HITACHI
Inspire the Next

パブリッククラウド活用ガイド [Linux]

Amazon EC2 Auto Scaling 対応編

2021.9
September

本ガイドには，下表に示す変更履歴が含まれています。

作成年月	変更内容	変更区分(*1)
2020 年 7 月	初版	－
2021 年 6 月	サーバ管理コマンドの設定において，以下 2 つのプロパティの設定を削除しました。 ・ vbroker.se.iiop_tp.host ・ ejbserver.rmi.naming.host	C
2021 年 9 月	「セッションマネージャの指定機能」について案内を追加しました。本機能の利用により，HTTP セッションを利用するアプリケーションが使用できるようになります。	A

なお，単なる誤字・脱字などは，お断りなく訂正しました。

注 *1

変更区分 C：案内を変更（変更又は削除）します。既存のユーザは，使い方を変更する必要があります。

変更区分 A：既存のユーザには影響ありません。新しい案内を適用する場合だけ，使い方を変更する必要があります。

変更区分 S：案内の変更はありません。説明の追加・変更があります。

はじめに

本ガイドは，Amazon EC2 Auto Scaling に対応した uCosminexus Application Server におけるシステム構築手順をご案内するものです。

１．対象とする読者

uCosminexus Application Server を使用し，システムを設計・構築・運用する立場にある方

２．対象とする製品

P-9W43-7KB1 uCosminexus Application Server

（論理 J2EE サーバを V9 互換モードで実行している場合は対象外）

３．Amazon Elastic Compute Cloud のインスタンスで使用可能な OS

対象製品の適用 OS と同じです。適用 OS は製品のリリースノートを参照してください。

■商標類

■用語の表記

本ガイドで用いる用語を次のように定義します。

用語	意味
uCAS	uCosminexus Application Server の略。
Web システム	uCAS 上に構築した論理サーバ群で実現するシステム。

■Amazon Web Services のサービス略称

Amazon Web Services （AWS） のサービス名称の略称は次の通りです。詳細は AWS のウェブサイト（http://aws.amazon.com）を参照してください。

サービス略称	AWS のサービス名称
Amazon EC2	Amazon Elastic Compute Cloud
Amazon VPC	Amazon Virtual Private Cloud
ELB	Elastic Load Balancing
AMI	Amazon Machine Image
EBS	Amazon Elastic Block Store
Amazon S3	Amazon Simple Storage Service
Amazon EFS	Amazon Elastic File System

■本ガイドをご使用いただく際の注意点

本ガイドに記載されている仕様は，製品の改良により予告なく変更されることがあります。

■発行元

株式会社日立製作所　サービス＆プラットフォームビジネスユニット
サービスプラットフォーム事業本部

目　次

1 構築するシステム

この章では，本ガイドで構築するシステムについて説明します。

本章の構成

本ガイドでは，uCAS を使って Amazon EC2 Auto Scaling に対応したシステムを構築していきます。

Amazon EC2 Auto Scaling は EC2 インスタンスを，システム負荷等の条件に応じてスケールアウトしたりスケールインしたりする AWS のサービスです。これによりシステムリソースを必要な時に必要なだけ利用して可用性向上やコスト最適を実現できます。

本章では，本ガイドで構築するシステムの構成や注意点について説明します。説明中に uCAS の各プロセスが登場しますが，各サーバプロセスの詳細については uCAS のマニュアルを参照してください。

本章で示すシステム構成以外の構成にする必要がある場合には，本ガイドの掲載元よりお問い合わせください。

1.1 システム構成

本ガイドでご案内する手順では Management Server の Smart Composer 機能を使用して，Amazon EC2 Auto Scaling に対応した Web システムを構築します。構築するシステムの構成を図 1-1 に示します。また，構築したシステムを Amazon EC2 Auto Scaling と組み合わせた環境の例を図 1-2 に示します。

構築するシステムは，論理 J2EE サーバ，論理 Web サーバ，論理パフォーマンストレーサ，Management Server，運用管理エージェントが各 1 つずつで構成されたもので，これらはすべて同一 EC2 インスタンス上に配置します。Web システムへのリクエストは HTTP / HTTPS を用います。

このシステムが配置された EC2 インスタンスを Amazon EC2 Auto Scaling によりスケーリングします。スケールアウトの際には，新たに生成した EC2 インスタンス上の Web システムが自動的にサービス提供可能な状態になるようにします。Web システムが動作する EC2 インスタンス群（Auto Scaling グループ）の前段には ELB（図 1-2 中の Application Load Balancer）を配置し，提供するサービスへのリクエストを各 Web システムへ振り分けることで負荷分散を行います。

