略）

- IBM RedbookおよびGitHubでは、初期構築後やアップグレード後に環境全体のバックアップをマスターバックアップとして取得した後、運用局面においては定期的に永続ストレージ、ICPコンポーネント、インフラのバックアップを取得する戦略が推奨されている
  - 参考情報: IBM Cloud Private System Administrator's Guide
    - *Chapter 3.4 Backup and restore strategy*
  - 参考情報: GitHub – ibm-cloud-architecture/icp-backup



出典: IBM Cloud Private System Administrator's Guide

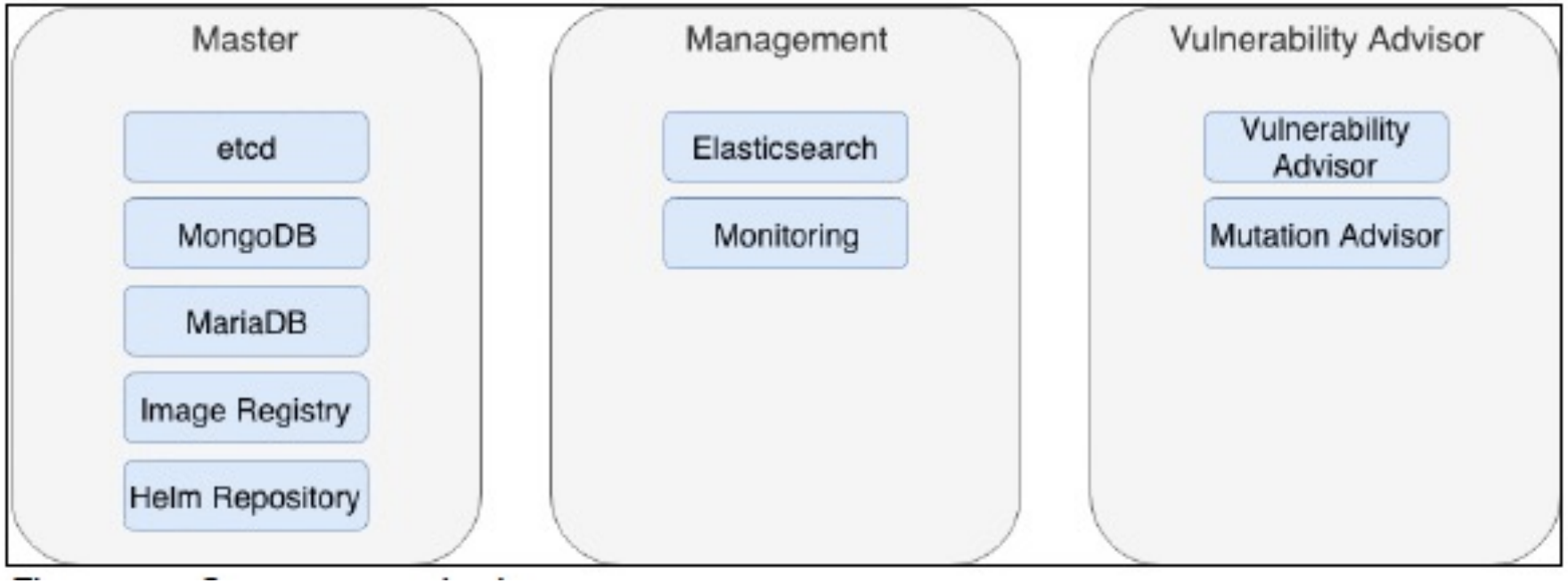
バックアップ・リストア（環境全体のバックアップ対象）

- 環境全体のバックアップはICPをホストするハイパーバイザーが提供または推奨するツール・機能でノード単位のOSバックアップを取得する
- ノードの種別によっては再作成可能なため、定期的なバックアップは必須ではない
  - 参考情報: IBM Cloud Private System Administrator's Guide
    - *Chapter 3.4 Backup and restore strategy*
  - 参考情報: GitHub – ibm-cloud-architecture/icp-backup

ノード種別	役割	バックアップの必要性
ブート・ノード	インストール、構成、ノードのスケーリング、クラスターのアップデートを行う際に使用する。クラスター構成情報及び証明書を保持する	○
マスター・ノード	管理サービスを提供し、ワーカー・ノードを制御する。リソースの割り当てや状態管理、スケジューリング、モニタリングを行う重要プロセスが稼働する。etcd, MongoDBなどのコア・コンポーネントが配置される	○
プロキシー・ノード	ICPクラスターで提供されるサービスへ外部からのリクエストをディスパッチする。再構築可能なため、ノードのホストファイルシステムに重要なデータが保管されていない場合はバックアップは必須ではない	オプション
管理ノード	Istio、モニタリング、メータリング、ロギングなどの管理サービスが稼働する。バックアップの重要度は管理サービスで収集されるデータのその環境における重要度に依存する。	オプション
脆弱性アドバイザー・ノード	脆弱性アドバイザーが稼働するノード（オプション）。バックアップの重要度は脆弱性アドバイザーが収集するデータのその環境における重要度に依存する	オプション
ワーカー・ノード	アプリケーション等が稼働するノード。再構築可能なため、ノードのホストファイルシステムに重要なデータが保管されていない場合はバックアップは必須ではない	オプション
etcd ノード	ICPクラスターにおける分散KVSコンポーネント。通常はマスター・ノードと統合されている。多数のワーカー・ノードが存在する環境で独立したノードとして存在する場合はバックアップが必須	○

バックアップ・リストア（コンポーネント単位でのバックアップ対象）

- IBM Redbook では、下図に示されたコンポーネントの保持するデータが運用局面でのバックアップ対象として挙げられている
- 各コンポーネントについては IBM Redbook や GitHub にて具体的バックアップ手順の例やサンプル・スクリプト等が示されているので、必要に応じて参考にできる
- IBM Redbook の First Edition(April 2019）は、ICP 3.1.2 を対象としている。一方で IBM GitHub はバージョンを限定しない記述が多いため、手順を参考とする場合は 使用環境で検証を行うようガイドされている



出典: IBM Cloud Private System Administrator's Guide

バックアップ・リストア（コンポーネント単位でのバックアップ手順）

- IBM Redbook、GitHubで示されている各コンポーネントの概要、バックアップ手順の概略は下表のとおり
  - 参考情報: IBM Cloud Private System Administrator's Guide
    - *Chapter 3 Backup and restore of an IBM Cloud Private cluster*
  - 参考情報: GitHub – ibm-cloud-architecture/icp-backup

バックアップ対象	バックアップ対象の概要	参考手順の提供	バックアップ手順の概略
clusterディレクトリー	クラスター導入時のコアデータが保管されているディレクトリー	あり (IBM Redbook)	ブート・ノードの <インストールディレクトリー>/cluster ディレクトリーをバックアップする
etcd (etcd ノード)	構成データを保管するためのkey-value型の分散DB	あり (IBM Redbook, GitHub)	定期的にetcd データベースのスナップショットをノード上のファイルとして取得する（マルチMaster環境ではいずれか一つのMasterノードで取得）
プライベート・イメージ・レジストリー	Dockerイメージを保管するためのプライベートレジストリ	あり (IBM Redbook, GitHub)	Masterノードのいずれかでプライベートレジストリが使用するファイルシステム（/var/lib/registy）をバックアップする
MongoDB	Meteringサービス、Helm Repositoryサービス、Helm APIサービスにより利用されるNoSQL DB LDAP構成情報、team/user/role情報も保持する	あり (IBM Redbook, GitHub)	MongoDBのデータを mongodump コマンドを使用してエクスポートする

バックアップ・リストア（コンポーネント単位でのバックアップ）

■ IBM RedbookでおよびGitHub示されている各コンポーネントの概要、バックアップ手順の概略は下表のとおり（前ページからの続き）

バックアップ対象	バックアップ対象の概要	参考手順の提供	バックアップ手順の概略
MariaDB	OIDCサービスが利用するDB	あり (IBM Redbook)	mysqldump コマンドを使用してmariadbのバックアップを取得
Helm リポジトリー	ICP カタログにアップロードされた全てのHelm チャートを保持するレポジトリー	あり (IBM Redbook)	Helm-repo ポッドをホストするマスター・ノード上の /var/lib/icp/helmrepoディレクトリーをバックアップする
Elasticsearch ログデータ	ElasticsearchおよびLogstashにより収集されるログデータ	あり (IBM Redbook)	管理ノード上のElasticsearchを停止し、/var/lib/icp/logging/elk-data/nodes/0/indicesディレクトリーをバックアップする
Elasticsearch設定データ	Elasticsearchに関わる設定データ（Elastic stack システム設定、認証用の証明書・パスワード	なし	ConfigMap, 証明書・パスワードを退避、Elasticsearchのスナップショット取得
モニタリング データ	Alert Manager, Prometheus, Grafanaにより収集・可視化されるモニタリングデータ	なし	モニタリング・データについては永続ボリュームに保管されたモニタリングデータをバックアップする
PrometheusおよびGrafanaの設定データ	PrometheusやGrafanaでユーザーが作成したアラート・ルールやダッシュボード	あり (IBM Redbook)	アラート・ルールやダッシュボードは yaml ファイルへエクスポートする ダッシュボードをjsonファイルへエクスポートする

バックアップ・リストア（コンポーネント単位でのバックアップ）

■ IBM RedbookでおよびGitHub示されている各コンポーネントの概要、バックアップ手順の概略は下表のとおり（前ページからの続き）

バックアップ対象	バックアップ対象の概要	参考手順の提供	バックアップ手順の概略
脆弱性アドバイザー/ミューテーション・アドバイザー	脆弱性アドバイザーおよびミューテーション・アドバイザーで使用する、KafkaおよびElasticsearchに保管されるスキャンデータ	あり (IBM Redbook)	脆弱性アドバイザーノード上の/var/lib/icp/va ディレクトリーをバックアップする Elasticsearchのデータをバックアップする（前ページのElasticsearch ログデータの箇所を参照）
永続ボリューム上のアプリケーションデータ	アプリケーションが使用する永続ボリュームのデータ	なし	データの保存に使用する永続ボリュームに依存

バックアップ・リストア（環境全体のバックアップ・プロセス）

- 環境全体（インフラ・レベル）でのバックアップは、次のように行うことが推奨されている
  - 初期構築後およびアップグレード後にクラスター全体を停止し、以下のノードのシステム・バックアップを取得する（運用開始後にクラスター全体の停止はサービス継続性の観点で難しいと想定されるため）
    - ブート・ノード
    - etcd ノード / マスター・ノード
    - 管理ノード（必須ではないが、障害時のリカバリー早期化の観点で推奨）
    - VA ノード（必須ではないが。障害時のリカバリー早期化の観点で推奨）
  - 一方で以下のノードは再構築が容易なため、バックアップは必須ではない。ノードのホストファイルシステム上に重要なデータが存在する場合はバックアップを取得する
    - Workerノード
    - Proxyノード
  - 定期運用においてetcd/マスター・ノードのシステム・バックアップを行う場合は、クラスターの冗長性を損なわないよう考慮する（3ノード構成の場合は1台ずつ停止し、バックアップを行う）
    - 参考情報: IBM Cloud Private System Administrator's Guide
      - *Chapter 3.4 Backup and restore strategy*
    - 参考情報: GitHub – ibm-cloud-architecture/icp-backup

バックアップ・リストア（環境全体のバックアップ・プロセス）

■ 環境全体（インフラ・レベル）でのバックアップは、次のように行うことが推奨されている（続き）

– 初期構築後のバックアップを取得する際には、ノードの停止・起動順序に注意する
  - 停止時はMasterノードを先に停止後、他のノードを停止する
  - 起動時は他のノードを起動後、Masterノードを起動する

- 参考情報: GitHub – ibm-cloud-architecture/icp-backup/docs/
  – *Backup and Restore an Entire ICP Topology*

バックアップ・リストア（環境全体のリストア・プロセス）

- 環境全体（インフラ・レベル）でのリストアは、次のように行うことが推奨されている
  - 環境全体のリストア時に、冗長化されたetcd/マスター・ノードをリストアする際には、etcdのリストアに注意が必要
    - etcd/マスター・ノードは最初に起動されたノードがリーダーとなり、そのノード上の最新のデータ（バックアップ取得時点のデータ）を元にノード間の同期を行う
    - etcdの不整合を避けるためにも、システム・バックアップのリストア後にコンポーネント単位でのetcdのリストア手順を実施することが推奨される（コンポーネント単位でのリストアについては、後述する「バックアップ・リストア（コンポーネントのリストア・プロセス）」を参照）
  - 参考情報: IBM Cloud Private System Administrator's Guide
    - *Chapter 3.4 Backup and restore strategy*

バックアップ・リストア（コンポーネントのバックアップ・プロセス）

- コンポーネントのバックアップは、IBM Redbook と GitHub それぞれで示されているバックアップ順序を考慮し、etcd -> Docker Registry -> MongoDB -> MariaDB （オプション）とすることが推奨される
  - IBM Redbook記述：クラスターアクティビティの少ない時間帯に、間隔をなるべく空けずに etcd -> MongoDB -> MariaDB（オプション） の順に行うことを推奨
  - GitHub記述：etcd -> Docker Registry -> MongoDB の順に行う必要がある
  - バックアップデータは永続ボリュームに配置する
  - その他コンポーネントのバックアップ順序は特に推奨はない

  - 参考情報:
    - IBM Cloud Private System Administrator's Guide
      - *Chapter 3.4 Backup and restore strategy*
    - GitHub – ibm-cloud-architecture/icp-backup

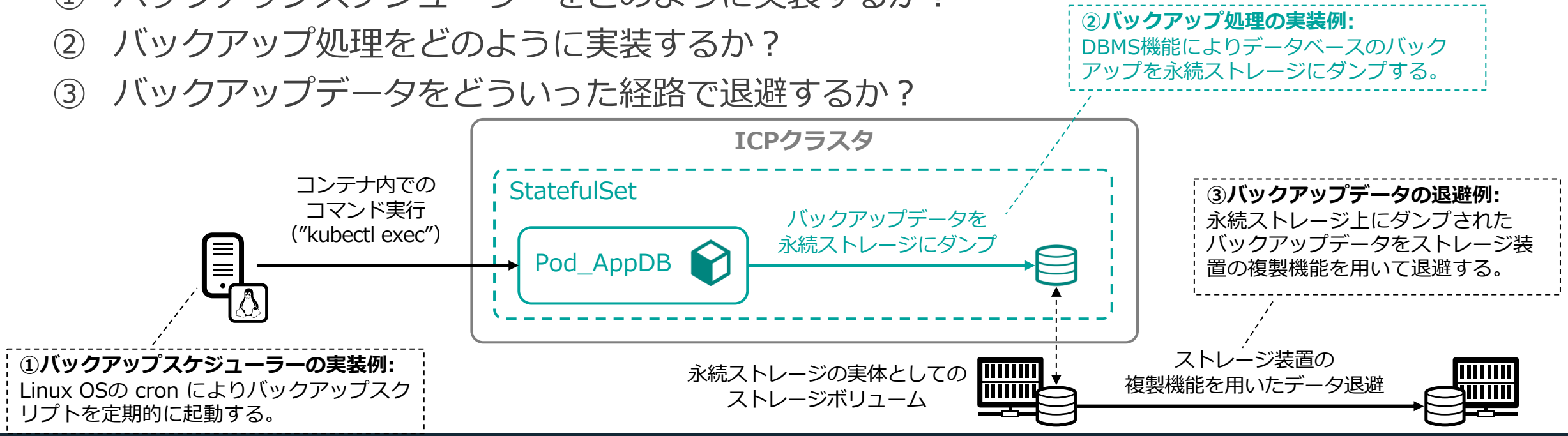
バックアップ・リストア（コンポーネントのリストア・プロセス）

- コンポーネントのリストアは、次のように行うことが推奨されている
  - リストアを開始する前に、環境全体のインフラレベルでのバックアップを行うことが推奨されている
  - また、etcdについてもkubectlコマンドを使用してリストア作業開始前のetcdの状態をバックアップすることが推奨されている（リストア作業は複数ステップにわたり、失敗するリスクが伴うため）
  - 冗長化されたetcd/マスター・ノードを全てリストアする場合は、単一マスター・ノードで取得したetcdのバックアップを全ノードにリストアする
  - データ破損またはデータロスの状態に陥ったetcd/マスター・ノードをリストアする場合は、etcdクラスターからそのメンバーを外し、データ・ディレクトリーの内容を削除してから再度メンバーとして追加する
  - コンポーネントのリストアは、MongoDB -> Docker Registry -> etcd -> 永続ボリュームのデータ の順に行うことが推奨されている
  - 脆弱性アドバイザーはElasticsearchを参照しているため、脆弱性アドバイザーのデータをリストア対象とする場合は、Elasticsearchのロギング・データを先にリストアすることが望ましい
  - その他のコンポーネントについてはリストア順序に指定はない

