TOSHIBA

オープンソースカンファレンス2022 Online/Fall セミナープログラム

Ansible・Terraform・Packerを活用したソフトウェア開発環境の自動構築

株式会社東芝 ソフトウェア技術センター
安村 明子
akiko.yasumura@toshiba.co.jp

© 2022 Toshiba Corporation

Contents

01 はじめに

02 大規模な、ソフトウェア開発環境の構築手順の概要

03 環境構築に使用する構成管理ツール (IaCツール)

04 ソフトウェア開発環境の構築方法の詳細

05 おわりに

© 2022 Toshiba Corporation

自己紹介

株式会社 東芝 ソフトウェア技術センター　安村 明子

組み込みソフトウェア開発等に従事。現在はソフトウェア開発環境に関する業務を担当。

ソフトウェア技術センターの技術領域

共通要素技術	共創ソフトウェア開発技術	ソフトウェアエンジニアリング技術
OSSを中心とした共通基盤ソフトウェアに関する技術	東芝グループの標準的なソフトウェア開発環境、開発プロセス構築に関する技術	ソフトウェア開発の効率化と信頼性向上に関する技術
・データベース ・リアルタイムWebアプリ ・Linux	・共創ソフトウェア開発環境 ・ソフトウェア開発環境 ・ソフトウェア開発プロセス	・ソフトウェア設計 ・システムズエンジニアリング ・テスト設計・テスト自動化

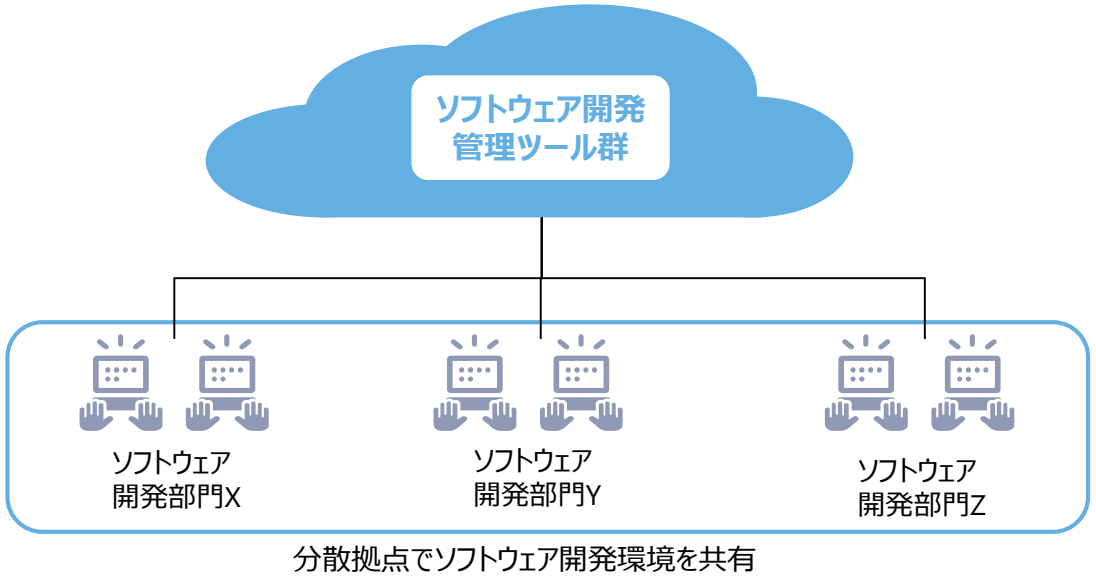
https://www.global.toshiba/jp/technology/corporate/swc.html

東芝グループ向けのソフトウェア開発環境を提供

© 2022 Toshiba Corporation

所属チームのお仕事

- 弊社内には、多くのソフトウェア開発部門がある
- 各部門で開発に必要なツール環境を構築すると、部門毎にサーバリソースやサーバ管理者が必要になる。また、ノウハウが分散する。



分散拠点でソフトウェア開発環境を共有

数百のソフトウェア開発部門に、開発管理ツール群をSaaSサービスで提供

© 2022 Toshiba Corporation

本日の内容

- ソフトウェア開発部門ごとに、それぞれのニーズにあった環境を即時提供するため、IaCツール(Ansible, Terraform,　Packer)を用いて、環境構築を自動化している。
  複数の開発環境を効率よく自動構築・管理するための構築方法の一例をご紹介。
- 構築する際のポイント
  - IaC ツール Ansible・Terraform・Packer をそれぞれ自動構築のどこで使用するか
  - 構築する環境ごとに異なる部分の取り扱い方