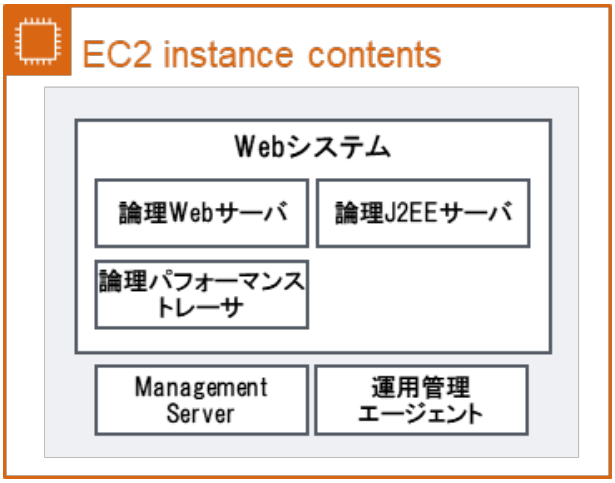

図 1-1 構築するシステム

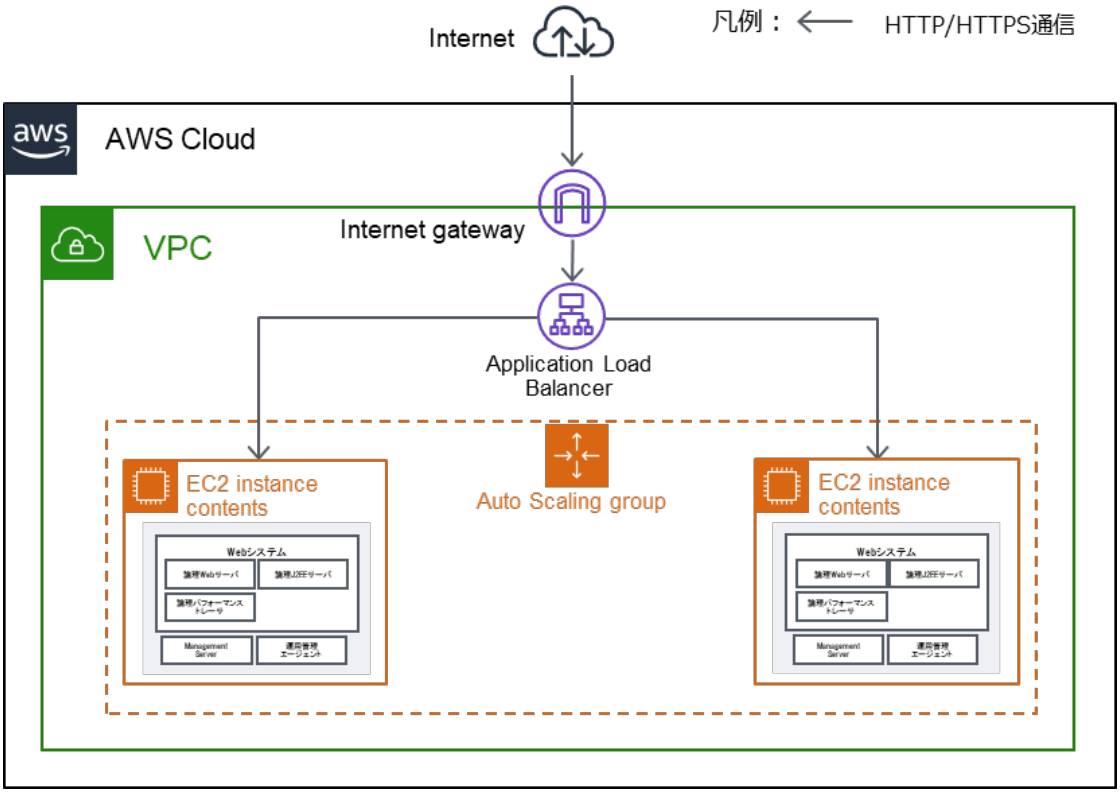

図 1-2 Amazon EC2 Auto Scaling と組み合わせた環境

1.2 制限事項

構築するシステムでは，以下に示すアプリケーションは実行しないでください。

- Web システム毎に異なる状態（Web システム固有の情報）を持つようなアプリケーション
- Auto Scaling グループに属する EC2 インスタンス上の Web システム同士が直接通信を行うようなアプリケーション
- グローバルトランザクションを使用するアプリケーション

構築するシステムでは，以下に示す設定および操作を行わないでください。

- IPv6 を使用した通信を行う uCAS の設定
- AWS Systems Manager Run Command を使用した uCAS のコマンド実行
- セッションフェイルオーバ機能を使用する設定

Web システム毎に異なる状態を持つアプリケーションは，

- ELB でリクエストを振り分けるため，期待した状態を持つ Web システムにリクエストが届くとは限らない
- スケールインにより EC2 インスタンスごと状態も削除されてしまいサービス継続できなくなる恐れがある

といった理由から，Amazon EC2 Auto Scaling と組み合わせて使用することができません。

ただし，状態の形式として HTTP セッションを利用するアプリケーションについては，本ガイドの案内に加えて，uCosminexus Application Server 11-10 以降で利用可能な「セッションマネージャの指定機能」を利用することで，Amazon EC2 Auto Scaling と組み合わせて使用することができます。「セッションマネージャの指定機能」を利用すると，HTTP セッションの情報を Amazon ElastiCache で管理し，複数の uCAS 間で HTTP セッションの情報を共有することができるようになります。機能の詳細については，マニュアル「Cosminexus V11 アプリケーションサーバ 機能解説 基本・開発編(Web コンテナ)」を参照してください。

※uCosminexus Application Server 11-00 を使用する場合の補足事項

uCosminexus Application Server 11-00 では，「セッションマネージャの指定機能」を利用することができません。
アプリケーションで HTTP セッションを利用する必要がある場合には，ELB のスティッキーセッションを使用することで，HTTP セッションを利用したアプリケーションを限定的に実行することができます。ELB のスティッキーセッションを使用する場合，ELB が特定のセッションと EC2 インスタンスの紐づけを行います。これにより，特定のセッションを持つ HTTP リクエストは，紐づけされた EC2 インスタンス上の Web システムに常に振り分けられるようになります。そのため，Auto Scaling グループ内に複数の EC2 インスタンスが存在する環境においても，セッションを持つ HTTP リクエストをそれに対応するセッション情報を持つ Web システムで処理することができます。
ただし，負荷分散効果を狙って EC2 インスタンスをスケールアウトしたとしても，セッションを持つ HTTP リクエストは紐づけされた EC2 インスタンス上の Web システムに常に振り分けられてしまうため，追加した EC2 インスタンスによる負荷分散効果が得られにくくなります。また，EC2 インスタンスをスケールインする場合，対象の EC2 インスタンス上で保持しているセッション情報は失われます。

2 構築手順

この章では，Amazon EC2 Auto Scaling に対応したシステムの構築手順について説明します。

本章の構成

2.1 AWS の事前準備
2.2 uCAS の構築手順

本章では，Amazon EC2 Auto Scaling に対応したシステムを構築する手順を示します。手順の流れは以下の通りです。

1.	uCAS インストール＆初期設定	・・・	2.2.1～2.2.3 項
2.	簡易構築定義ファイルの作成＆Web システム構築	・・・	2.2.4～2.2.5 項
3.	Amazon EC2 Auto Scaling と組み合わせて自動運用するための仕組みの作成	・・・	2.2.6～2.2.7 項
4.	AMI 作成準備	・・・	2.2.8～2.2.9 項

説明中に uCAS の機能やコマンドを使用しますが，必要に応じて uCAS のマニュアルを参照してください。

本章に示す手順以外の操作を行う必要がある場合には，本ガイドの掲載元よりお問い合わせください。

2.1 AWS の事前準備

システムの構築を始める前に，EC2 インスタンスを 1 つ作成してください。作成した EC2 インスタンス上に uCAS でシステムを構築し，運用時にはこの EC2 インスタンスから作成した AMI を用いてスケーリングします。AMI の作成元となる EC2 インスタンスを，マスターEC2 インスタンスと呼びます。EC2 インスタンスのインスタンスタイプや，紐づけるストレージの種類・容量を決定する際には，システム要件やアプリケーションの内容に加えて，uCAS のメモリ所要量，ディスク占有量を考慮する必要があります。uCAS のメモリ所要量，ディスク占有量はリリースノートで確認してください。

2.2 uCAS の構築手順

マスターEC2 インスタンス上に 1.1 節で示したシステムを構築するための手順を示します。以下の各項に続く手順を順番に実施してください。

Amazon EC2 Auto Scaling と組み合わせたシステムでは，構築済みの uCAS が配置されたマスターEC2 インスタンスから AMI を作成し，その AMI をもとに Amazon EC2 Auto Scaling によって新たな EC2 インスタンスをスケールアウトしていきます。つまり，各 EC2 インスタンス上で動作する uCAS のシステムは原則すべて同一の設定で動作することになります。そのため，EC2 インスタンス固有の情報を使用した設定はできません。uCAS の設定の中には，ホスト名や IP アドレスを指定するものがありますが，これらは EC2 インスタンス固有の情報ですので，**ホスト定義やホスト関連の設定では「localhost」を使用した設定**を行う必要があります。その他，ポート番号やファイルパス等の各設定に関しても，スケールアウトで生成される EC2 インスタンス上で動作する uCAS のシステムがすべて同一の設定になることを前提に設定してください。