  - 参考情報:
    - IBM Cloud Private System Administrator's Guide
      - *Chapter 3.4 Backup and restore strategy*
    - GitHub – ibm-cloud-architecture/icp-backup

バックアップ・リストア（業務データのバックアップ・リストア）

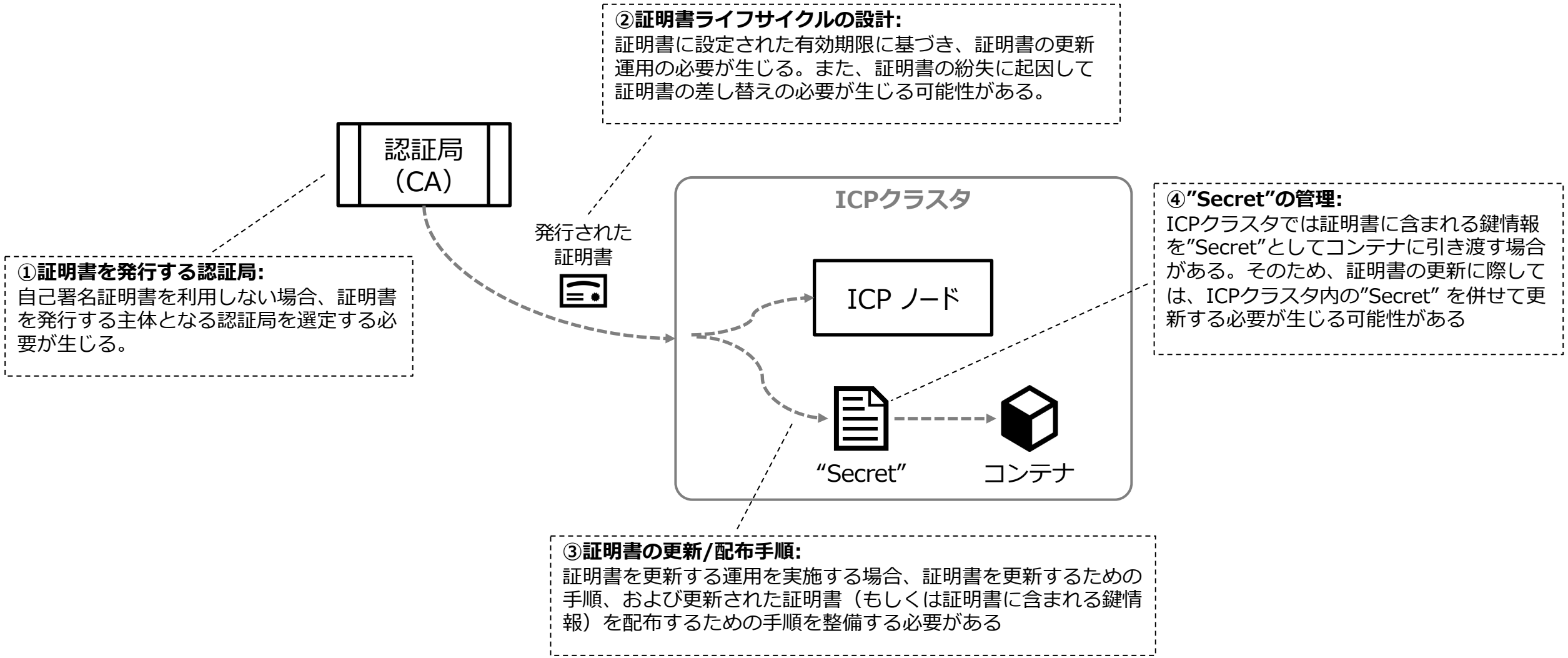
- ICPでは業務データを保管するためのストレージ・ソリューションが複数提供されており、各システムで採用されるソリューションは異なるため、Kubernetes環境上で稼動する業務システムの保管データをバックアップ・リストアするための具体的ソリューションは明示されていない
  - –そのため、業務データのバックアップ・リストア運用を自力で確立する必要が生じる。
- バックアップ運用にあたって考慮すべきポイントは以下の3点
  1. バックアップスケジューラーをどのように実装するか？
  2. バックアップ処理をどのように実装するか？
  3. バックアップデータをどういった経路で退避するか？



第4章 第3節:
証明書管理

ICP基盤運用に関わる証明書の管理

■ ICP基盤運用に関わる証明書の管理運用における考慮事項を以下の概念図に示す



証明書管理（ICP Kubernetes環境での証明書の更新・置き換え）

■ ICP Kubernetes環境で基盤コンポーネントが利用する各種証明書の概要を下表に示す

– 各種証明書の更新・置き換え可否や、具体的手順は IBM Knowledge Centerに記載がある。

証明書種別	証明書更新運用の可否	証明書持ち込み（BYOK*）	解説	対応するKnowledge Center記述
ICPクラスタ管理通信向け	可	一部可	ICPクラスタで行われる各種の管理通信の保護のために用いられる。当該の証明書はICPクラスタのセットアップ時にブート・ノードから各ノードの /etc/cfc/confディレクトリーへ配布される。一部証明書については、インストール時に証明書の持ち込み(BYOK)とインストール後の置き換えが可能である。	IBM Knowledge Center – IBM Cloud Private の証明書 IBM Knowledge Center – IBM Cloud Private サービス用の独自証明書の指定 IBM Knowledge Center – 証明書の置き換え
ノード間SSH認証向け	可	可	ブート・ノードから各ノードへのSSH認証のために用いられる。	IBM Knowledge Center – クラスターノードでのSSH鍵の共有
ノード間IPSec接続向け	可	可	ICPクラスタに属するノード間の通信をIPSec暗号化する場合に用いられる。証明書差し替えの手順がKnowledge Centerに明記されている。	IBM Knowledge Center – IPSecを使用したクラスター・データ・ネットワーク・トラフィックの暗号化
SSL LDAP接続向け	可	連携先LDAPサービスの仕様に依存する	ICPクラスタが参照するLDAPサービスへの通信を保護するために用いられる。証明書をICPクラスタにSecretとして登録する手順がKnowledge Centerに明記されている。	IBM Knowledge Center – LDAP接続の構成

BYOK* : Bring Your Own Keyの略。選定した認証局によって発行された証明書を持ち込むこと。

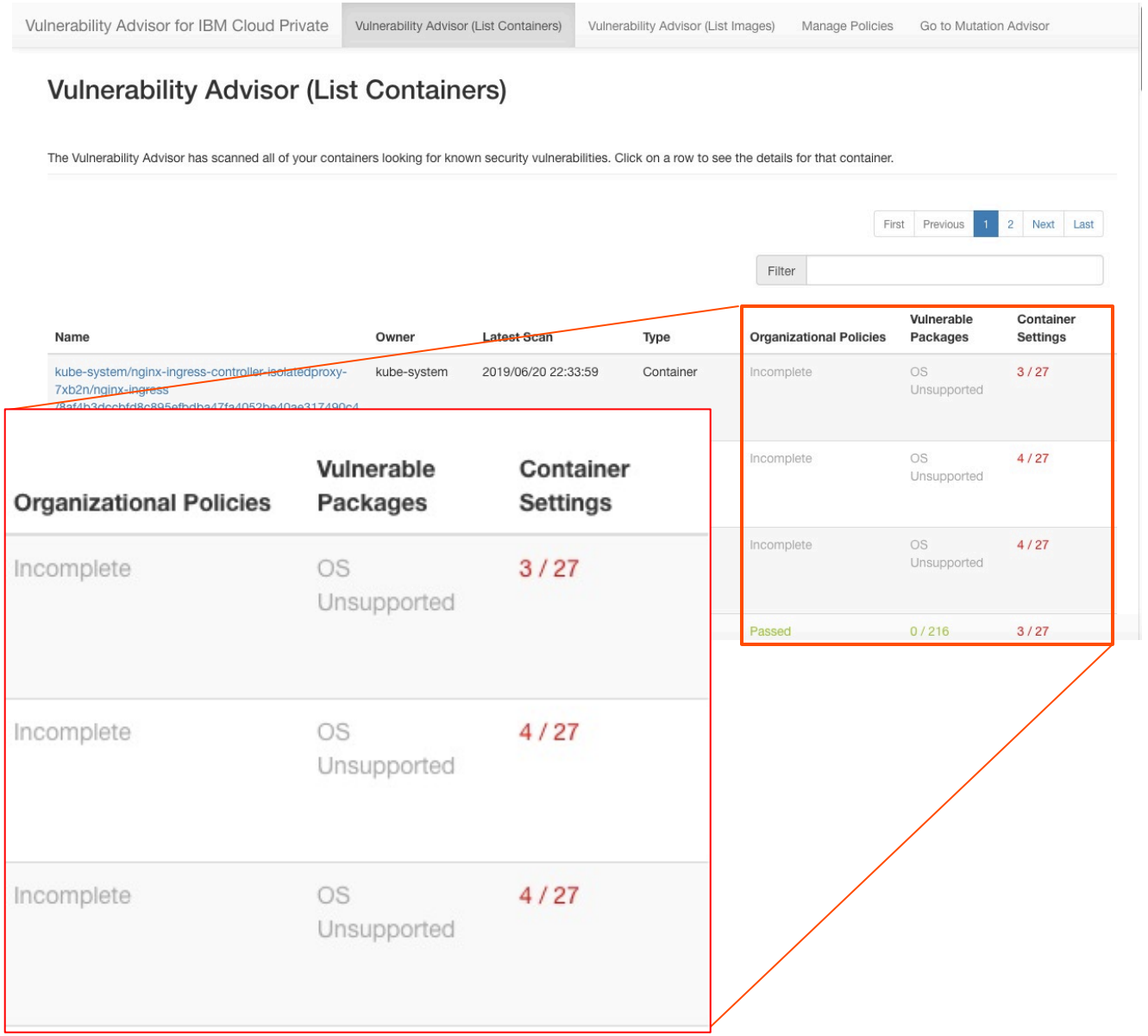
第4章 第4節:
脆弱性対応

脆弱性対応（脆弱性アドバイザーとは）

- **製品組み込みのコンテナ脆弱性評価ツール**
  - 脆弱性評価は以下が対象
    - ICPのプライベート・レジストリに配置されたイメージ
    - ICP上で実行中のコンテナ
  - IBM Cloud Container Registry（パブリックのIBM Cloud Container Serviceで使用されるイメージ・レジストリ）と同様の脆弱性評価機能をオンプレミスで実行可能
- **動作要件**
  - 1,3,5台のVA専用ノードが必要(3台または5台でHA構成)
  - ICP Cloud Native/Enterprise Editionが必要(CE版不可)
  - ノードの必要スペック
    - CPU　　：8コア以上、2.4GHz以上
    - メモリ　　：16GB以上
    - ディスク容量　　：800GB以上
    - OS　　：ICPに準拠

(参考) Knowledge Center – Hardware Requirements and recommendations

※PowerVM環境, Linux on IBM Z, LinuxONE の場合は要件が異なる（参考リンク参照）

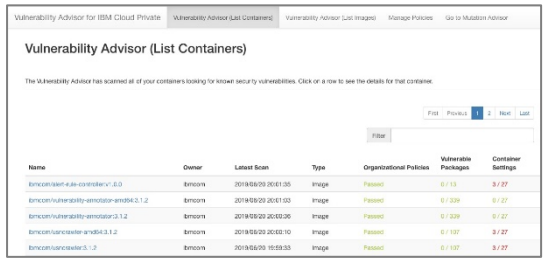


脆弱性対応（セキュリティ・レポート）

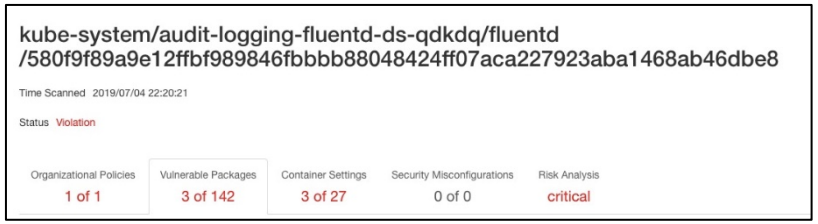
■ セキュリティ・レポート

（ICP UI）> ツール > 脆弱性アドバイザー > (namespace)

– コンテナ・イメージの脆弱性評価の一覧を表示



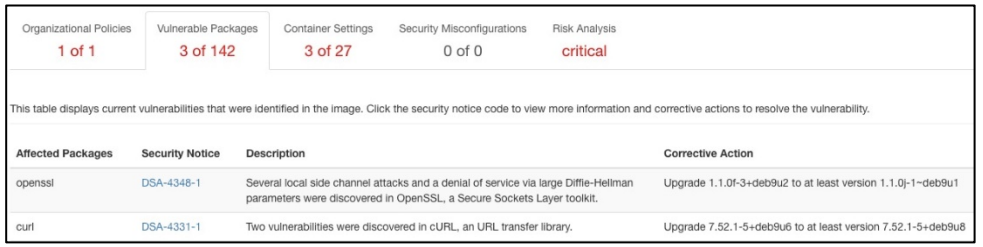
– コンテナまたはイメージをクリックして詳細を確認



Organizational Policies	• Manage Policyで設定した検査項目の脆弱性評価
Vulnerable Packages	• コンテナまたはイメージ内で使用されているパッケージの、既知の脆弱性に関する脆弱性評価
Container Settings	• コンテナまたはイメージ内で設定されているパスワードの長さ等、推奨設定の脆弱性評価
Security Misconfigurations	• アプリケーションにおけるセキュリティー設定の脆弱性評価
Risk Analysis	• コンテナイメージのリスク分析結果

■ 脆弱性が検出された場合の対応

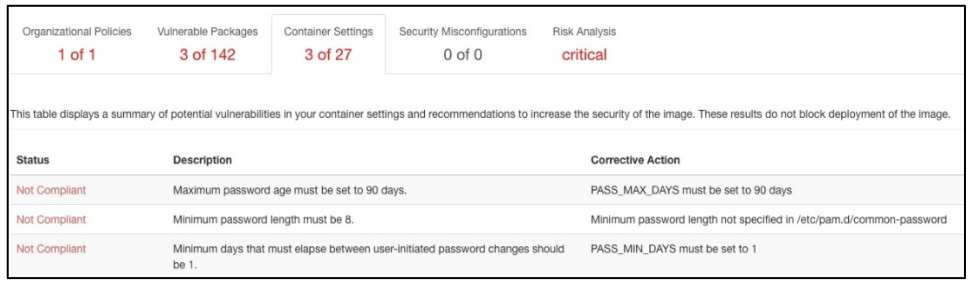
（Vulnerable Packages）



– 脆弱と判定されたパッケージのSecurity Noticeを確認し、脆弱性が解消されたバージョンのパッケージに置き換える、またはパッケージの置き換えられたコンテナ・イメージに更新する

– レポートされたパッケージの脆弱性がアプリケーションとして使用されないことを確認する

（Container Settings）



– "Not Compliant" と検出された項目を参照、評価し、必要な場合はコンテナおよびイメージの設定を修正する

(参考) IBM Cloud Docs – 脆弱性アドアイザーを使用したイメージ・セキュリティーの管理

脆弱性対応（脆弱性アドバイザーのシステム構成）

■ 動作コンポーネント

– VAノード上で動作する主要なコンポーネント

Security Analytics Service (SAS) components	Vulnerability Advisorダッシュボード及びSAS APIを提供
Kafka/minio	VA用の分析ログ検索、取込、分類
VA Annotators	脆弱性分析、コンプライアンス・チェック、パスワード分析、構成分析、rootkit検出を実行
VA Usncrawler	外部のsecurity noticeを取り込み。
Registry crawler	コンテナ・イメージから脆弱性評価のための情報を抽出する