各手順で共通の注意事項を以下に示します。

- 各項の操作は必ずスーパーユーザで実行してください。
- 環境変数「$COSMINEXUS_HOME」は uCAS のインストールディレクトリの絶対パスを指しています。
- Amazon EC2 Auto Scaling に対応した uCAS によるシステムの構築に必要な設定を中心に案内しています。案内していない設定についてもシステム要件に合わせた設定が必要です。

2.2.1 uCAS のインストール

マスターEC2 インスタンスを起動し，uCAS を Linux にインストールします。以下の方法に従ってインストールを実施してください。

1. uCAS のマニュアル「CosminexusV11 アプリケーションサーバ システム構築・運用ガイド 2 章」に従い，パッケージの確認，および言語設定の確認を行います。
2. uCAS の電子媒体（ISO ファイル形式の日立オープンミドルウェア DVD-ROM 提供媒体）を準備します。uCAS の電子媒体に併せて修正パッチ CD が提供されている場合は，修正パッチ CD の電子媒体も準備してください。電子媒体をお持ちでない場合は，弊社営業までご相談ください。
3. ISO ファイルを EC2 インスタンスの Linux にファイル転送します。
4. スーパーユーザでログインして転送した ISO ファイルをマウントします。
5. uCAS のマニュアル「CosminexusV11 アプリケーションサーバ システム構築・運用ガイド 2 章」に従いマウント後の手順を実施してください。
6. 修正パッチ CD の電子媒体を準備している場合は，電子媒体と併せて提供されている「修正パッチ CD のご説明」を参照しながら修正パッチを適用してください。

インストールが完了したら，マニュアル「Cosminexus V11 アプリケーションサーバ リファレンス コマンド編 付録 H」を参考に必要な環境変数の設定を行ってください。

2.2.2 サーバ管理コマンドの設定

サーバ管理コマンドの設定を行います。表 2-1 に従った設定になるよう，必要に応じて設定ファイルを編集してください。

表 2-1 サーバ管理コマンド用システムプロパティファイルの設定

対象の設定ファイル		
$COSMINEXUS_HOME/CC/admin/usrconf/usrconf.properties		
#	プロパティ	値
1	ejbserver.naming.host	localhost

2.2.3 Management Server と運用管理エージェントのセットアップ&設定

Management Server と運用管理エージェントの設定を行います。表 2-2，表 2-3 に従った設定になるよう，必要に応じて設定ファイルを編集してください。

表 2-2 運用管理エージェントプロパティファイルの設定

対象の設定ファイル		
$COSMINEXUS_HOME/manager/config/adminagent.properties		
#	プロパティ	値
1	adminagent.adapter.allowedHosts	localhost

表 2-3 Management Server 環境設定ファイルの設定

対象の設定ファイル		
$COSMINEXUS_HOME/manager/config/mserver.properties		
#	プロパティ	値
1	mngsvr.myhost.name	localhost
2	com.cosminexus.mngsvr.on_start	true

com.cosminexus.mngsvr.on_start プロパティは Management Server の起動と連動して論理サーバの一括起動を行うための設定です。この設定により，スケールアウトで新たな EC2 インスタンスが生成されたとき，EC2 インスタンス

上の Web システムが自動的に起動し，業務実行可能な状態になるようにします。

設定が完了したら，Management Server のセットアップを行います。以下にセットアップコマンドの実行例を示します。

```
# Management Server のセットアップ
$COSMINEXUS_HOME/manager/bin/mngsvrctl setup -u <管理ユーザ ID> -p <管理ユーザパスワード>
```

続いて，Management Server と運用管理エージェントの自動起動の設定をします。この設定により，OS 起動と同時に Management Server と運用管理エージェントが自動的に起動するようになります。以下に自動起動設定コマンドの実行例を示します。

```
# Management Server と運用管理エージェントの自動起動の設定
$COSMINEXUS_HOME/manager/bin/mngautorun respawn both
```

実行例に示した mngautorun コマンドを実行すると，以下 2 つのサービスユニットが登録され，Management Server と運用管理エージェントが自動起動・自動停止するようになります。

- CoAA.service : 運用管理エージェントを自動起動，自動停止，自動再起動するサービスユニット
- CoMS.service : Management Server を自動起動，自動停止，自動再起動するサービスユニット

実行例に示した mngautorun コマンドでは，引数「respawn」により，自動起動に加えて自動再起動の設定も行っています。自動再起動の設定をすると Management Server や運用管理エージェントがダウンした場合に自動的に再起動するようになります。Management Server と運用管理エージェントがダウンしてしまった場合，収集できる情報が少なくなる，論理サーバが一括停止できない等の影響があるため，自動再起動設定を行ってください。

なお，表 2-3 の設定で，Management Server の起動と連動して論理サーバの一括起動を実行するようにしているため，この Management Server の自動起動の設定により，OS 起動に連動して Management Server と論理サーバが自動的に起動され，Web システムを業務実行可能な状態にすることができます。

ここまでの設定が完了したら，Management Server と運用管理エージェントを起動します。以下に起動コマンドの実行例を示します。

```
# 運用管理エージェントの起動
$COSMINEXUS_HOME/manager/bin/adminagentctl start -sync

# Management Server の起動
$COSMINEXUS_HOME/manager/bin/mngsvrctl start -sync
```

2.2.4 論理 Web サーバのアクセスログの設定

Amazon EC2 Auto Scaling と組み合わせた環境では，EC2 インスタンス上の Web システムは ELB 経由で HTTP リ

クエストを受け付けます。このとき論理 Web サーバのデフォルトの設定では，アクセスログにクライアントの情報として直接のリクエスト元である ELB の情報を出力するため，本来のリクエスト元のクライアントが特定できません。

本来のリクエスト元のクライアント情報が取得できるよう，論理 Web サーバのアクセスログの設定を行う必要があります。ELB からの HTTP リクエストには X-Forwarded-For ヘッダーが付与されているため，X-Forwarded-For ヘッダーに格納されている値をアクセスログに出力するように設定してください。

また，リクエストログとの紐づけのためにスレッド ID を，論理 J2EE サーバの性能解析トレース情報との紐づけのためにルートアプリケーション情報を，アクセスログに出力するように設定してください。これらの紐づけ情報により，ログ解析時のリクエストの追跡可能性を向上させることができます。

これらの設定を簡易構築定義ファイルにて行った場合の例を，付録 A. に掲載します（param-name タグが HttpsdCustomlogFormat の箇所）。アクセスログの採取方法やフォーマットについてはマニュアル「Cosminexus V11 アプリケーションサーバ Cosminexus HTTP Server」を参照してください。

2.2.5 Web システムの構築

マスターEC2 インスタンスにインストールした uCAS 上に Web システムを構築します。本ガイドでは，簡易構築定義ファイルを使って Web システムを構築します。簡易構築定義ファイルを作成する際には，1.1 節で示した Web システムを構築するために，以下の観点に注意してください。

- 論理サーバとして論理 J2EE サーバ，論理 Web サーバ，論理パフォーマンストレーサを 1 つずつ定義する
- 論理 Web サーバから同一 Web システム内の論理 J2EE サーバに HTTP リクエストを転送するように設定する
- 設定に EC2 インスタンス固有の情報を使用しない

上記の観点により，必ず設定すべきパラメタを表 2-4 に示します。また，表 2-4 の設定を行った簡易構築定義ファイルの例を付録 A. に掲載します。

表 2-4 簡易構築定義ファイルの設定

設定の対象となる論理サーバの種類		
論理 J2EE サーバ		
#	パラメタ	値
1	vbroker.se.iiop_tp.host	localhost
2	ejbserver.naming.host	localhost
3	ejbserver.rmi.naming.host	localhost
4	webserver.connector.nio_http.bind_host	localhost

論理サーバが確実に起動・停止できるよう，論理サーバの起動監視時間，停止監視時間，および強制停止監視時間も必要に応じて設定してください。起動監視時間，停止監視時間，および強制停止監視時間の設定方法はマニュアル「Cosminexus V11 アプリケーションサーバ リファレンス 定義編 (サーバ定義) 4 章」を参照してください。

簡易構築定義ファイルの作成が完了したら，Web システムを構築し起動します。以下に簡易構築定義ファイルを使った Web システムの構築，起動コマンドの実行例を示します。

```
# Web システムの構築
$COSMINEXUS_HOME/manager/bin/cmx_build_system -m localhost:<Management Server 接続 HTTP ポート番号> -u <管理ユーザ ID> -p <管理ユーザパスワード> -f <簡易構築定義ファイルのパス>

# Web システムの起動
$COSMINEXUS_HOME/manager/bin/mngsvrutil -m localhost:<Management Server 接続 HTTP ポート番号> -u <管理ユーザ ID> -p <管理ユーザパスワード> -t localhost -k host -s start server
```

Web システムが起動したら，アプリケーションをデプロイします。アプリケーションを論理 J2EE サーバにインポート，起動し，アプリケーションが正常に動作することを確認してください。以下にアプリケーションのインポートおよび起動を行うコマンドの実行例を示します。

```
# アプリケーションのインポート
$COSMINEXUS_HOME/CC/admin/bin/cjimportapp <論理 J2EE サーバ名> -f <J2EE アプリケーションのファイルパス>

# アプリケーションの起動
$COSMINEXUS_HOME/CC/admin/bin/cjstartapp <論理 J2EE サーバ名> -name <アプリケーション名>
```

2.2.6 スケールアウト後の処理の準備

uCAS の動作の前提として，/etc/hosts ファイルにホスト名と IP アドレス（ループバックアドレス不可）の組を定義する必要があります。そのため，スケールアウト時に EC2 インスタンスが生成され，uCAS の各プロセスが起動する前に /etc/hosts ファイルに必要な設定を行うための仕組みが必要になります。

本ガイドでは，Linux の起動時にサービスユニットを自動的に起動することによりこれを実現する方法をご紹介します。/etc/hosts ファイルの編集は，スケールアウト時の AWS による EC2 インスタンスの設定（ホスト名や IP アドレスの割り当て等）の後，かつ uCAS のプロセスが起動する前に実施する必要があります。これを実現するために， ユニットファイルを作成する際には，multi-user.target と multi-user.target が依存するサービスが起動した後かつ，以

下に示すサービスユニットより前に起動するように依存関係を定義してください。また，/etc/hosts ファイルの編集に失敗した場合に以下のサービスユニットが起動しないように依存関係を定義してください。

- CoAA.service : 運用管理エージェントを自動起動，自動停止，自動再起動するサービスユニット
- CoMS.service : Management Server を自動起動，自動停止，自動再起動するサービスユニット

ユニットファイルでは，ユニットの起動時に実行するスクリプトを指定する Service セクションの ExecStart に/etc/hosts ファイルの編集処理を定義したスクリプトを指定しておきます。またスクリプトの処理がタイムアウトしないよう，サービスユニットのタイムアウト値を設定してください。サービスユニットのタイムアウト値は，スクリプトが完了するまでに必要な時間を実測した上で，その時間よりも十分大きな値を設定してください。ユニットファイルを作成したら /etc/systemd/system ディレクトリ以下に配置し，systemctl enable コマンドで自動起動を有効にしてください。

参考として，/etc/hosts ファイルの編集を行うユニットファイルの例を付録 B. に，スクリプトの例を付録 C. に掲載します。

2.2.7 スケールイン前の処理の準備

スケールインの発生に備えて，Linux の終了前に必要な処理を行う仕組みを作成します。Linux の終了前に実施するべき処理は以下の 2 つです。

1. 論理サーバの停止
2. ログの取得と永続化

ログの取得においては，uCAS のログとして snapshot ログを取得してください。snapshot ログを取得する際には snapshotlog コマンドを使用し，一次送付資料と二次送付資料の両方を取得してください。取得方法はマニュアル「Cosminexus V11 アプリケーションサーバ 機能解説 保守／移行編 2 章」を参照してください。

ログの永続化においては，取得した snapshot ログを永続化可能なストレージ（Amazon S3, Amazon EFS 等）へ保存してください。snapshot ログを収集しておくことで，障害問い合わせ時に利用することができるほか，アプリケーションのエラーやレスポンスタイム悪化などの原因分析にも役立てることができます。snapshot ログを永続化可能なストレージに保存する際，shapshot ログの ZIP ファイル名は，収集元の判別がつき，他の ZIP ファイル名と重複しないファイル名（<EC2 インスタンスの IP アドレス>+<ログの取得日時>など）に変更してください。また，スケールインの度に永続化する snapshot ログの数が増えていくため，保存期間はシステム要件に合わせて決定し，定期的に不要なログを削除することを推奨します。

本ガイドでは一例として，サービスユニットの停止時にスクリプトを実行することによりこれを実現する方法をご紹介します。Management Server および運用管理エージェントのプロセスが終了する前に停止するサービスユニットを作成し，そのサービスユニットが自動起動するように設定します。これを実現するために，ユニットファイルを作成する際には以下に示すサービスユニットより後に起動するようにサービスユニットの依存関係を定義してください。サービスユニットの停止は起動と逆順で実行されるため，このように定義することで作成するサービスユニットの停止処理を uCAS のプロセス停止前に実行することができます。

- CoAA.service : 運用管理エージェントを自動起動，自動停止，自動再起動するサービスユニット
- CoMS.service : Management Server を自動起動，自動停止，自動再起動するサービスユニット

ユニットファイルでは，サービスユニットの停止時に実行するコマンドを指定する Service セクションの ExecStop にスケールイン前の処理を定義したスクリプトを指定しておきます。またスクリプトの処理がタイムアウトしないよう，サービスユニットのタイムアウト値を設定してください。サービスユニットのタイムアウト値は，スクリプトが完了するまでに必要な時間を実測した上で，その時間よりも十分大きな値を設定してください。ユニットファイルを作成したら /etc/systemd/system ディレクトリ以下に配置し，systemctl enable コマンドで自動起動を有効にしてください。本設定を行った場合，スケールイン以外の Linux のシャットダウン時にもスクリプトが実行されるため注意してください。