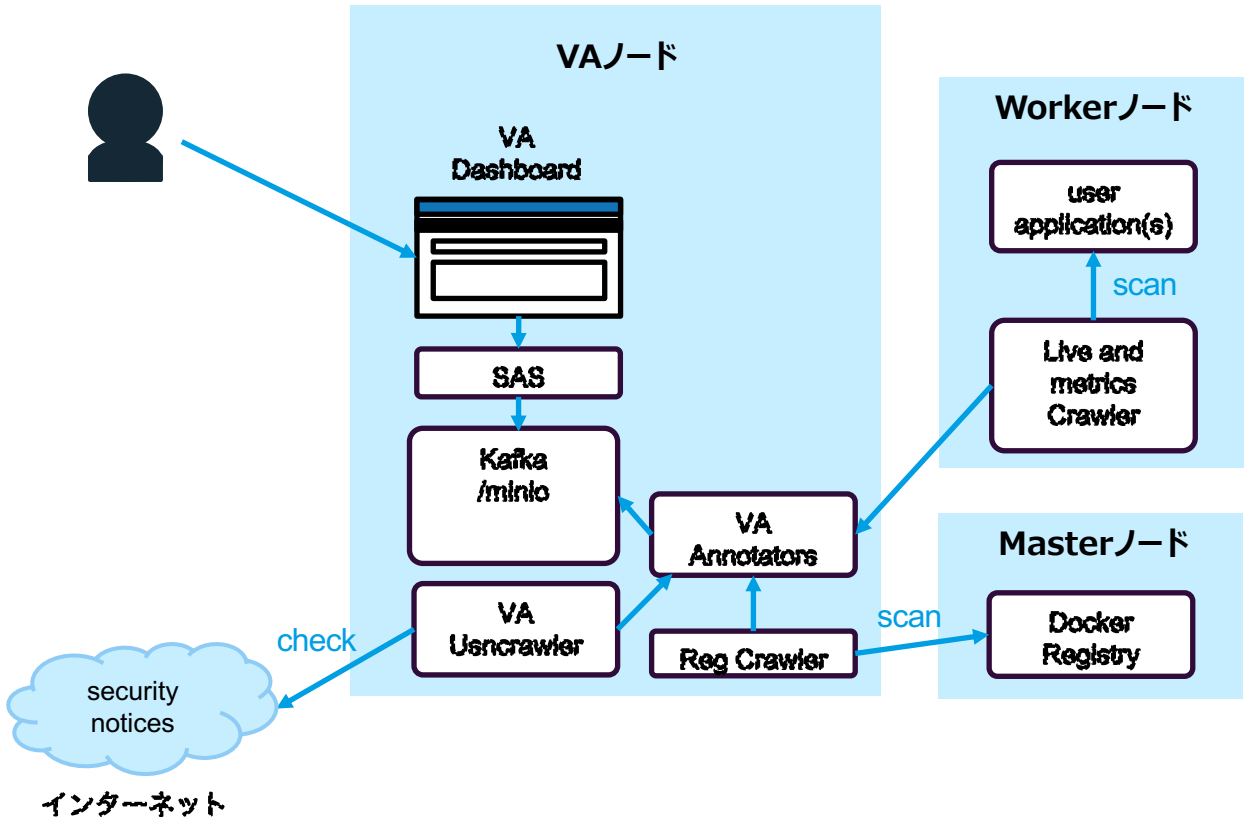
– その他のノード上で動作するコンポーネント

Live and metrics crawler	実行中のコンテナから脆弱性評価のための情報を抽出する

(参考) Knowledge Center - IBM Cloud Private コンポーネント

■ 脆弱性のスキャン周期

- 実行中のコンテナ　　：１日（86400秒、デフォルト）
- コンテナ・イメージ　：ほぼリアルタイム（イメージpush後数分内）



脆弱性対応（脆弱性アドバイザーの構成）

■ VAの導入

（ICPインストール時）

– cluster/hostsファイルに1,3,5台のvaノードを指定する
– config.yaml の management_services セクションで “vulnerability-advisor : enabled” とする
– 複数VAノードの場合は、各VAノード上の ”/var/lib/icp/va/minio”ディレクトリーを共有ストレージに配置する

（参考）脆弱性アドバイザーの使用可能化

■ VA導入後のカスタマイズ（主な設定）

（ICP UI）＞ 「構成」 ＞ ConfigMaps

vulnerability-advisor-live-crawler	• 実行中コンテナのcrawler有効・無効を指定 • crawlの頻度を指定（デフォルト: 86400秒 = 1日）
vulnerability-advisor-reg-crawler	• コンテナ・イメージのcrawler有効・無効を指定 • 有効とした場合の、再スキャンを設定

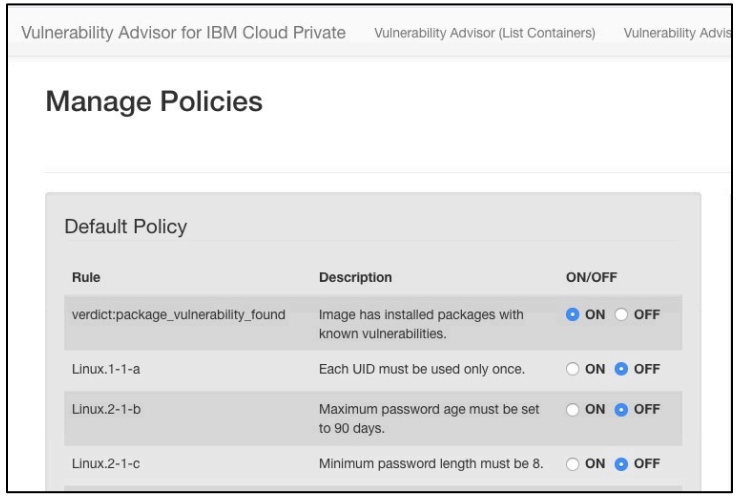
スキャンデータの保持期間など他の設定については、下記リンクを参照

（参考）脆弱性アドバイザー

■ VAのManage Policies設定

（ICP UI）＞ ツール ＞ 脆弱性アドバイザー ＞ (namespace) ＞ Manage Policies

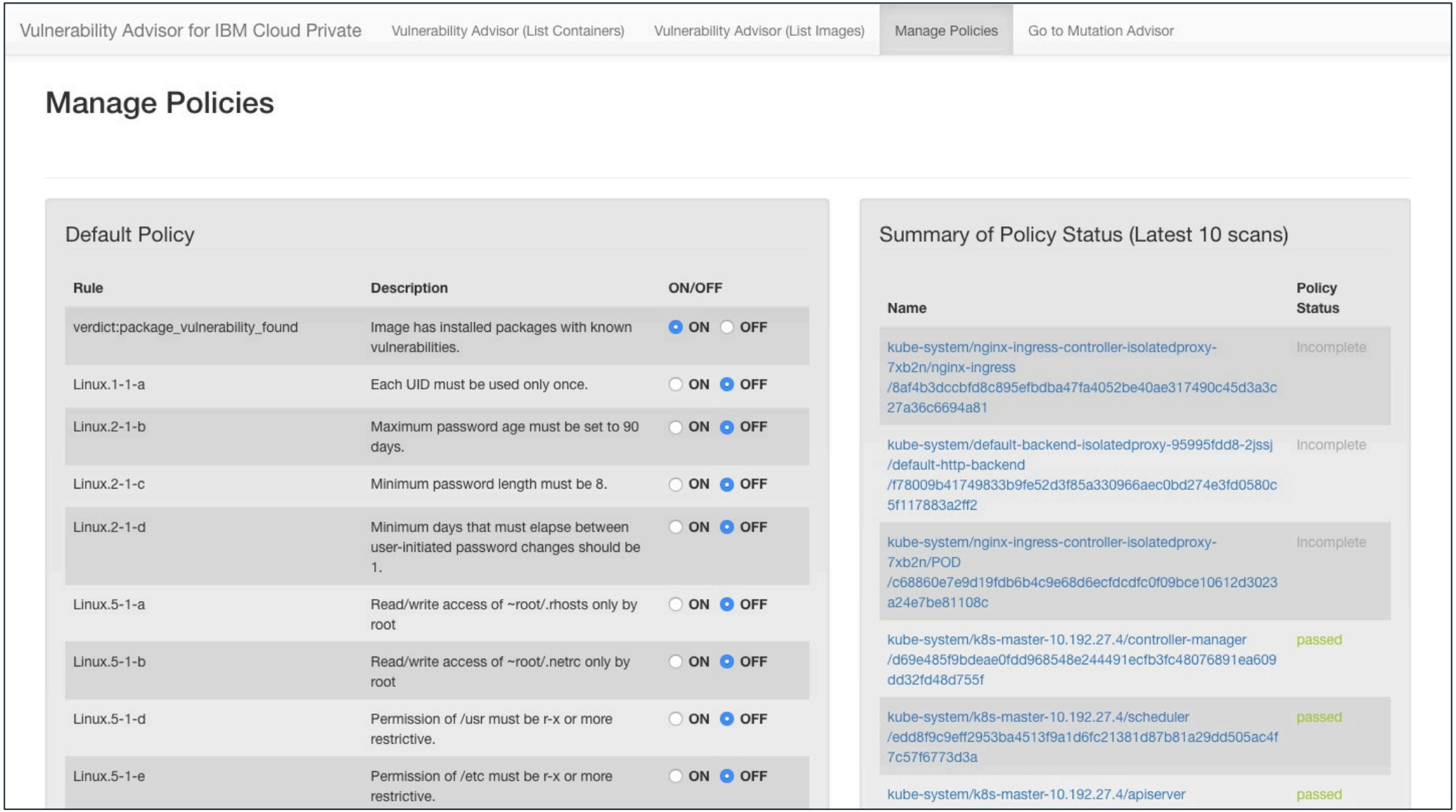
– ネームスペース毎に、crawlerがチェックする項目を設定



(設定項目の一部抜粋)

verdict:package_vulnerability_found	• コンテナ・イメージに既知の脆弱性が含まれるかをチェック（デフォルト: ON）
Linux.2-1-b	• パスワード有効期限が90日であることをチェック（デフォルト: OFF）
Linux.2-1-c	• パスワードの長さが8文字以上であることをチェック（デフォルト: OFF）
Linux.9-0-a	• sshサーバーがインストールされているかをチェック（デフォルト: OFF）
verdict:ssh_installed	• コンテナ・イメージにsshサーバーがインストールされているかをチェック（デフォルト: OFF）

脆弱性対応（Manage Policyの項目一覧）（その1）



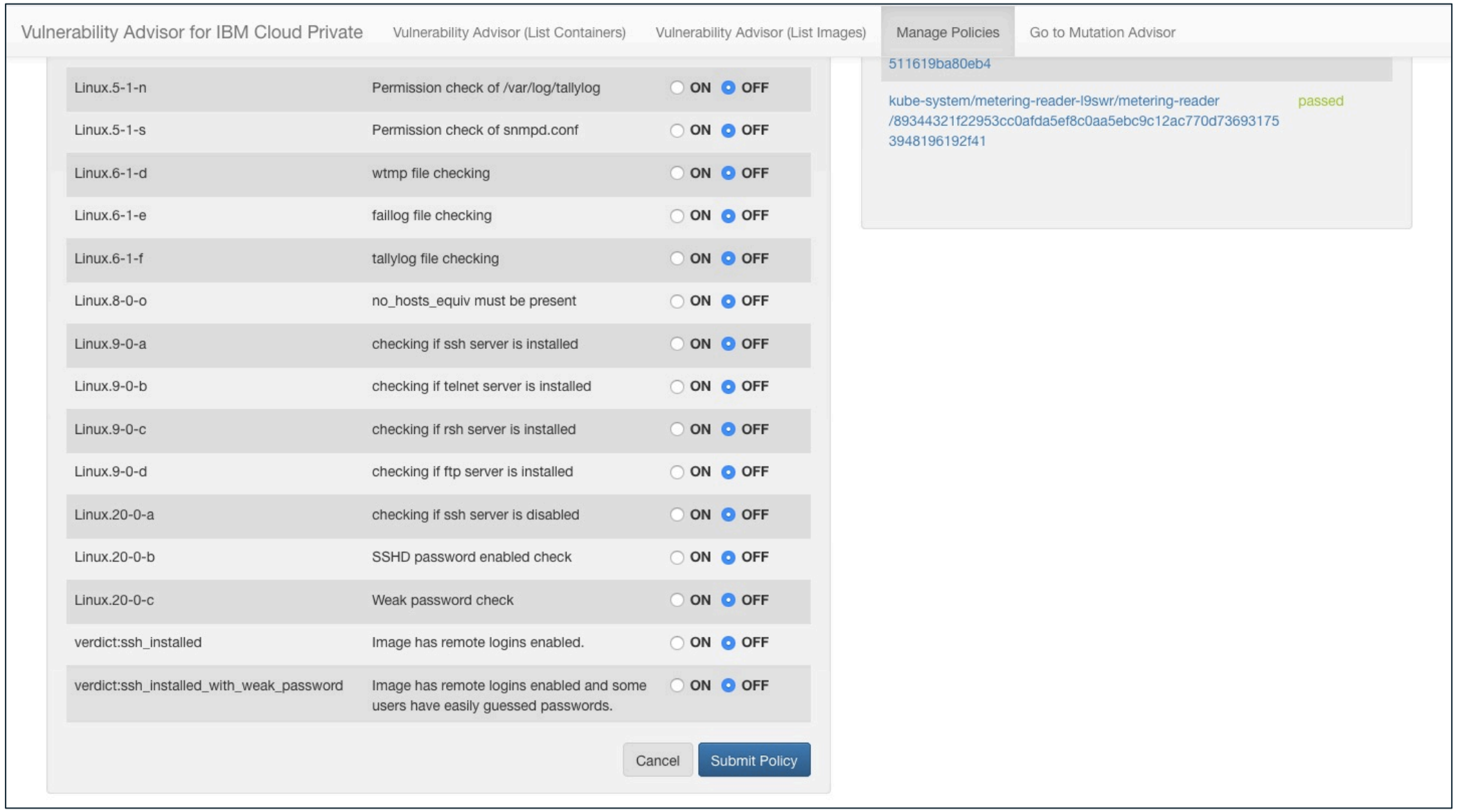
脆弱性対応（Manage Policyの項目一覧）（その2）

Vulnerability Advisor for IBM Cloud Private | Vulnerability Advisor (List Containers) | Vulnerability Advisor (List Images) | Manage Policies | Go to Mutation Advisor

Linux.5-1-e	Permission of /etc must be r-x or more restrictive.	ON OFF
Linux.5-1-f	The file /etc/security/opasswd must exist and the permission must be rw------- or more restrictive.	ON OFF
Linux.5-1-j	Permission settings of /var for other must be r-x or more restrictive.	ON OFF
Linux.5-1-k	Permission of /var/tmp must be rwxrwxrwt.	ON OFF
Linux.5-1-l	Permission setting of /var/log for other must be r-x or more restrictive.	ON OFF
Linux.5-1-m	Permission check of /var/log/faillog	ON OFF
Linux.5-1-n	Permission check of /var/log/tallylog	ON OFF
Linux.5-1-s	Permission check of snmpd.conf	ON OFF
Linux.6-1-d	wtmp file checking	ON OFF
Linux.6-1-e	faillog file checking	ON OFF
Linux.6-1-f	tallylog file checking	ON OFF
Linux.8-0-o	no_hosts_equiv must be present	ON OFF
Linux.9-0-a	checking if ssh server is installed	ON OFF
Linux.9-0-b	checking if telnet server is installed	ON OFF
Linux.9-0-c	checking if rsh server is installed	ON OFF
Linux.9-0-d	checking if ftp server is installed	ON OFF

7c57f6773d3a	
kube-system/k8s-master-10.192.27.4/apiserver/9dea4a69836e3b3eba58bd9a65c75278392134a0a8b70530e34b6a4fe071a193	passed
kube-system/tiller-deploy-64458f7ff4-n9fc9/tiller/90b0be937f3e4c26bbfe92dcc25f3d17fd5112d489b9ee956ed881777523921d	passed
kube-system/tiller-deploy-64458f7ff4-n9fc9/POD/61db1844862b4eb95c6ccbb89fc3a4af66e54f5de18619fc10800fd6e0e65cd5	Incomplete
kube-system/monitoring-prometheus-nodeexporter-b5ksf/router/4522f2d0fe573793a3eaeb08840b16226faf0c2ca45d6a89de6511619ba80eb4	passed
kube-system/metering-reader-l9swr/metering-reader/893443321f22953cc0afda5ef8c0aa5ebc9c12ac770d736931753948196192f41	passed