参考として，ユニットファイルの例を付録 B. に，スクリプトの例を付録 C. に掲載します。

2.2.8 uCAS プロセスの停止

AMI の作成に向けて，uCAS の各プロセスを停止させておきます。また，Web システム構築時のログとして snapshot ログを取得し，永続化可能なストレージに保存してください。以下に各プロセスの停止およびログの収集を行うコマンドの実行例を示します。

```
# 論理サーバの停止
$COSMINEXUS_HOME/manager/bin/mngsvrutil -m localhost:<Management Server 接続HTTP ポート番号> -u <管理ユーザ ID> -p <管理ユーザパスワード> -t localhost -k host -s stop server

# snapshot ログの取得
$COSMINEXUS_HOME/manager/bin/snapshotlog <出力ファイル名 1>
$COSMINEXUS_HOME/manager/config/snapshotlog.conf
$COSMINEXUS_HOME/manager/bin/snapshotlog <出力ファイル名 2>
$COSMINEXUS_HOME/manager/config/snapshotlog.2.conf

# Management Server の停止
$COSMINEXUS_HOME/manager/bin/mngsvrctl stop -sync

# 運用管理エージェントの停止
$COSMINEXUS_HOME/manager/bin/adminagentctl stop -sync
```

なお，この時論理 J2EE サーバ上で起動させていたアプリケーションは停止させないでください。スケールアウトにより生成された EC2 インスタンスを自動的に業務実行可能な状態にするためには，Web システムが起動した時にアプリケーションが開始状態になっている必要があります。

2.2.9 構築時のログの削除

Web システム構築時のログが運用時のログと混在しないように，ここまでに出力されたログを削除することを推奨します。 (1) ～ (3) に従って構築時のログを削除してください。

(1) Cosminexus Component Container のログ

次に示すディレクトリ内のファイルを確認し，ファイルが存在する場合には削除してください。削除する際にはディレクトリは削除しないようにしてください。

$COSMINEXUS_HOME/CC/admin/logs/*
$COSMINEXUS_HOME/CC/admin/logs/CC/maintenance/*
$COSMINEXUS_HOME/CC/admin/logs/CC/maintenance/mmap/*
$COSMINEXUS_HOME/CC/admin/logs/CC/rmi/*
$COSMINEXUS_HOME/CC/admin/logs/CC/rmi/mmap/*
$COSMINEXUS_HOME/CC/admin/logs/mmap/*
$COSMINEXUS_HOME/CC/admin/logs/TPB/logj/comtrc/*
$COSMINEXUS_HOME/CC/admin/logs/TPB/logj/mdltrc/*
$COSMINEXUS_HOME/CC/admin/logs/TPB/logj/*
$COSMINEXUS_HOME/CC/client/logs/*
$COSMINEXUS_HOME/CC/client/logs/mmap/*
$COSMINEXUS_HOME/CC/client/logs/system/ejbcl/mmap/*
$COSMINEXUS_HOME/CC/client/logs/system/ejbcl/*
$COSMINEXUS_HOME/CC/client/logs/system/TPB/logj/comtrc/*
$COSMINEXUS_HOME/CC/client/logs/system/TPB/logj/mdltrc/*
$COSMINEXUS_HOME/CC/client/logs/system/TPB/logj/*
$COSMINEXUS_HOME/CC/web/containers/<Web コンテナサーバ名>/logs/TPB/logj/comtrc/*
$COSMINEXUS_HOME/CC/web/containers/< Web コンテナサーバ名>/logs/TPB/logj/mdltrc/*
$COSMINEXUS_HOME/CC/web/containers/< Web コンテナサーバ名>/logs/TPB/logj/*
$COSMINEXUS_HOME/CC/web/containers/< Web コンテナサーバ名>/logs/*
$COSMINEXUS_HOME/manager/spool/default/*
$COSMINEXUS_HOME/manager/spool/lsinfo/*
$COSMINEXUS_HOME/manager/tmp/*

$COSMINEXUS_HOME/manager/setup/log/*

$COSMINEXUS_HOME/manager/setup/log/maintenance/*

<作業ディレクトリ>/ejb/<論理 J2EE サーバ名>/logs/watch/*

<作業ディレクトリ>/ejb/<論理 J2EE サーバ名>/logs/CC/maintenance/*

<作業ディレクトリ>/ejb/<論理 J2EE サーバ名>/logs/CC/rmi/*

<作業ディレクトリ>/ejb/<論理 J2EE サーバ名>/logs/connectors/*

<作業ディレクトリ>/ejb/<論理 J2EE サーバ名>/logs/connectors/<アプリケーション表示名>/*

<作業ディレクトリ>/ejb/<論理 J2EE サーバ名>/logs/connectors/TEST#<アプリケーション表示名>/*

<作業ディレクトリ>/ejb/<論理 J2EE サーバ名>/logs/TPB/logj/mdltrc/*

<作業ディレクトリ>/ejb/<論理 J2EE サーバ名>/logs/TPB/logj/comtrc/*

<作業ディレクトリ>/ejb/<論理 J2EE サーバ名>/logs/TPB/logj/namelog/*

<作業ディレクトリ>/ejb/<論理 J2EE サーバ名>/logs/TPB/logj/*

<作業ディレクトリ>/ejb/<論理 J2EE サーバ名>/logs/WS/*

<作業ディレクトリ>/ejb/<論理 J2EE サーバ名>/logs/WS/maintenance/*

<作業ディレクトリ>/ejb/<論理 J2EE サーバ名>/logs/*

<作業ディレクトリ>/ejb/<論理 J2EE サーバ名>/stats/*

<Manager ログ出力ディレクトリ>/maintenance/*

<Manager ログ出力ディレクトリ>/maintenance/mmap/*

<Manager ログ出力ディレクトリ>/maintenance/CC/maintenance/*

<Manager ログ出力ディレクトリ>/maintenance/CC/rmi/*

<Manager ログ出力ディレクトリ>/maintenance/WS/*

<Manager ログ出力ディレクトリ>/maintenance/WS/maintenance/*

<Manager ログ出力ディレクトリ>/maintenance/http/maintenance/thr/*

<Manager ログ出力ディレクトリ>/maintenance/http/maintenance/comm/*

<Manager ログ出力ディレクトリ>/maintenance/watch/*

<Manager ログ出力ディレクトリ>/message/*

<Manager ログ出力ディレクトリ>/message/mmap/*

<Manager ログ出力ディレクトリ>/trace/*

<Manager ログ出力ディレクトリ>/trace/mmap/*

<Manager ログ出力ディレクトリ>/*

<snapshot ログの出力先ディレクトリ>/*

<監査ログ出力ディレクトリ>/*

<監査ログのメッセージログ出力ディレクトリ>/*

<監査ログのメッセージログ出力ディレクトリ>/mmap/*

<監査ログの例外情報出力ディレクトリ>/*

<監査ログの例外情報出力ディレクトリ>/mmap/*

なお，上のリスト中に登場する各ディレクトリのデフォルト値は表 2-5 の通りです。

表 2-5 各ディレクトリのデフォルト値

#	条件	前提とする値
1	作業ディレクトリ	/opt/Cosminexus/CC/server/public
2	Manager ログ出力ディレクトリ	/opt/Cosminexus/manager/log
3	snapshot ログの出力先ディレクトリ	<Manager ログ出力ディレクトリ>/snapshot
4	監査ログ出力ディレクトリ	/opt/Cosminexus/auditlog
5	監査ログのメッセージログ出力ディレクトリ	/opt/Cosminexus/auditlog
6	監査ログの例外情報出力ディレクトリ	/opt/Cosminexus/auditlog