脆弱性対応（Manage Policyの項目一覧）（その3）



第4章 第5節:
ノード追加・削除

ノード追加・削除（ノード追加・削除のサポート状況）

- IBM Cloud Private製品のKnowledge Center記述上で、ノードの追加手順が明記されているノード種別は以下の通り
  - ワーカー・ノード（ホスト・グループ含む）
  - 管理ノード
  - プロキシー・ノード（HA構成であることが前提）
  - 脆弱性アドバイザー・ノード
    - 参考情報: IBM Knowledge Center – IBM Cloud Private クラスター・ノードの追加

- IBM Cloud Private製品のKnowledge Center記述上で、ノードの削除手順が明記されているノード種別は以下の通り
  - ワーカー・ノード
  - 管理ノード
  - プロキシー・ノード
    - 参考情報: IBM Knowledge Center – IBM Cloud Private クラスター・ノードの削除

ノード追加・削除（ノード追加・削除の手順）（その1）

■ ノード追加・削除の手順として、2通りの方式が提供されている

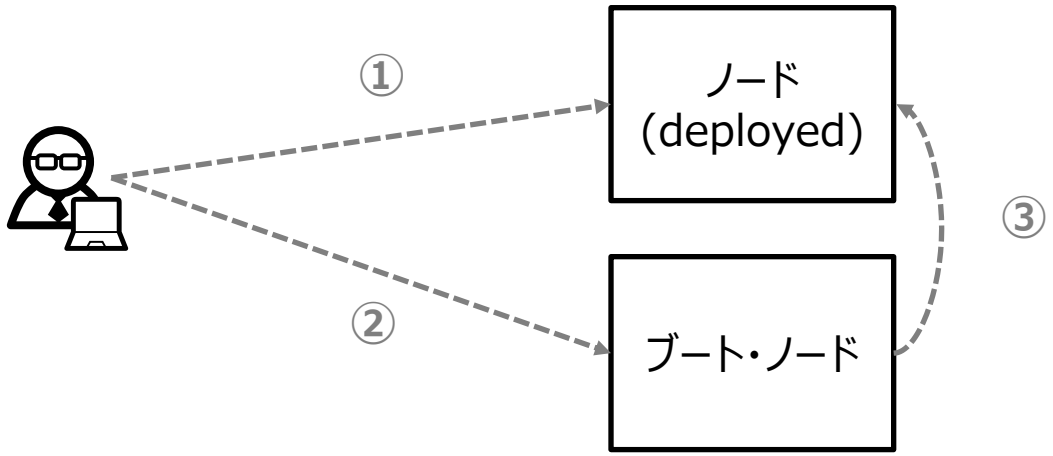
A) ブート・ノード配置のインストーラーを用いた方式

- 参考情報: IBM Knowledge Center – IBM Cloud Private クラスター・ノードの追加
- 参考情報: IBM Knowledge Center – IBM Cloud Private クラスター・ノードの削除

① 運用担当者はコンピューティング・インスタンス（=Linux OSがセットアップされた仮想マシン）を払い出す

② 運用担当者は払い出されたコンピューティング・インスタンスをICPクラスタのノードとしてセットアップまたは削除するための指示をブート・ノードに配置されたICPソフトウェアのインストーラーに対して行う

③ ICPソフトウェアのインストーラーは払い出されたコンピューティング・インスタンスをセットアップまたは削除する（削除はワーカー・ノード、プロキシー・ノード、管理ノードのみ）



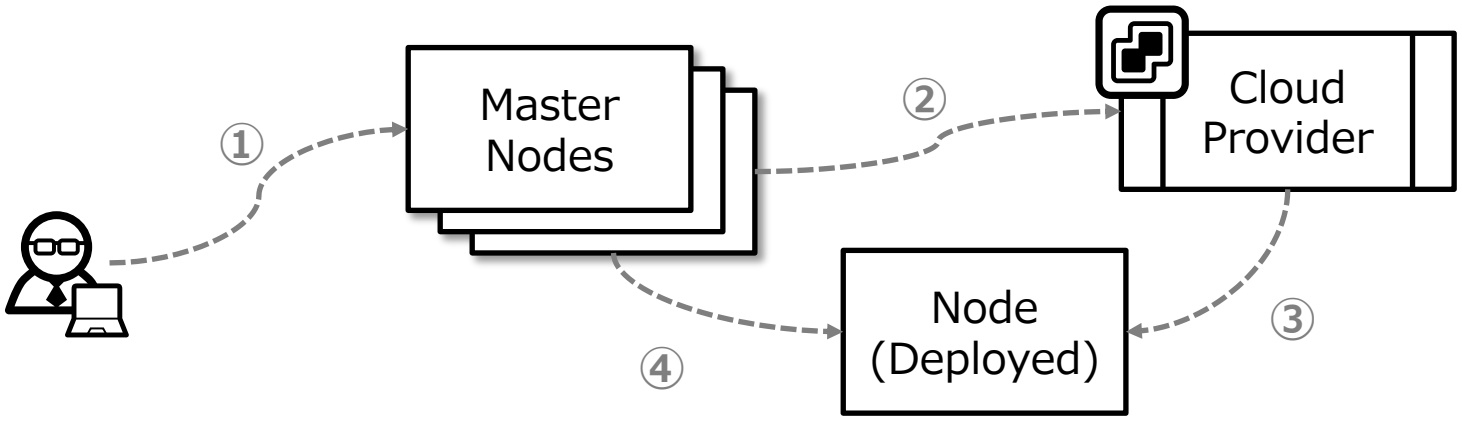
ノード追加・削除（ノード追加・削除の手順）（その2）

■ ノード追加・削除の手順として、2通りの方式が提供されている

B) クラウド・プロバイダー連携方式

- OpenStackおよびVMWare利用プロバイダーが前提
- クラウド・プロバイダーへの接続設定が必要
- プロキシ・ノードおよびワーカー・ノードの追加・削除が可能
- 参考情報: IBM Knowledge Center – OpenStackまたはVMWareを使用したクラスター・ノードの追加または削除

① システム管理者は、ICPクラスタ（=マスター・ノード）に対して、ノードの追加を指示する
② ICPクラスタはコンピューティング・インスタンスの払い出しをクラウド・プロバイダー（VMware vSphere or OpenStack）に指示する
③ 指示に従い、クラウド・プロバイダーはコンピューティング・インスタンスを払い出す
④ ICPクラスタは、払い出されたコンピューティング・インスタンスをセットアップする



ノード追加・削除（ノード追加・削除の手順）（kubectl CLI）

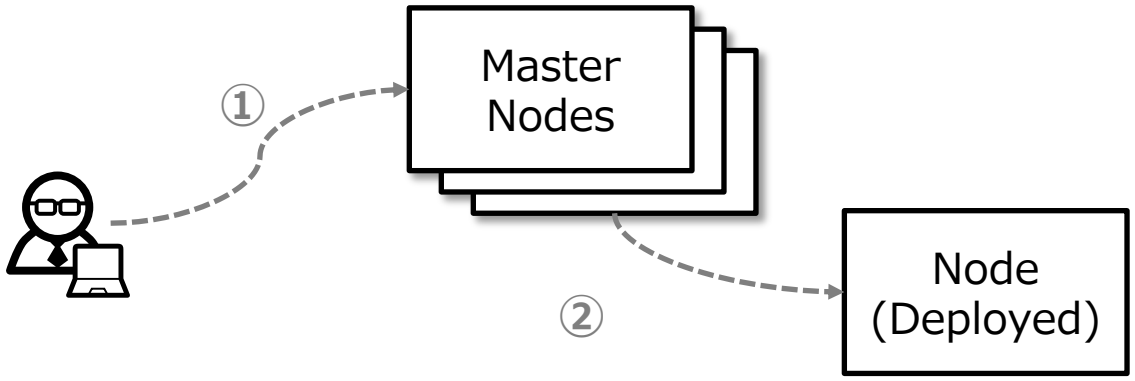
■ また、前述A)、B)のいずれの手順も利用できない場合のノード追加・削除の手順として、次の手順が提供されている

kubectl CLIを用いた方式

- 応答しない IBM Cloud Private クラスター・ノードを削除する方法として提供されている
- 参考情報: IBM Knowledge Center – 応答しない IBM Cloud Private クラスター・ノードの削除

① システム管理者は、kubectl を使用してノードの削除を指示する。

② ICPクラスタからノードが削除される



第4章 第6節:
管理サービスの有効化・無効化

管理サービスの有効化・無効化

- IBM Cloud Private製品のインストール時に有効化される管理サービスは以下の通り
  - custom-metrics-adapter
  - image-security-enforcement
  - service-catalog
  - Metering
  - Monitoring
- 一方で、製品インストール後は無効化されており、利用する場合は有効化が必要な管理サービスは以下の通り
  - Istio
  - vulnerability-advisor
  - storage-glusterfs
  - storage-minio
- これらの管理サービスは要件に応じて有効化・無効化が可能
  - 参考情報: IBM Knowledge Center – IBM Cloud Private 管理サービスの有効化と無効化

管理サービスの有効化・無効化手順

- config.yaml ファイルを編集し、アドオン・コマンドで有効化・無効化する
  - 無効にするサービスを指定するには、config.yaml ファイル内の management_services パラメーター・リストにサービスを追加し、”disabled”とする
  - 有効にするサービスを指定するには、config.yaml ファイル内の management_services パラメーター・リストからサービスを削除するか、サービスを追加して “enabled”とする
    - /<ICPインストールディレクトリー>/cluster/config.yaml の編集例

```
management_services:
  istio: disabled
  vulnerability-advisor: enabled
  storage-glusterfs: disabled
  storage-minio: disabled
  platform-security-netpols: disabled
  node-problem-detector-draino: disabled
  multicluster-hub: disabled
  multicluster-endpoint: disabled
```

<= 脆弱性アドバイザーを有効にしている例

  - 参考情報: IBM Knowledge Center – IBM Cloud Private 管理サービスの有効化と無効化

第4章 第7節:
etcdの管理

etcdの管理

- etcdはICP Kubernetes環境のリソースの永続化に使用される重要コンポーネントであり、その保守作業としてIBM Knowledge Center では以下の作業が示されている
  - スペース割当量（容量制限）の設定
    - etcdで使用するディスク割当量を設定する（デフォルトは2GB）
    - インストール後に変更する場合はマスター・ノードの /etc/cfc/pods/etcd.json ファイルを編集
  - 履歴の圧縮
    - 指定期間以前のetcdへの変更操作の履歴を圧縮する（デフォルトでは5分おきに圧縮）
    - 変更する場合はマスター・ノードの /etc/cfc/pods/master.json ファイルを編集
  - デフラグ
    - 履歴の圧縮によって発生した断片化を解消する
    - デフラグはetcdctl defragコマンドで手動実行するか、デフラグ用のジョブを作成し定期的に自動実行する（IBM Knowledge Center ではデフラグを自動化する手順が示されている）
    - デフラグ中はデータの読み取り・書き込みが制限されるため、メンテナンス時間中に実施することが推奨されている
  - 参考情報：IBM Knowledge Center – etcd クラスターの管理
  - 参考情報：etcd version 3.2.17 – etcd operations guide - Maintenance

第4章 第8節:
ソフトウェア・アップデート

ICPソフトウェアのアップグレード（アップグレード対象）

■ ICP Kubernetes環境におけるアップグレード対象となるソフトウェアは下表の通り分類出来る。

–アップグレード対象となるソフトウェア毎にアップグレード運用の計画を立案すべき

アップグレード対象ソフトウェア	アップグレード手順リンク（ICP v3.1.2 の場合）	備考/考慮点
ICP製品（Kubernetes環境システム基盤）	IBM Knowledge Center – IBM Cloud Private のアップグレード	ICP バージョン 3.1.0 および 3.1.2 からのアップグレードパスのみサポート バージョン 2.1.0.3 以前のバージョンは最初にバージョン 3.1.0へアップグレードする必要がある
バンドルされたソフトウェア製品（Helmチャート）	IBM Knowledge Center – バンドル製品のアップグレード	ICP製品にバンドルされるソフトウェア製品（主にミドルウェア製品）に相当する。
GlusterFS	IBM Knowledge Center - GlusterFS のアップグレード	ICP 3.1.1 のクラスターインストール時にGlusterFSを構成した、または ICP 3.1.1 インストール後にアドオン・コマンドを使用して GlusterFSを構成した場合は、自動的にアップグレードされる 上記以外の場合、GlusterFSのアップグレードはICP製品のアップグレードとは別に行う

ICPソフトウェアのアップグレード（アップグレードのアプローチ）

- ICPソフトウェアのアップグレードアプローチとして、次の2つが考えられる
  - 導入済みのICPソフトウェアを既存ハードウェア上でアップグレードする（インプレース・アップグレード）
    - メリット：既存ハードウェア・リソースを有効活用できる
    - 考慮点：アップグレード失敗のリスクに備え、フォールバックプランをよく検討する必要がある。移行元のバージョンにより、２段階のアップグレードが必要となる場合がある
  - 新規クラスターを新バージョンで構築し、ワークロードを移行する
    - メリット：既存環境を温存しながらアップフレードが可能。移行元のバージョンは問われない。
    - 考慮点：オンプレミス環境の場合、追加のハードウェアが必要となる
  - プロジェクトの要件や許容可能リスクを考慮し、適切なアプローチを選択する

ICPソフトウェアのアップグレード（アップグレード時の制約事項）

- ICPソフトウェアのインプレース・アップグレードに伴う制約事項
  - サポートされるアップグレード・パス
    - ICP v3.1.1 から v3.1.2へのアップグレード
    - ICP v3.1.0 から v3.1.2へのアップグレード
    - 上記以外のバージョンからのアップグレードを行う場合は、V3.1.0へアップグレード後に3.1.2へアップグレードする
  - 次のアップグレード・シナリオにおいて一時的にアプリケーションへのアクセスが遮断される可能性がある
    - kube-dnsのアップグレード
    - 外部のロード・バランサーを使用して Ingress Controller へリクエストをルーティングしている場合、ロード・バランサーによるヘルスチェックが失敗する可能性がある（外部ロード・バランサーと Ingress Controller 間の接続が確立されるとリクエストを受け付けられるようになる）
  - アップグレード中は ICP管理コンソールへはアクセスできない
  - クラウド・プロバイダー・オプションの設定ができない（ vSphrere の構成、NSX-Tの使用など）
    - 参考情報：IBM Cloud Private のアップグレード
- ICPソフトウェアのロールバック（”revert”）に関わる制約事項
  - バージョンアップ後にアプリケーションをデプロイしてしまうとロールバックが失敗する可能性がある
    - 参考情報: IBM Knowledge Center – 前のバージョンの IBM Cloud Private への復帰

第4章 第9節:
モニタリング

ICPにおけるモニタリングとは？

- リソース使用率などのデータの収集、数値の表示、グラフの表示、アラートを通知すること
- ICPで提供されているモニタリング用のコンポーネント
  - ICP Dashboard
    - k8s API経由でクラスタ内のリソース状況を取得
  - Prometheus
    - データの収集、数値の表示、アラート通知を実施

    *本資料では、AlertmanagerはPrometheusに含まれる
  - Grafana
    - Prometheusが収集した情報に対して、グラフ表示を実施

ICP Dashboard

- ICP Dashboard(ICPコンソールのトップ画面)上で、現在のシステムとリソースの状況を確認可能
  - 参照できるのはクラスタ管理者ロールのユーザーのみ
  - 画面構成
    - システムの概要
      - *クラスタ内の稼働中のノードの数、共有ストレージの使用率、稼動しているアプリケーションの数*
    - リソースの概要
      - *クラスタ内のリソース(CPU、Memory、GPU)の使用状況*

参考情報: [IBM Knowledge Center - IBM Cloud Private クラスター・モニタリング]

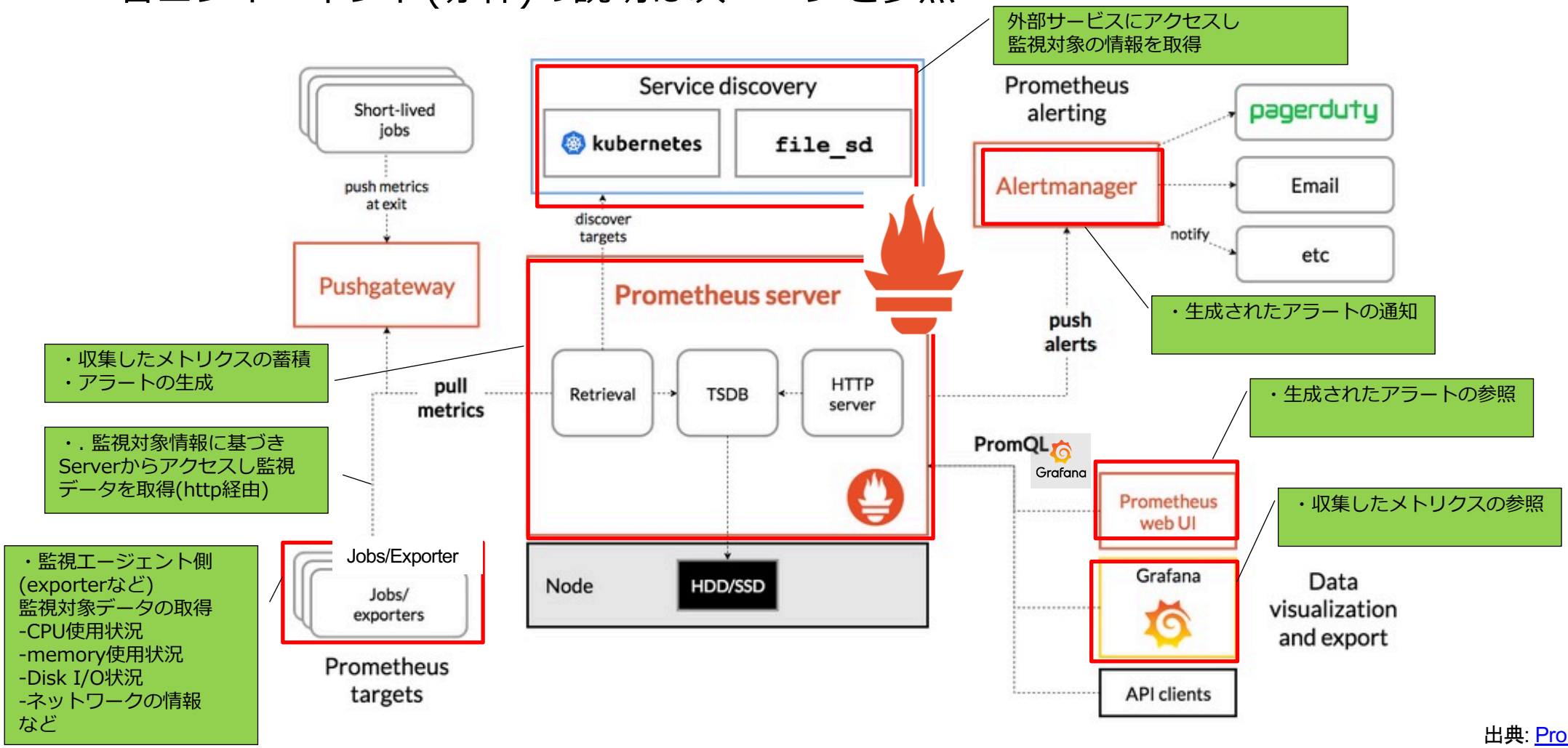
Prometheusとは? (1)

- Google社内の監視ツールBorgmonの思想を取り入れ、開発されたOSSツール
- 特徴
  - 構築の容易性
    - インストールはバイナリ配置のみ。データベース/プラグインのインストール不要
  - Auto Scaling対応
    - 監視対象の増減に柔軟に対応
  - 柔軟な可視化
    - 多次元データの取り扱い(Grafanaの利用)
  - 不要なアラートを抑止
    - 緊急性の高いものだけを通知
  - Pull型監視
    - Prometheus Serverが、各ノードのexporterに対して情報を取得
  - コンテナ/クラスタ管理ツールとの連携
    - DockerやKubernetesと連携し、監視対象を追加
  - 参考資料：Prometheus.io - Overview

Prometheusとは? (2)

■アーキテクチャー

–各コンポーネント(赤枠)の説明は次ページを参照



出典: [Prometheus.io - Overview]

Prometheusとは? (3)

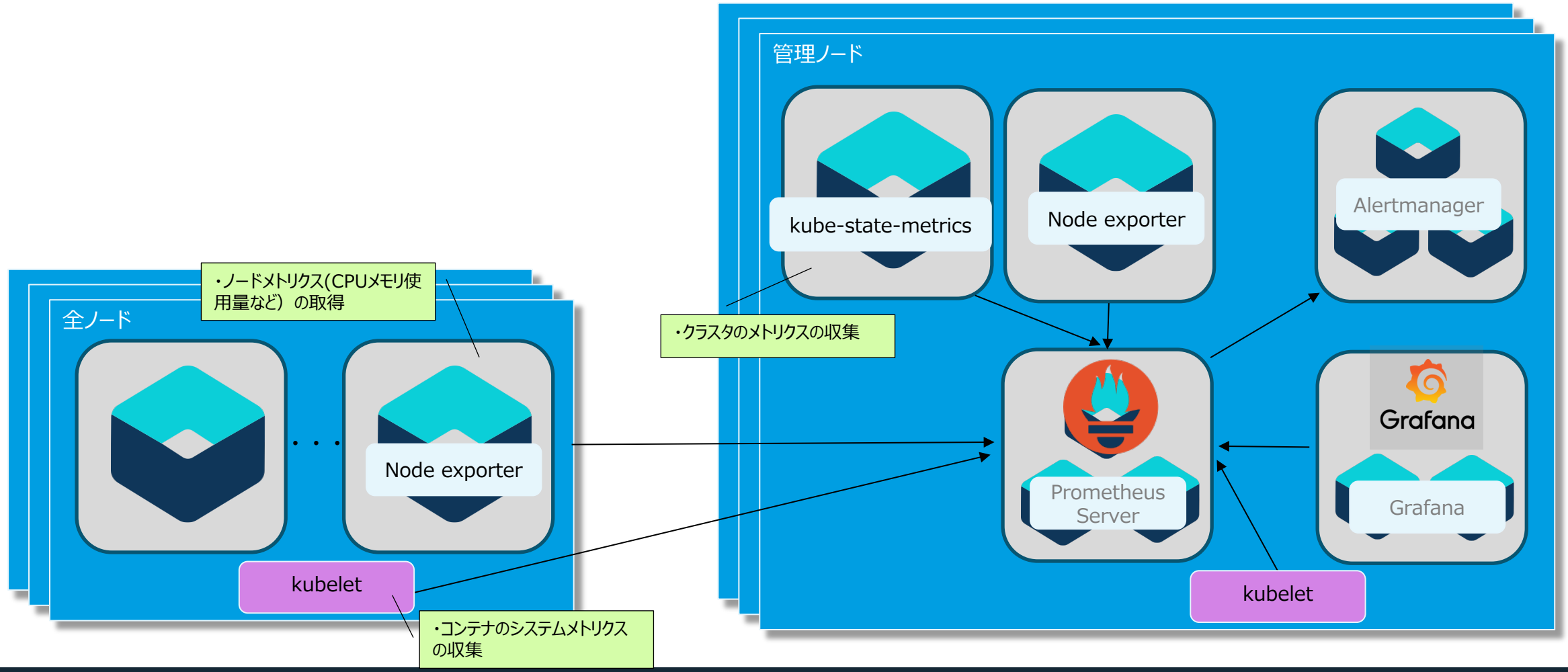
■ Prometheusに関係する主要なコンポーネント

コンポーネント名	概要
Prometheus Server	サーバーコンポーネントで、監視データ(メトリック)の収集、アラートの生成を行う。
Exporter	監視データをPrometheus Serverに送るためのコンポーネント。一般的な監視エージェントとは異なり、データを監視サーバーに自ら送信しない。サーバーのリソースを収集して送信する「Node/system metrics exporter」やMySQLサーバーの情報を収集・送信する「MySQL exporter」など、Linux用、各種ミドルウェア用のExporterが存在する。
Alertmanager	Prometheusが生成したアラートを同種類にまとめ、Email、Slack 、Webhook等に通知する。
Service Discovery	監視対象を発見する機能。Prometheusのconfigファイルで利用する発見方法を指定/設定する。 ・AWS、GCP、Azureのようなcloud service ・Kubernetes、Consulのようなツール ・File、DNS機能
Grafana	高機能な可視化ツール。開発元はPrometheusとは異なる。

ICP k8s環境 – Prometheus関連のコンポーネント

■ ICP k8s環境におけるPrometheus関連のコンポーネント配置図は以下となる

–ICP導入後、各コンポーネントは起動済



アプリケーションのメトリクス収集

- この資料では、アプリケーションとは、データベースやHTTPサーバーといったソフトウェア、ミドルウェアが稼働するコンテナ/Podを指す
- ICPのデフォルト状態において、アプリケーションのメトリクス収集は実装されていない
  - 収集する場合、別途方法の検討が必要
- 対象のアプリケーションの状態によって方法を選択
  - ICPのカタログに登録されている、「 ibm-icpmonitoring」を利用して、アプリケーション用のモニタリング・サービス(Prometheus、Grafana)をデプロイ
  - アプリケーションが Prometheus 形式のメトリクスのエクスポートをサポートしている場合はそのまま利用
  - Prometheusのクライアントライブラリを利用して実装
  - メトリクスのエクスポートをサポートしていない場合、別途、exporterを導入
    - <参考>用意されているexporterの一覧(100種類以上)

    Prometheus - Exporters and integrations

k8s環境でのメトリクス収集方法のまとめ

■ K8s環境の各レイヤーでメトリクスを取得するためのツールや手法は以下となる

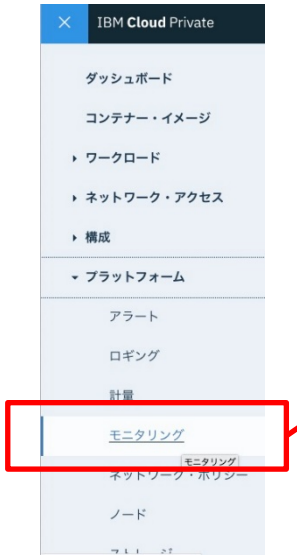
対象レイヤー	メトリクス取得のためのツール・手法
システム (ホストマシン)	Node exporter
システム (コンテナ)	cAdvisor (Kubelet)
k8s クラスタ	kube-state-metrics
アプリケーション	・ICPのカタログに登録されている、「 ibm-icpmonitoring」を利用 ・アプリケーションがprometheusのメトリクスエクスポート対応 ・クライアントライブラリを利用 ・別途、exporterを導入

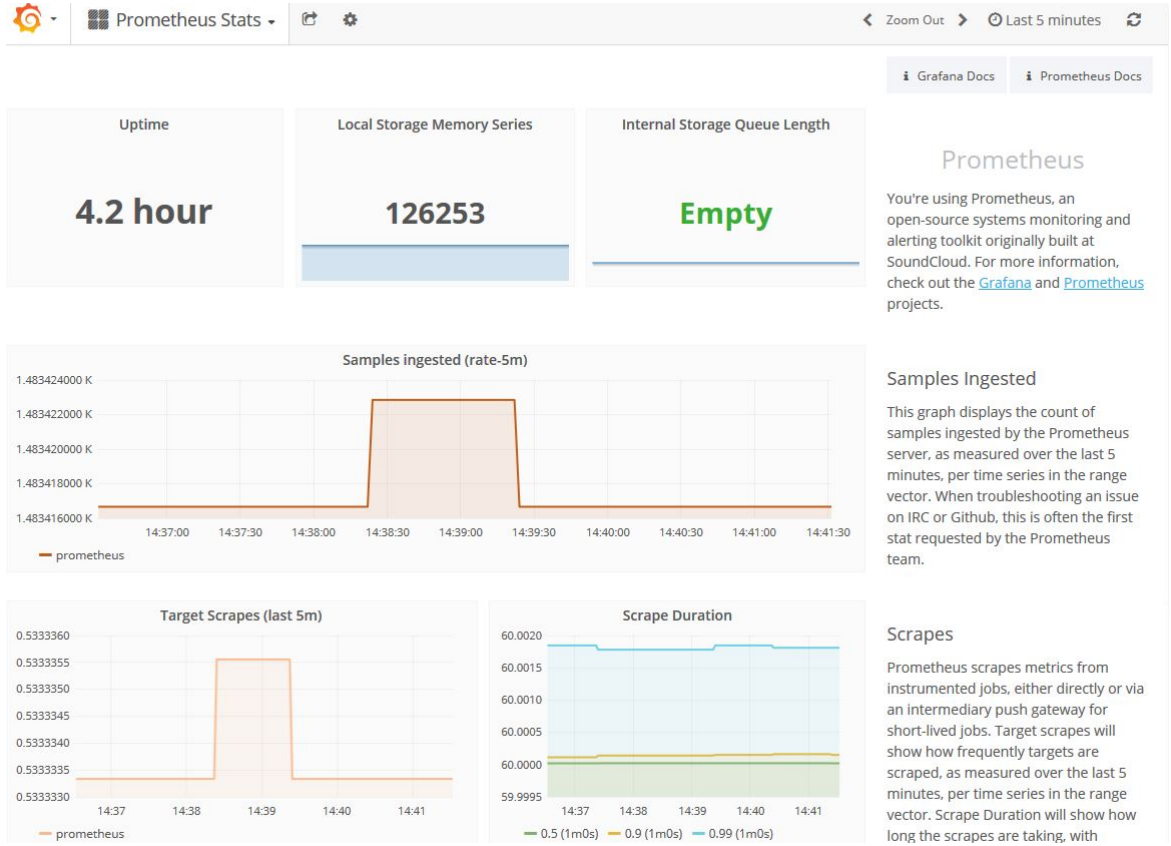
Grafana

■ 視覚化ツール

– 時系列DB (time series database)の視覚化ツール
  • Prometheusの時系列データを視覚化

– アクセス方法
  • ICPコンソール上で「モニタリング」をクリック
  • (ICPでは)Grafanaのログイン画面からのログイン不可

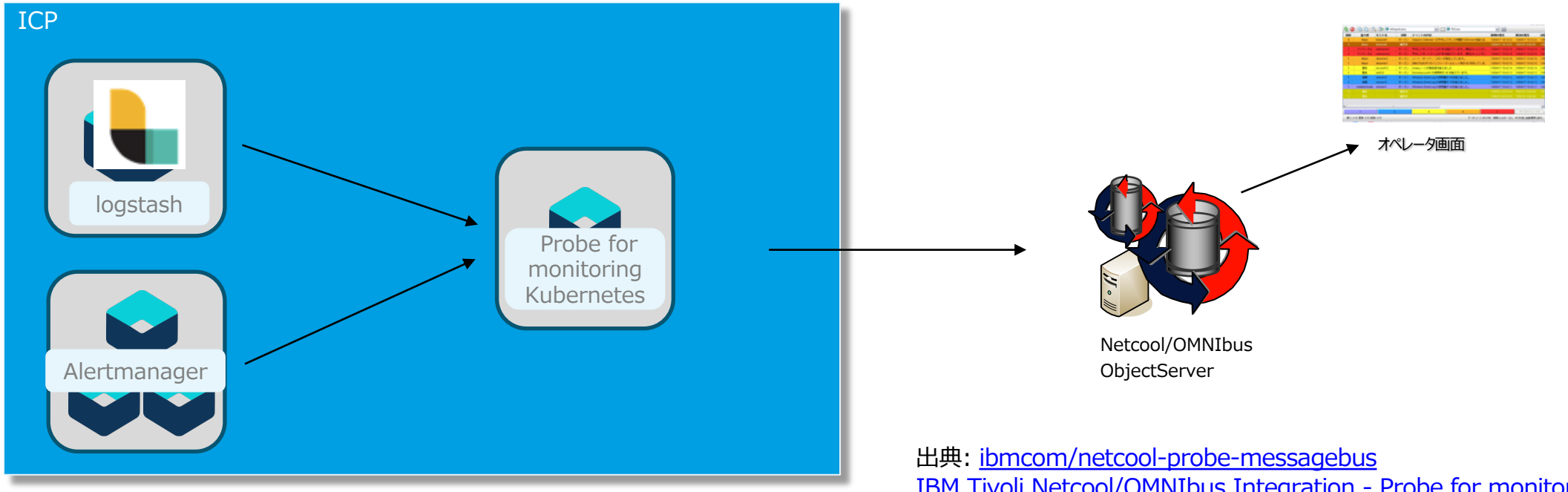




• 参考資料：[Grafana – Documentation]