(2) Cosminexus HTTP Server のログ

以下の設定に指定したファイルを削除してください。

- 論理 Web サーバの Web サーバ定義ファイル（httpsd.conf）の PidFile ディレクティブに指定したファイル
- 論理 Web サーバの Web サーバ定義ファイル（httpsd.conf）のログ関連ディレクティブに指定したファイル

(3) Cosminexus Performance Tracer のログ

環境変数 PRFSPOOL に指定したディレクトリ内にあるファイルとディレクトリを削除してください。

ここまでの操作が完了したらマスターEC2 インスタンスを停止し，マスターEC2 インスタンスの AMI を作成してください。その後，AWS のドキュメントに従って，マスターEC2 インスタンスの AMI を使用した Amazon EC2 Auto Scaling の環境を作成してください。

3 障害対策の考え方

この章では，本ガイドで構築したシステムにおける障害対策の考え方について説明します。

本章の構成

クラウド上のシステム設計で求められる観点の 1 つに可用性があります。高い可用性を備えたシステムにするためには，特定の EC2 インスタンスで障害が発生した場合，その影響を最小限に留め，システム全体では業務やサービスを継続できるような設計が求められます。uCAS のシステムで想定される主な障害に対する対策の考え方を示しますのでシステム設計の参考にして下さい。

なお，本ガイドで構築したシステム全体における HTTP リクエストの追跡可能性を向上させるため，ELB でアクセスログを取得することを推奨します。ELB のアクセスログには，ELB がどのクライアントから HTTP リクエストを受信し，どの Web システムに転送し，どのような応答を受けたのかといった情報が記録されています。このアクセスログと uCAS のログを合わせて確認することで，システム全体における HTTP リクエストの処理が追跡できるようになります。

3.1 Web システムの起動に失敗した場合

Web システムの起動に失敗した場合，その Web システムでは HTTP リクエストを受け付けできない，または処理できない状態になります。Web システムがこの状態に陥ったことを自動的に検知する方法として，ELB のヘルスチェックを利用することを推奨します。スケールアウトにより生成された EC2 インスタンス自体の起動に成功していても，Web システムが起動失敗している場合，ELB が新たな Web システムへ HTTP リクエストを振り分け始める前に実施するヘルスチェックで異常を検知することができます。これにより，障害が発生した Web システムへの HTTP リクエストの振り分けを未然に防止するとともに，Amazon EC2 Auto Scaling による EC2 インスタンス再作成を促すことができます。

これを実現するためには，ELB のヘルスチェックで Web システムが正常に動作していることが確認できるようにする必要があります。ヘルスチェックの「パス」の設定では，Web システム内の論理 Web サーバと論理 J2EE サーバの両方が正常に動作していることが確認できるよう，論理 J2EE サーバ上のアプリケーションが正常起動していないとアクセスできないようなリクエスト先を設定するようにしてください。

3.2 Management Server や運用管理エージェントがダウンした場合

Management Server や運用管理エージェントがダウンした場合に備えて，2.2.3 項で紹介した方法を用いて Management Server と運用管理エージェントを自動再起動するように設定しておくことを推奨します。Management Server や運用管理エージェントがダウンしてしまうと，運用管理エージェントを介した各論理サーバへの操作ができなくなる他，稼働情報等の情報も取得できなくなってしまいます。

3.3 論理 J2EE サーバや論理 Web サーバがダウンした場合

論理 J2EE サーバや論理 Web サーバがダウンした場合，論理 J2EE サーバや論理 Web サーバの再起動は行わずに ELB のヘルスチェックで異常を検知させてください。ELB による異常検知により，Amazon EC2 Auto Scaling に EC2 インスタンスを再作成させることで Web システムを正常化させます。

論理 J2EE サーバおよび論理 Web サーバの自動再起動の設定は簡易構築定義ファイルで行います。設定方法についてはマニュアル「Cosminexus V11 アプリケーションサーバ リファレンス 定義編（サーバ定義）4 章」を参照してください。

ELB のヘルスチェックで異常検知されるまでは，ダウンした論理サーバにも ELB から HTTP リクエストが振り分けられてしまいます。そのため，ELB のヘルスチェック設定を検討する際には，異常検知をできるだけ早めることを検討し，ELB からダウンした論理サーバへの HTTP リクエストの振り分けを停止できるようにしてください。ヘルスチェックの設定項目のうち，「間隔」と「非正常のしきい値」の設定が異常判定までの時間に関係します。ただし，「間隔」が短い場合には Web システムへの負荷に影響する可能性があること，「非正常のしきい値」が少ない場合には，Web システムが正常稼働しているにもかかわらず高負荷等の影響によりヘルスチェックに失敗してしまい，誤って異常検知されてしまう可能性があることに注意してください。これらの注意すべき事象の発生リスクと論理サーバの異常検知にかかる時間はトレードオフの関係であるため，システム要件に応じて適切な値を検討・設定してください。

付録A. 簡易構築定義ファイルの例

簡易構築定義ファイルの例を以下に示します。簡易構築定義ファイルのテンプレートを基に作成したもので，Amazon EC2 Auto Scaling に対応する Web システムに必要な設定や推奨する設定を追加・編集した部分を赤字で強調表示しています。

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- All Rights Reserved. Copyright (C) 2020, Hitachi, Ltd. -->
<model-definition
xmlns="http://www.cosminexus.com/mngsvr/schema/ModelDefinition-2.5">
  <web-system>
    <name>MyWebSystem</name>
    <!--Specify the configurations for the Tier.-->
    <tier>
      <tier-type>combined-tier</tier-type>
      <!--combined-tier includes a Web Server(HWS), a J2EE Server(CCC) and a
Performance Tracer on a host.-->

      <!-- Performance Tracer -->
      <configuration>
        <logical-server-type>performance-tracer</logical-server-type>
        <param>
          <param-name>mstartup.start.watchtime</param-name>
          <param-value>60</param-value>
        </param>
        <param>
          <param-name>mstartup.watchtime</param-name>
          <param-value>600</param-value>
        </param>
        <param>
          <param-name>mstartup.restartcnt</param-name>
          <param-value>1</param-value>
        </param>
        <param>
          <param-name>mstartup.retrywait</param-name>
          <param-value>60</param-value>
        </param>
        <param>
          <param-name>PrfTraceLevel</param-name>
          <param-value>00140000</param-value>
        </param>
        <param>
          <param-name>PrfTraceFileSize</param-name>
          <param-value>32768</param-value>
        </param>
        <param>
          <param-name>PrfTraceCount</param-name>
```