外部モニタリングシステムとの連携例（ICP – Netcool/OMNIbus連携）

- Netcool/OMNIbus連携用のHelmを利用
  - ICP上にICP連携用のOMNIbusのProbe(Agent)を導入
  - 以下のICPのコンポーネントからメッセージを受け取り、(ICP外部の)Netcool/OMNIbus Object Server(Netcool Server)にイベントを通知
    - logshtash
    - Prometheus Alertmanager



出典: ibmcom/netcool-probe-messagebus
IBM Tivoli Netcool/OMNIbus Integration - Probe for monitoring Kubernetes

<参考>イベント収集・統合監視のコア製品 – Netcool/OMNIbus

様々な環境からの情報収集を可能にする、高可用性な情報収集・連携レポジトリエンジン

ObjectServer

- 業界最高速のメモリ常駐型
  情報収集レポジトリエンジン
- リアルタイム性を重視した
  高性能イベント処理エンジン

Probes

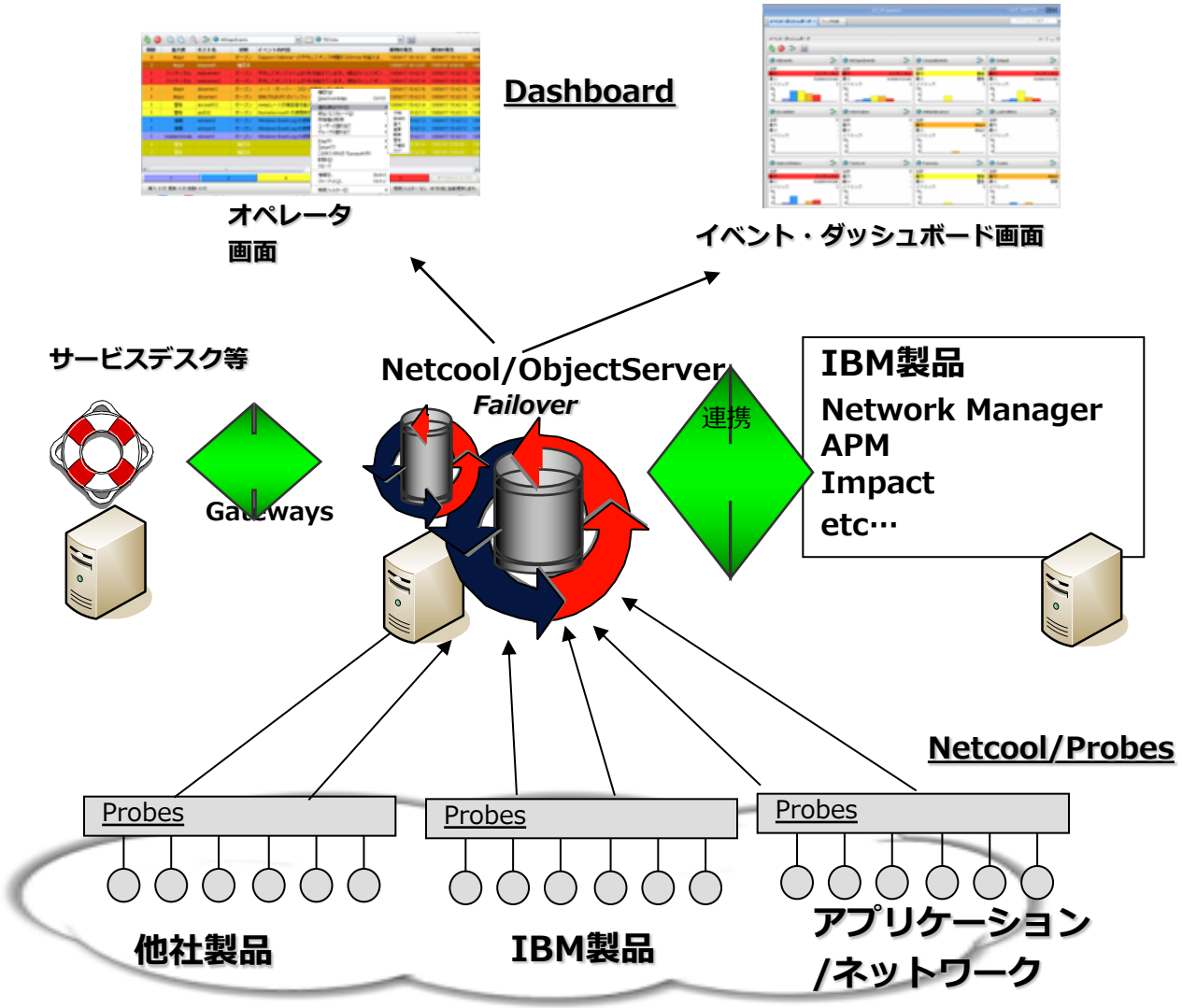
- 様々な環境から情報を取得するために、様々な種類のモジュールを提供。
  高速に収集するための軽量モジュール
- 定義ファイルレベルによるイベントの相関機能
- 業界標準のシステムとの連携では、
  設定の雛形が多数用意され、
  迅速な導入期間を提供

Gateways

- 軽量のデータソース連携モジュール
- 双方向でデータを送信し、
  お互いの最新データを共有化

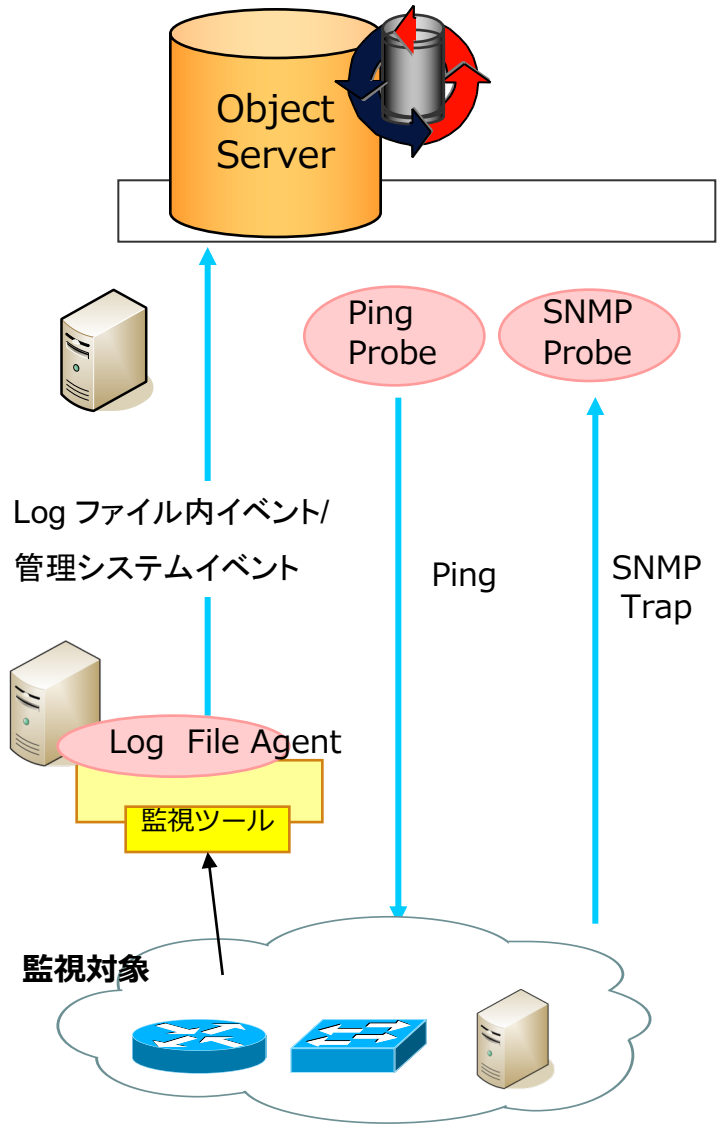
Dashboard Application Services Hub

- 様々な情報が付与されたイベントを基にリアルタイムに様々に可視化された情報を提供
- 容易なフィルタリング作成機能



<参考>イベント収集・統合監視のコア製品 – Netcool/OMNIbus

Object Server – Probe アーキテクチャー



ObjectServer の機能

- メモリ常駐型のデータベース
- イベントデータの蓄積、更新、削除、自動処理
- 自動アクション(Automation)の実行
  - コマンドの実行（メール、別システムへの通知）

Probe の機能

- 他ベンダの管理システムや、機器からのアラート信号等、外部情報を受動的に収集し、ObjectServer にデータを渡す軽量なモジュール
- 様々なネットワークエレメント、管理ツールに対応

例)

syslog probe – UNIX/Linux syslog 監視
Ping Probe – PingリストのIPアドレスへPing監視
SNMP Probe – SNMP Trapの受信

- ルールとルックアップテーブルを使用して、イベントに情報を定義、カテゴライズ、および追加が可能

モニタリング機能の実装に向けて

■ モニタリング機能の実装に向けて必要となる、検討・設計項目について記載する

– メトリクス収集対象と手法の検討
  • ホストマシン、k8sコンテナについてはデフォルトでメトリクス収集設定済み
  • アプリケーション(ミドルウェア)のメトリクスについてPrometheusの収集対象とするか？
  収集する場合、手法をどうするか？(例: ICPのカタログに登録されている「 ibm-icpmonitoring」の利用)
  検討が必要

– Grafana画面(視覚化画面)の検討
  • デフォルトでインポートされているDashboardで充足されるか。充足されない場合、Grafana Labで提供されているものを活用できないか、検討が必要。既存のナレッジを活用できない場合は、一からの開発が必要となる

– アラート発生条件の検討
  • 何をアラート化するのか、その条件について検討が必要

– アラート通知先・手法の検討
  • 発生したアラートを外部に連携したい場合、どの手法(例:メール)を使用するか検討が必要

第4章 第10節:
ロギング

ICPにおけるロギングとは？(1)

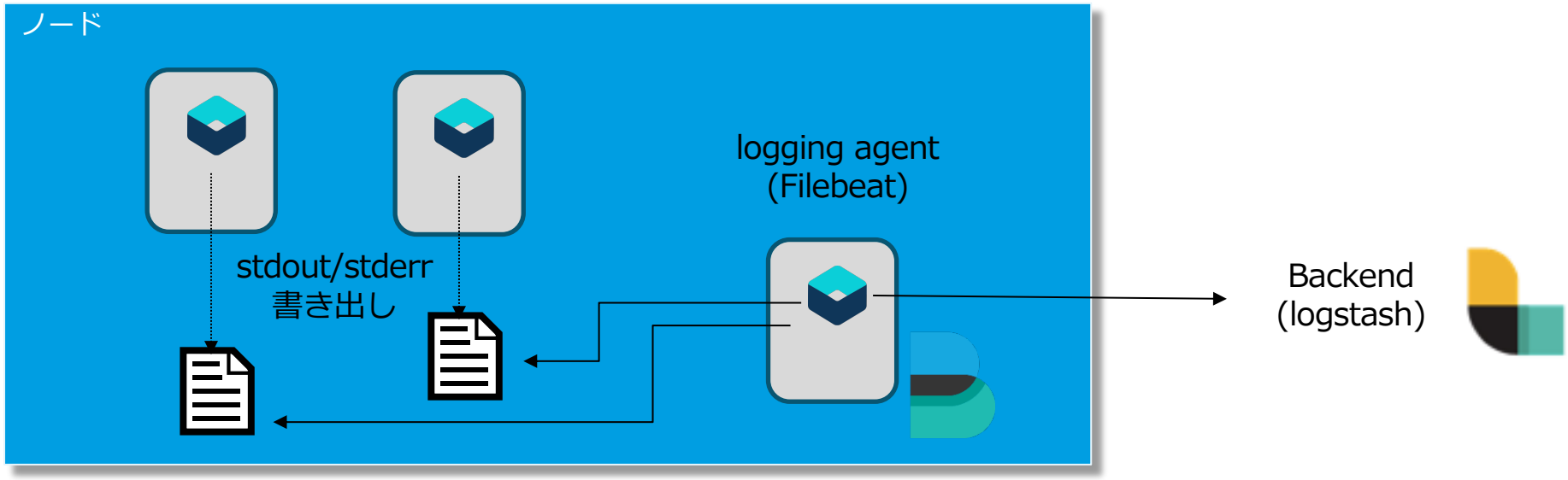
- ログ収集・解析をすること
  - ログを一箇所に集め、すべてのログをまとめて検索できるようにする
    - サーバーやアプリケーションの問題を特定する際、すべてのログを一箇所で検索することが可能
    - 特定の期間のログを相互に関連付けて、複数のサーバーにまたがる問題を特定することも可能
    - 特に、Dockerコンテナでは、コンテナ内のファイルシステムに書き込まれたデータはコンテナの削除時に一緒に破棄される。そのため、保存したいデータをコンテナ外に出力し、保存する必要がある

ICPにおけるロギングとは？(2)

■ ロギングの仕組み

–標準出力/標準エラー出力をContainer Engine(Docker)のlogging driverがファイルとして出力

–logging agent(例: Filebeat)がbackend(例: logstash)にログを転送



Elastic Stackとは? (1)