```
    <param-value>4</param-value>
  </param>
</configuration>

<!-- J2EE Server -->
<configuration>
  <logical-server-type>j2ee-server</logical-server-type>
  <param>
    <param-name>ejbserver.http.port</param-name>
    <param-value>28008</param-value>
  </param>
  <param>
    <param-name>webserver.connector.nio_http.port</param-name>
    <param-value>8008</param-value>
  </param>
  <param>
    <param-name>ejbserver.naming.port</param-name>
    <param-value>900</param-value>
  </param>
  <param>
    <param-name>ejbserver.rmi.naming.port</param-name>
    <param-value>23152</param-value>
  </param>
  <param>
    <param-name>ejbserver.rmi.remote.listener.port</param-name>
    <param-value>23550</param-value>
  </param>
  <param>
    <param-name>mstartup.start.watchtime</param-name>
    <param-value>600</param-value>
  </param>
  <param>
    <param-name>mstartup.watchtime</param-name>
    <param-value>60</param-value>
  </param>
  <param>
    <param-name>mstartup.restartcnt</param-name>
    <param-value>0</param-value>
  </param>
  <param>
    <param-name>mstartup.retrywait</param-name>
    <param-value>60</param-value>
  </param>
  <param>
    <param-name>ejbserver.ext.method_observation.interval</param-name>
    <param-value>10</param-value>
  </param>
  <param>
    <param-name>ejbserver.manager.agent.MEventAgent.enabled</param-name>
```

```
        <param-value>true</param-value>
      </param>
      <param>
        <param-name>ejbserver.jta.TransactionManager.defaultTimeOut</param-
name>
        <param-value>60</param-value>
      </param>
      <param>
        <param-
name>ejbserver.logger.channels.define.MaintenanceLogFile.filesize</param-
name>
        <param-value>16777216</param-value>
      </param>
      <param>
        <param-
name>ejbserver.logger.channels.define.MaintenanceLogFile.filenum</param-name>
        <param-value>4</param-value>
      </param>
      <param>
        <param-
name>ejbserver.logger.channels.define.WebServletLogFile.filesize</param-name>
        <param-value>16777216</param-value>
      </param>
      <param>
        <param-
name>ejbserver.logger.channels.define.WebServletLogFile.filenum</param-name>
        <param-value>4</param-value>
      </param>
      <param>
        <param-name>ejbserver.connector.logwriter.filesize</param-name>
        <param-value>2097152</param-value>
      </param>
      <param>
        <param-name>ejbserver.connector.logwriter.filenum</param-name>
        <param-value>4</param-value>
      </param>
      <param>
        <param-name>ejbserver.webj2ee.connectionAutoClose.enabled</param-
name>
        <param-value>true</param-value>
      </param>
      <param>
        <param-name>use.security</param-name>
        <param-value>false</param-value>
      </param>
      <param>
        <param-name>add.jvm.arg</param-name>
        <param-value>-Xms512m</param-value>
        <param-value>-Xmx512m</param-value>
```

```
        <param-value>-XX:MetaspaceSize=128m</param-value>
        <param-value>-XX:MaxMetaspaceSize=128m</param-value>
        <param-value>-XX:+HitachiVerboseGC</param-value>
        <param-value>-XX:HitachiVerboseGCIntervalTime=0</param-value>
        <param-value>-XX:+HitachiVerboseGCPrintCause</param-value>
        <param-value>-XX:+HitachiOutputMilliTime</param-value>
        <param-value>-XX:+HitachiCommaVerboseGC</param-value>
        <param-value>-XX:HitachiJavaLogFileSize=4096k</param-value>
        <param-value>-XX:HitachiJavaLogNumberOfFile=4</param-value>
      </param>
      <param>
        <param-
name>ejbserver.connectionpool.applicationAuthentication.disabled</param-name>
        <param-value>true</param-value>
      </param>
      <param>
        <param-name>ex.properties</param-name>
        <param-value>sun.nio.cs.map=Windows-31J/Shift_JIS</param-value>
      </param>
      <param>
        <param-name>ejbserver.management.statistics.interval</param-name>
        <param-value>60</param-value>
      </param>
      <param>
        <param-name>ejbserver.management.stats_file.num</param-name>
        <param-value>7</param-value>
      </param>
      <param>
        <param-name>ejbserver.management.stats_file.base_time</param-name>
        <param-value>0</param-value>
      </param>
      <param>
        <param-name>ejbserver.management.stats_file.period</param-name>
        <param-value>24</param-value>
      </param>
      <param>
        <param-name>vbroker.se.iiop_tp.host</param-name>
        <param-value>localhost</param-value>
      </param>
      <param>
        <param-name>ejbserver.naming.host</param-name>
        <param-value>localhost</param-value>
      </param>
      <param>
        <param-name>ejbserver.rmi.naming.host</param-name>
        <param-value>localhost</param-value>
      </param>
      <param>
        <param-name>webserver.connector.nio_http.bind_host</param-name>
```

```
    <param-value>localhost</param-value>
  </param>
</configuration>

<!-- Web Server -->
<configuration>
  <logical-server-type>web-server</logical-server-type>
  <param>
    <param-name>ServerName</param-name>
    <param-value>www.example.com</param-value>
  </param>
  <param>
    <param-name>Listen</param-name>
    <param-value>80</param-value>
  </param>
  <param>
    <param-name>User</param-name>
    <param-value>bin</param-value>
  </param>
  <param>
    <param-name>Group</param-name>
    <param-value>bin</param-value>
  </param>
  <param>
    <param-name>mstartup.start.watchtime</param-name>
    <param-value>60</param-value>
  </param>
  <param>
    <param-name>mstartup.watchtime</param-name>
    <param-value>60</param-value>
  </param>
  <param>
    <param-name>mstartup.restartcnt</param-name>
    <param-value>0</param-value>
  </param>
  <param>
    <param-name>mstartup.retrywait</param-name>
    <param-value>60</param-value>
  </param>
  <param>
    <param-name>ThreadsPerChild</param-name>
    <param-value>20</param-value>
  </param>
  <param>
    <param-name>ThreadsPerChild.worker</param-name>
    <param-value>20</param-value>
  </param>
  <param>
    <param-name>MaxClients.worker</param-name>
```

```
        <param-value>20</param-value>
      </param>
      <param>
        <param-name>StartServers.worker</param-name>
        <param-value>1</param-value>
      </param>
      <param>
        <param-name>HWSKeepStartServers</param-name>
        <param-value>On</param-value>
      </param>
      <param>
        <param-name>KeepAliveTimeout</param-name>
        <param-value>3</param-value>
      </param>
      <param>
        <param-name>LogLevel</param-name>
        <param-value>info</param-value>
      </param>
      <param>
        <param-name>HWSLogTimeVerbose</param-name>
        <param-value>On</param-value>
      </param>
      <param>
        <param-name>HttpsdCustomlogFormat</param-name>
        <param-value>"%h %l %u %t ¥"%r¥" %>s %b %T %P %{tid}P
¥"%{hws_ap_root}n¥" %{X-Forwarded-For}i"</param-value>
      </param>
      <param>
        <param-name>HttpsdCustomLogFileDir</param-name>
        <param-value>logs</param-value>
      </param>
      <param>
        <param-name>CustomDivideTimeInterval</param-name>
        <param-value>86400</param-value>
      </param>
      <param>
        <param-name>CustomDivideFileNum</param-name>
        <param-value>8</param-value>
      </param>
      <param>
        <param-name>CustomDivideTimeDifference</param-name>
        <param-value>540</param-value>
      </param>
      <param>
        <param-name>HttpsdRequestLogFileDir</param-name>
        <param-value>logs</param-value>
      </param>
      <param>
        <param-name>RequestDivideTimeInterval</param-name>
```

```
      <param-value>86400</param-value>
    </param>
    <param>
      <param-name>RequestDivideFileNum</param-name>
      <param-value>8</param-value>
    </param>
    <param>
      <param-name>RequestDivideTimeDifference</param-name>
      <param-value>540</param-value>
    </param>
    <param>
      <param-name>HttpsdErrorLogFileDir</param-name>
      <param-value>logs</param-value>
    </param>
    <param>
      <param-name>HttpsdErrorMethod</param-name>
      <param-value>Wrap</param-value>
    </param>
    <param>
      <param-name>ErrorWraparoundFilesize</param-name>
      <param-value>8192</param-value>
    </param>
    <param>
      <param-name>ErrorWraparoundFileNum</param-name>
      <param-value>5</param-value>
    </param>
  </configuration>
</tier>

<!--Specify the hosts for each Service Unit.-->
<unit>
  <name>unit1</name>
  <allocated-host>
    <host-ref>localhost</host-ref>
    <hosts-for>combined-tier</hosts-for>
    <define-server>
      <logical-server-name>MyPerformanceTracer</logical-server-name>
      <logical-server-type>performance-tracer</logical-server-type>
    </define-server>
    <define-server>
      <logical-server-name>MyJ2EEServer</logical-server-name>
      <logical-server-type>j2ee-server</logical-server-type>
    </define-server>
    <define-server>
      <logical-server-name>MyWebServer</logical-server-name>
      <logical-server-type>web-server</logical-server-type>
    </define-server>
  </allocated-host>
</unit>
```

```
  </web-system>

  <!--Specify the settings for the host.-->
  <host>
    <host-name>localhost</host-name>
    <agent-host>localhost</agent-host>
    <agent-port>20295</agent-port>
  </host>
</model-definition>
```

付録B. ユニットファイルの例

スケールアウト後処理定義用のユニットファイルの定義例を以下に示します。

```
[Unit]
Description=Sequence of before start uCAS
After=multi-user.target
Before=CoAA.service CoMS.service

[Service]
Type=oneshot
ExecStart=<スケールアウト後処理定義用スクリプトを実行するコマンド>
# スクリプトが完了するまでの時間を実測し，その時間よりも十分大きな値を設定してください。
TimeoutStartSec=<スケールアウト後処理定義用スクリプトのタイムアウト時間>

[Install]
WantedBy=multi-user.target
RequiredBy=CoAA.service CoMS.service
```

スケールイン前処理定義用のユニットファイルの定義例を以下に示します。

```
[Unit]
Description=Sequence of before terminate uCAS
After=multi-user.target CoAA.service CoMS.service
Wants=CoAA.service CoMS.service

[Service]
Type=oneshot
RemainAfterExit=yes
ExecStop=<スケールイン前処理定義用スクリプトを実行するコマンド>
# スクリプトが完了するまでの時間を実測し，その時間よりも十分大きな値を設定してください。
TimeoutStopSec=<スケールイン前処理定義用スクリプトのタイムアウト時間>

[Install]
WantedBy=multi-user.target
```

付録C. スクリプトの例

スケールアウト後処理定義用スクリプトの例を示します。

```
#!/bin/sh

# hosts ファイルに EC2 インスタンスのホスト名の定義がすでに存在する場合には削除する
sudo sed -i -e "/$(hostname)/d" /etc/hosts
# hosts ファイルに EC2 インスタンスの IP アドレスに対するホスト名の定義を記載する
sudo sed -i -e "1i $(hostname -i) $(hostname)" /etc/hosts
```

スケールイン前処理定義用スクリプトの例を示します。ログの永続化処理では，EC2 インスタンスのインスタンスストアや EC2 インスタンスと紐づいた EBS ではなく，永続化可能なストレージ（Amazon S3, Amazon EFS 等）にログを保存してください。

```
#!/bin/sh

COSMINEXUS_HOME=<uCAS のインストールディレクトリの絶対パス>
SNAPSHOT_LOGDIR=<snapshot ログの出力先ディレクトリ>
SNAPSHOT_PREFIX=snapshot-localhost-log-
ZIP_FILE_NAME=snapshot-$(hostname -i | sed -e 's/¥./-/g')-log-$(date
+%Y%m%d%H%M%S).2

# 全ての論理サーバを停止する。
sudo $COSMINEXUS_HOME/manager/bin/mngsvrutil -m localhost:<Management Server 接
続 HTTP ポート番号> -u <管理ユーザ ID> -p <管理ユーザパスワード> -t localhost -k host -
s stop server


# snapshotlog コマンドを使用して snapshot ログを取得し，ZIP ファイルにまとめる。
sudo $COSMINEXUS_HOME/manager/bin/snapshotlog $SNAPSHOT_LOGDIR/<出力ファイル名 1>
$COSMINEXUS_HOME/manager/config/snapshotlog.conf
sudo $COSMINEXUS_HOME/manager/bin/snapshotlog $SNAPSHOT_LOGDIR/<出力ファイル名 2>
$COSMINEXUS_HOME/manager/config/snapshotlog.2.conf
(cd $SNAPSHOT_LOGDIR && sudo zip $ZIP_FILE_NAME.zip <出力ファイル名 1> <出力ファイ
ル名 2>)

# snapshot ログの取得途中で作成されたファイルを削除する。
sudo rm -f $SNAPSHOT_LOGDIR/<出力ファイル名 1>
sudo rm -f $SNAPSHOT_LOGDIR/<出力ファイル名 2>


# snapshot ログが含まれる ZIP ファイルのファイル名称を変更
for i in $(find $SNAPSHOT_LOGDIR -name "$SNAPSHOT_PREFIX*")
do
    # snapshot ログの取得時間および定義送付資料の種類を表す文字列を取得
    BEFORE_FILENAME=$i
```

```
    TIMESTAMP=$(echo $BEFORE_FILENAME | cut -c $((${#SNAPSHOT_LOGDIR} + 25))-)
    # IP アドレスおよび snapshot ログ取得時間を使用してファイル名称を変更
    AFTER_FILENAME=$SNAPSHOT_LOGDIR/snapshot-$(hostname -i | sed -e 's/¥./-
/g')-log-$TIMESTAMP
    sudo mv $BEFORE_FILENAME $AFTER_FILENAME
done


# ここでログの永続化処理を行ってください。
```

－以上－