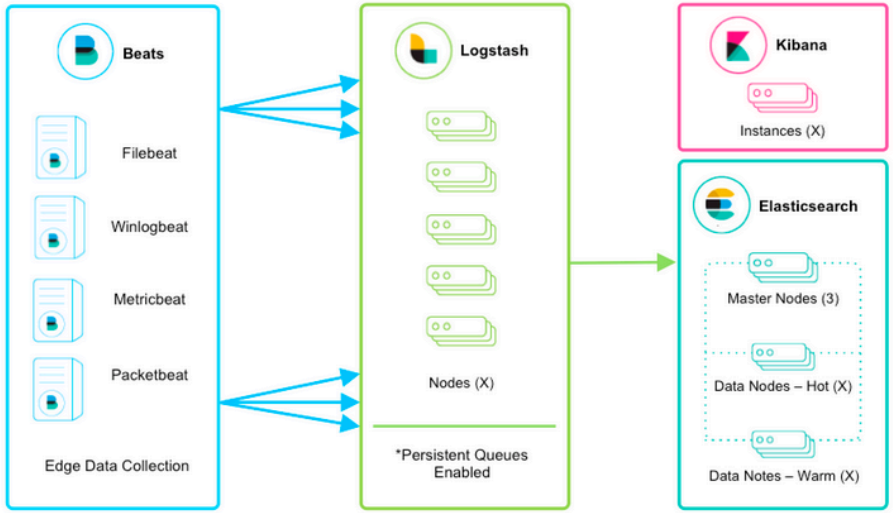
■ロギングデータを収集・分析・可視化するツール

–主要機能
- ログ収集・加工・転送
- データ全文検索（Elasticsearch併用）
- 可視化（Elasticsearch, Kibana併用）

–特徴
- 多彩なプラグイン（200以上）により、様々な入力データ、加工形式、出力形式に対応

–使い方
- ログの分析・解析の際、GUI(Web UI)にアクセス

–各コンポーネントの説明は次ページ参照



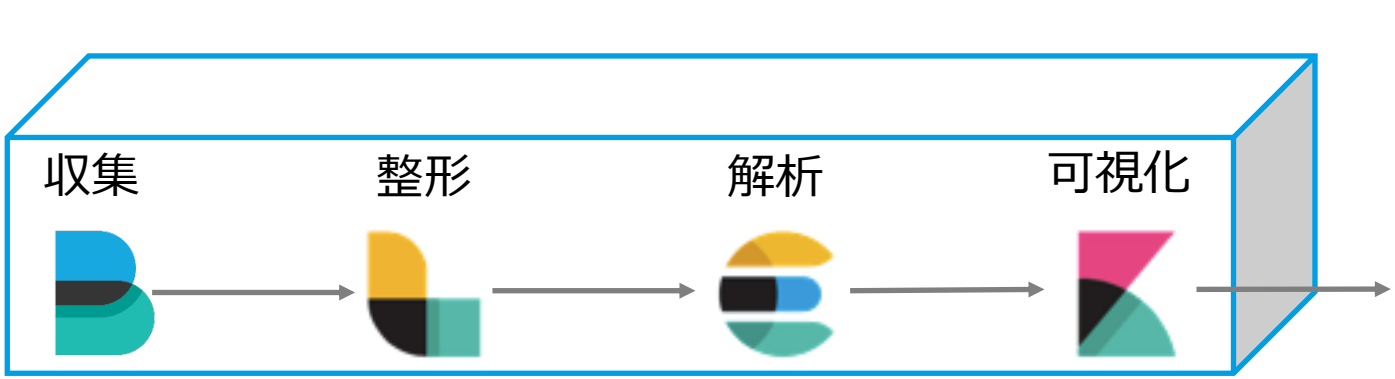
出典: Deploying and Scaling Logstash

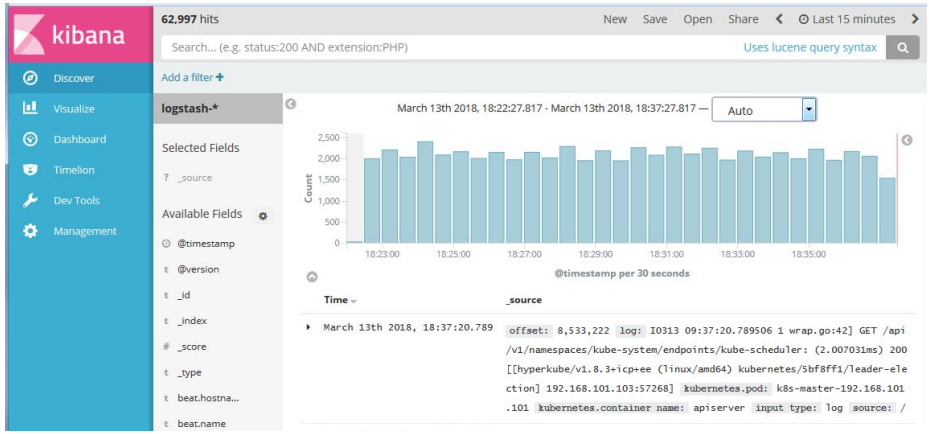
Tips：従来、ELK（Elasticsearch、Logstash、Kibana）という略称で呼ばれていたものにBeatsを加え、新たに「Elastic Stack」として再定義。
ICPのドキュメントでは、ELK Stackの呼称が使用されているが、
この資料ではElastic Stackの呼称を使用する。
elastic - Elastic Stack

Elastic Stackとは? (2)

■ Elastic Stackに関係する主要なコンポーネント

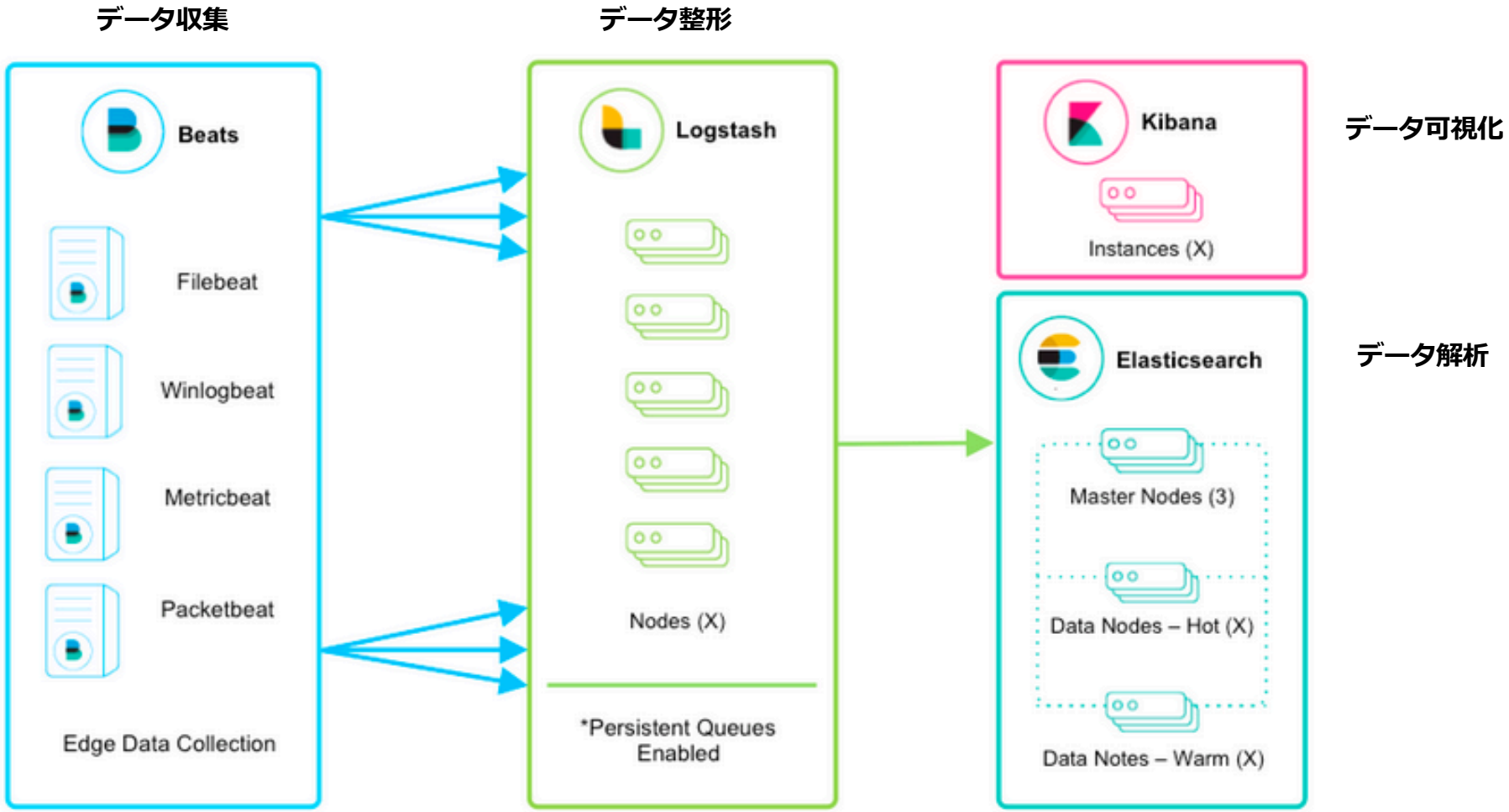
コンポーネント名	概要
Beats	データ収集ツール群の総称。主なBeatとして、ログファイルを収集し、Logstashに転送するFilebeatやCPUやメモリなど統計情報を送信するMetricbeatがある
Logstash	Beatsから収集したログの整形とElasticsearchへのデータ出力をするLogstashでもログ収集できるが、メモリーを必要するため、軽量なBeatsを配置することが一般的
Elasticsearch	Logstashから送られたデータを保管する。リアルタイムの検索・分析エンジン
Kibana	Elasticsearchに保管されたログ・データを検索、分析、可視化する 。 Web ベースのGUI





Elastic Stackとは? (3)

■ 構成例



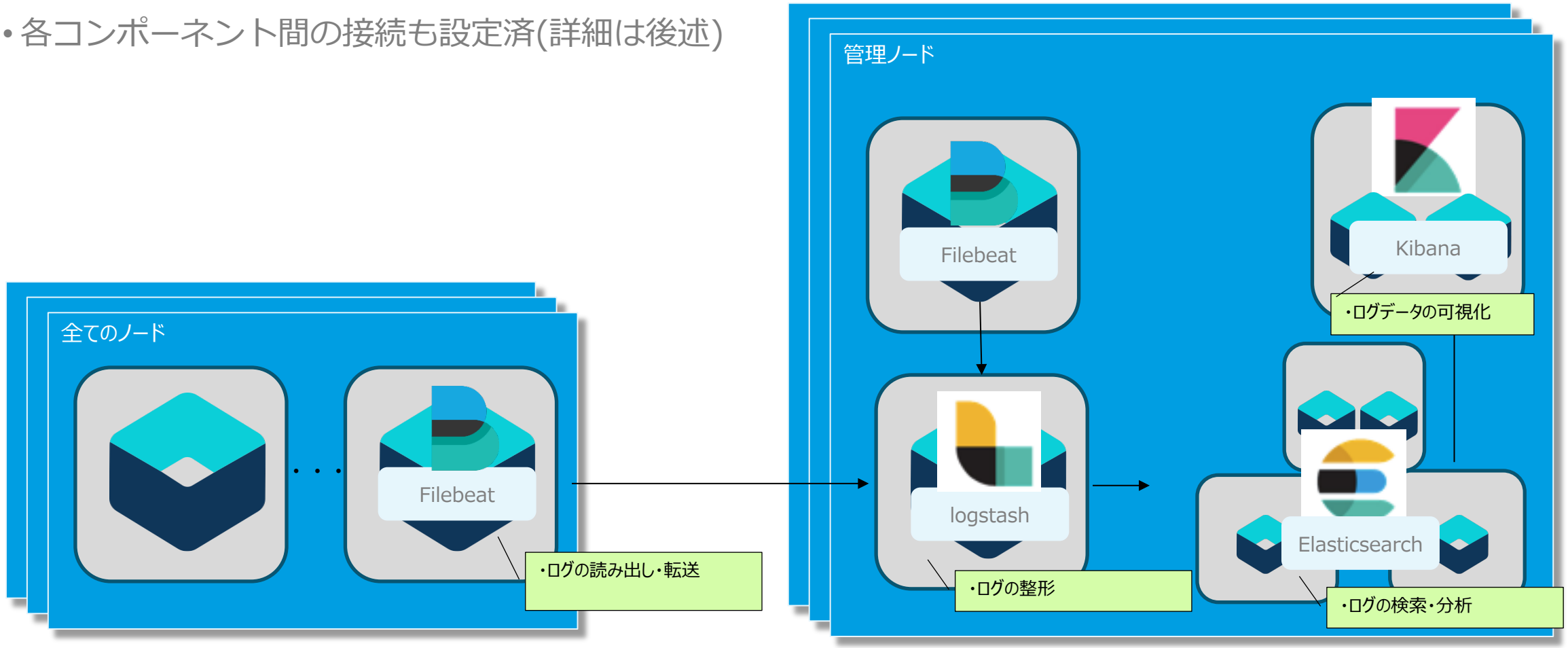
出典: Deploying and Scaling Logstash

ICP k8s環境 - Elastic Stackコンポーネント

■ ICP k8s環境におけるElastic Stackのコンポーネント配置図は以下となる

–ICP導入後

- 各コンポーネントは起動済
- 各コンポーネント間の接続も設定済(詳細は後述)



アプリケーション・ログに対するロギング(1)

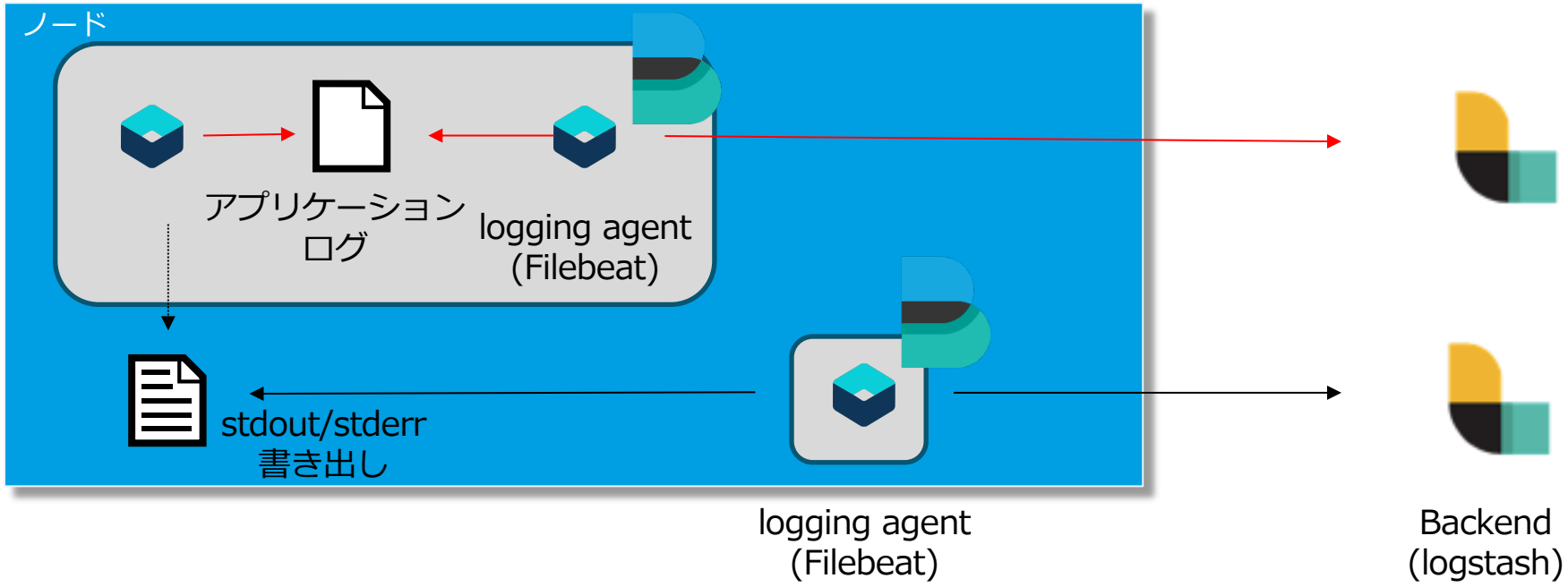
- デフォルトで導入されるElastic Stackは、システム・ログのロギング用として使用される
- IBM Redbookでは、アプリケーション・ログのロギングについては、カタログから追加でElastic Stackをデプロイすることが推奨されている
  - カタログに登録されているchart
    - ibm-icplogging
  - 参考資料：
    - IBM Knowledge Center - IBM Cloud Private ロギング・サービス
    - IBM Redbooks - IBM Cloud Private System Administrator's Guide

アプリケーション・ログに対するロギング(2)

■ アプリケーションが個別にログ出力する場合、収集をどうするかを検討する

– アプリケーション・コンテナと同じPod内にログ収集用のサイドカー・コンテナを追加

• アプリケーションをデプロイする際に、サイドカー・コンテナを一緒にデプロイするように構成



アラート通知について

- ICPで導入されるElastic Stackはロギング機能を提供するが、アラート通知機能は提供していない。
- アラート通知のためには、別途コンポーネントを導入し、Elasticsearchに収集されたログデータに対し条件を設定しアラート通知する方法が考えられる
  - 案1) X-Packの利用
    - X-PackはElastic Stackの拡張機能でICPでは導入されているが無効化されている。使用のためには、ライセンス取得(別途有償)が必要
  - 案2) ElastAlertの利用
    - Yelp社が開発したOSSツール
  - 参考情報：
    - IBM Knowledge Center – Updating Elastic X-Pack licenses

ロギング機能の実装に向けて

■ ロギング機能の実装に向けて必要となる、検討・設計項目について記載する

– 収集対象のログの選定と手法の検討
  • dockerコンテナのログについてはデフォルトで収集設定済み
  • アプリケーション(ミドルウェア)のログについても収集対象とするか
  収集する場合、サイドカー・コンテナを作成し、dockerコンテナのログと収集経路を分けるか、検討が必要
– Kibana画面(視覚化画面)の検討
  • Visualize画面、Dashboard画面、Timelion画面において、どういった内容を表示させるのか設計が必要
  • また、画面については一から作成することになる。(要件にあうサンプルがあれば、インポートも可能)
– ログ・データとフィールドのマッピング (logstash)の検討
  • Kibanaでログを絞り込んだり、グラフ化表示する際にフィールドを使用する。デフォルトのフィールド設定で、充足されるか検討が必要
– データ・メンテナンス(Elasticsearch)の検討
  • データ・メンテナンスについて、実施タイミング・対象データ・保持期間がデフォルトで充足されるか検討が必要

第4章 第11節:
（基盤担当者向け）CLI操作

ICPを操作するためのCLIツール

- ICP には、いくつかのコマンド・ライン・インターフェースが用意されている
  - cloudctl, kubectl, helm, istioctl はICP導入時にマスター・ノードに導入されるが、必要に応じて操作用端末にインストール可能
  - 下表にCLIの種類・用途・参考リンクを示す

CLI種別	用途	参考リンク
IBM Cloud Private CLI (cloudctl)	クラスターに関する情報の表示、クラスターの管理、Helm チャートおよびワークロードのインストールなどを行う	IBM Knowledge Center – IBM Cloud Private CLI (cloudctl) を使用したクラスターの管理
Kubernetes CLI (kubectl)	Kubernetes クラスターの管理を行う	IBM Knowledge Center - Kubernetes CLI (kubectl) のインストール Kubernetes.io - Overview of kubectl
Helm CLI (helm)	Helmチャートの管理やデプロイ、リリースの管理を行う	IBM Knowledge Center - Helm CLI(helm)のインストール Code - helm/docs/ - GitHub
Istio CLI (istioctl)	クラスター内のサービス・メッシュの管理を行う	IBM Knowledge Center Istio CLI (istioctl) のインストール(istioctl) Istio.io istioctl
Calico CLI (calicoctl)	Calicoネットワークとセキュリティー・ポリシーの管理を行う （マスター・ノード、ワーカー・ノード、プロキシー・ノードにインストール）	IBM Knowledge Center - Calico CLI(calicoctl)のインストール

IBM Cloud Private CLI (cloudctl)のセットアップ

- IBM Cloud Private CLI (cloudctl) セットアップの例（macOS）
  - IBM Cloud Private管理コンソールから、「メニュー」＞「コマンド・ライン・ツール」＞「IBM Cloud Private CLI」をクリックし、該当するOS用の curl コマンドをコピーする
    - curl -kLo <インストールパッケージ名> https://<master-node-ip>:8443/api/cli/cloudctl-darwin-amd64
  - ターミナルを開き、コピーしたcurl コマンドでコマンド・ライン・ツールをダウンロードする
  - ダウンロードしたファイルのパーミッションを変更し、適切なディレクトリーへ移動する
    - chmod 755 <インストールパッケージ名>
    - sudo mv <インストールパッケージ名> /usr/local/bin/cloudctl
  - IBM Cloud Private CLIがインストールされていることを確認する
    - cloudctl –help
  - 続いてKubernetes CLI をセットアップする（次ページ参照）

参考情報：IBM Cloud Private CLI のインストール

Kubernetes CLI (kubectl) のセットアップ（1/2)

- Kubernetes (kubectl) セットアップの例（macOS）
  - IBM Cloud Private管理コンソールから、「メニュー」＞「コマンド・ライン・ツール」＞「Kubernetes CLI」をクリックし、該当するOS用の curl コマンドをコピーする
    - curl -kLo <インストールパッケージ名> https://<master-node-ip>:8443/api/cli/kubectl-darwin-amd64
  - ターミナルを開き、コピーしたcurl コマンドでコマンド・ライン・ツールをダウンロードする
  - ダウンロードしたファイルのパーミッションを変更し、適切なディレクトリーへ移動する
    - chmod 755 <インストールパッケージ名>
    - sudo mv <インストールパッケージ名> /usr/local/bin/kubectl

（次ページへ続く）

参考情報：Kubernetes CLI (kubectl) のインストール

Kubernetes CLI (kubectl) のセットアップ(2/2)

- Kubernetes (kubectl) セットアップの例（macOS）
  - IBM Cloud Private管理コンソールのユーザー・アイコンを選択し、「クライアントの構成」をクリックする

  - 以下のようなクラスター構成詳細が表示されるので、コピーしてターミナルへ貼り付ける

CLI を構成するには、表示される構成コマンドを端末ウィンドウに貼り付けて実行します。

```
kubectl config set-cluster mycluster --server=https://128.168.83.249:8001 --insecure-
kubectl config set-context mycluster-context --cluster=mycluster
kubectl config set-credentials admin --token=eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9
kubectl config set-context mycluster-context --user=admin --namespace=accountsyste
kubectl config use-context mycluster-context
```

  - この構成は12時間で有効期限が切れるため、12時間ごとにログインして再構成が必要となる
  - この制限を回避したい場合は、下記の手順に従ってCLIを構成する（ServiceAccountを作成して権限を付与し、その認証トークン（有効期限なし）をCLIに設定する）
    - 参考情報：Configuring the Kubernetes CLI by using service account tokens

cloudctl, kubectl の使用(1/3)

- CLIをセットアップした端末からICPへログインする
  - 「cloudctl login -a https://<マスター・ノードのIPアドレス>:8443 --skip-ssl-validation」を実行

```
macbook-air:~ USER01$ cloudctl login -a https://<MasterNode IP>:8443 --skip-ssl-validation

ユーザー名> admin                                          <= ICPのユーザーは前ページクラスター構成詳細でセットされたユーザーを入力する。

パスワード>                                                <= 上記ユーザーのパスワードを入力する
認証しています...
OK

ターゲットのアカウント mycluster Account (id-mycluster-account)

名前空間を選択してください:
1. accountsystem
2. bookingsystem
3. cert-manager
4. db2
5. default
6. ibmcom
7. isolatednamespace
8. istio-system
9. kube-public
10. kube-system
11. marketingsystem
12. platform
13. services
14. was
数値を入力してください> 6          <= 名前空間は任意の番号を入力する。（kubectlコマンドで操作したい名前空間を選択する。ここでは例として「6」を入力。）
ターゲット名前空間 ibmcom
（中略）
Helm の構成中: /Users/USER01/.helm
OK
macbook-air:~ USER01$
```

cloudctl, kubectl の使用(2/3)

- 「cloudctl login」でICPにログインが成功すると、kubectlの接続情報（~/.kube/config）が設定され、kubectlコマンドでICPに接続して操作できるようになる。
  - 「kubectl version --short」を実行する。
    - Client、Serverの両エントリーが表示されることを確認する。

```
macbook-air:~ USER01$ kubectl version --short
Client Version: v1.12.4
Server Version: v1.12.4+icp-ee
macbook-air:~ USER01$
```

  - 「kubectl get nodes」を実行する。

```
macbook-air:~ USER01$ kubectl get nodes
NAME           STATUS   ROLES                AGE     VERSION
10.192.27.17   Ready    va                   6d22h   v1.12.4+icp-ee
10.192.27.18   Ready    management           7d      v1.12.4+icp-ee
10.192.27.4    Ready    etcd,master,proxy    7d1h    v1.12.4+icp-ee
10.192.27.46   Ready    worker               7d      v1.12.4+icp-ee
10.192.27.58   Ready    isolatedworker       6d16h   v1.12.4+icp-ee
10.192.27.60   Ready    isolatedproxy        6d16h   v1.12.4+icp-ee
macbook-air:~ USER01$
```

cloudctl, kubectl の使用(3/3)

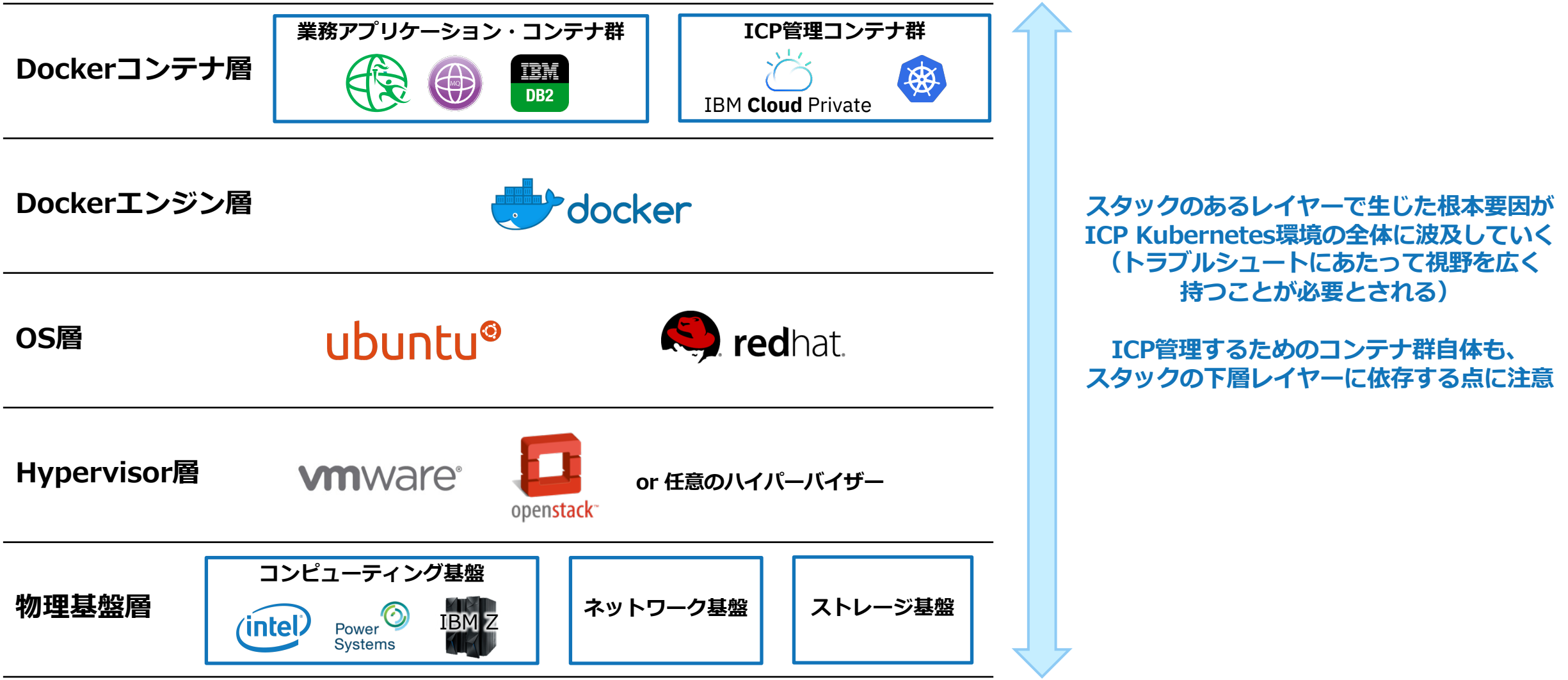
– 「cloudctl logout」を実行し、ICPからログアウトする。

```
macbook-air:~ USER01$ cloudctl logout
ログアウトしています...
OK

kubectl 構成を削除しています...
Property "clusters.mycluster" unset.
Property "users.mycluster-user" unset.
Property "contexts.mycluster-context" unset.
OK

Helm 証明書を削除しています: /Users/USER0!/.helm
OK
macbook-air:~ USER01$
```

第4章 第12節:
トラブルシュート

トラブルシュート（スタック構成を意識したトラブルシュート）

■ ICP Kubernetes環境のスタック構成を意識したトラブルシュートが必要とされる



スタックのあるレイヤーで生じた根本要因が
ICP Kubernetes環境の全体に波及していく
（トラブルシュートにあたって視野を広く
持つことが必要とされる）

ICP管理するためのコンテナ群自体も、
スタックの下層レイヤーに依存する点に注意

トラブルシュート（トラブルシュート手順の整備）

■ トラブルの発生に備えて整備することが望ましい運用手順を下表に示す

トラブルシュート手順の分類	トラブルシュート手順の概要	手順整備にあたっての参考情報
ダッシュボードの確認	ICP基盤が提供するダッシュボード上でKubernetes環境の稼動状況を確認する	IBM Knowledge Center – システムおよびリソースのモニタリング IBM Knowledge Center - IBM Cloud Private クラスター・モニタリング IBM Knowledge Center – ポッド情報の表示 IBM Knowledge Center – IBM Cloud Private ロギング
業務アプリケーションコンテナ上での任意の診断コマンドの実行・ファイル転送	コンテナのシェルにログインした上で、任意の診断コマンドを実施・ファイル転送を実行（”kubectl exec”, “kubectl cp”）	Get a Shell to a Running Container – Kubernetes Copying Container Files - Kubernetes
ダッシュボードへのアクセスが不可能な状況下でのコンテナ稼働の状況確認	Kubernetes CLIインターフェースからの状況確認（”kubectl get”, kubectl describe”, ”kubectl logs”, …etc.)、およびKubernetes関連のログ確認を行う。 cloudctl loginが不可能な場合はMasterノードでkubectl-configファイルを使用する。	IBM Knowledge Center – Kubernetes CLI(kubectl)からクラスターへのアクセス Troubleshoot Applications – Kubernetes Troubleshoot Clusters – Kubernetes kubectl Cheat Sheet – Kubernetes IBM Cloud Private troubleshooting – Troubleshooting a typical workload
Kubernetes環境での管理運用操作が不可能な状況下でのコンテナ稼働の状況確認	各ノードのDocker CLIインターフェースからの状況確認を行う（”docker ps”, “docker logs”, “docker top”）	docker ps | Docker Documentation docker logs | Docker Documentation docker top | Docker Documentation
OS層より下層での状態確認	（従来型のサーバー基盤と同様の考え方に基づき、トラブルシュート手順を整備する）	

Kubernetes環境の状況確認（”kubectl”コマンドの使用例）

■ “kubectl”コマンドの使用例を以下に示す

```
$ kubectl get pods                    --> Pod 一覧を確認
NAME                                  READY     STATUS    RESTARTS   AGE
nginx-deployment-1006230814-6winp     1/1       Running   0          11s
nginx-deployment-1006230814-fmgu3     1/1       Running   0          11s
```

```
$ kubectl describe pod nginx-deployment-1006230814-6winp          → Pod の状態を取得する
  (中略)
Status:                 Running
  (中略)
Containers:
  nginx:
  (中略)
    State:              Running
      Started:          Thu, 24 Mar 2016 01:39:51 +0000
    Ready:              True
    Restart Count:      0
Conditions:
  Type                  Status
  Initialized           True
  Ready                 True
  PodScheduled          True
Events:
  FirstSeen   LastSeen   Count   From                          SubobjectPath                Type      Reason      Message
  54s         54s        1       {kubelet kubernetes-node-wul5}  spec.containers{nginx}     Normal    Pulling     pulling image "nginx“
  (後略)
```

Kubernetes環境の状況確認（”kubectl”コマンドの使用例）

■ “kubectl”コマンドの使用例を以下に示す（続き）

```
$ kubectl exec nginx-deployment-1006230814-6winp ls          → コンテナへ接続し、任意のコマンドを実行する
bin
boot
config
dev
etc
home
…(中略)
mnt
opt
output
proc
root
run
sbin
srv
sys
tmp
usr
var
Satoko-no-MacBook-Air:~ AA283135$
```

トラブルシュート（トラブルシュートのための情報源）

■ ICP Kubernetes環境でのトラブルシュートのための情報源を下表にまとめる

ソフトウェア製品	トラブルシュート向け情報源	既知の障害情報
IBM Cloud Private	IBM Knowledge Center – イベントとログ (CLI) IBM Knowledge Center – イベントおよびログ (クラスター management console)	IBM Knowledge Center – 既知の問題および制限 IBM Knowledge Center – トラブルシューティングとサポート
Kubernetes	Troubleshoot Applications – Kubernetes Troubleshoot Clusters – Kubernetes Application Introspection and Debugging - Kubernetes Debug Init Containers – Kubernetes Debug Pods and ReplicationControllers – Kubernetes Debug Services – Kubernetes Debug a StatefulSet – Kubernetes Determine the Reason for Pod Failure - Kubernetes	Issues · kubernetes/kubernetes · GitHub
Prometheus	FAQ | Prometheus	Issues · prometheus/prometheus · GitHub Issues · prometheus/alertmanager · GitHub Issues · prometheus/node_exporter · GitHub
Grafana	Troubleshooting | Grafana Documentation	Issues · grafana/grafana · GitHub
Elastic Stack	Troubleshooting | Elastic Stack Overview [7.2] | Elastic	Issues · elastic/elasticsearch · GitHub Issues · elastic/logstash · GitHub Issues · elastic/kibana · GitHub Issues · elastic/beats · GitHub