IaC： Infrastructure as Code

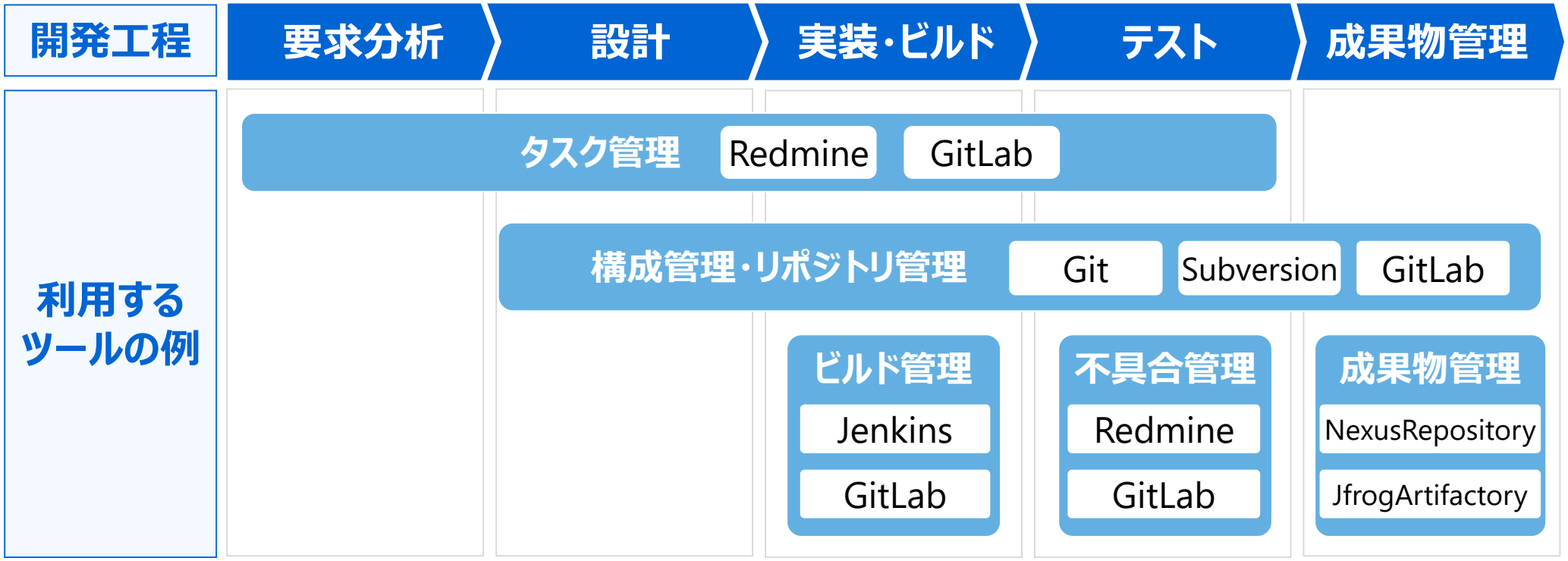
© 2022 Toshiba Corporation

01

はじめに

© 2022 Toshiba Corporation

ソフトウェア開発を支える様々なOSSツール

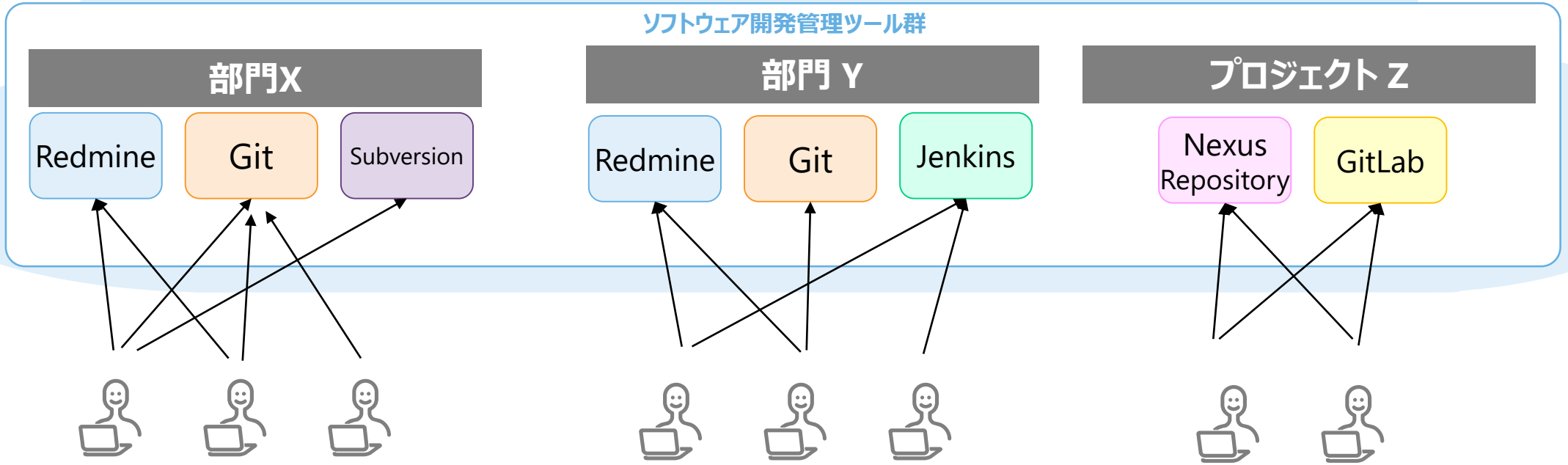


各開発工程において、ソフトウェア開発用OSSツールが複数存在

© 2022 Toshiba Corporation

各ソフトウェア開発部門やプロジェクトでのツール利用イメージ1

ソフトウェア開発部門やプロジェクトによって使用するツールが異なる



※ ソフトウェア開発部門単位またはプロジェクト単位で、ニーズに応じてツールを提供していますが、以降のスライドでは、簡単化のためプロジェクト単位での説明とします。

© 2022 Toshiba Corporation

各ソフトウェア開発プロジェクトでのツール利用イメージ２

ソフトウェア開発プロジェクトによってツールの設定を変えて提供

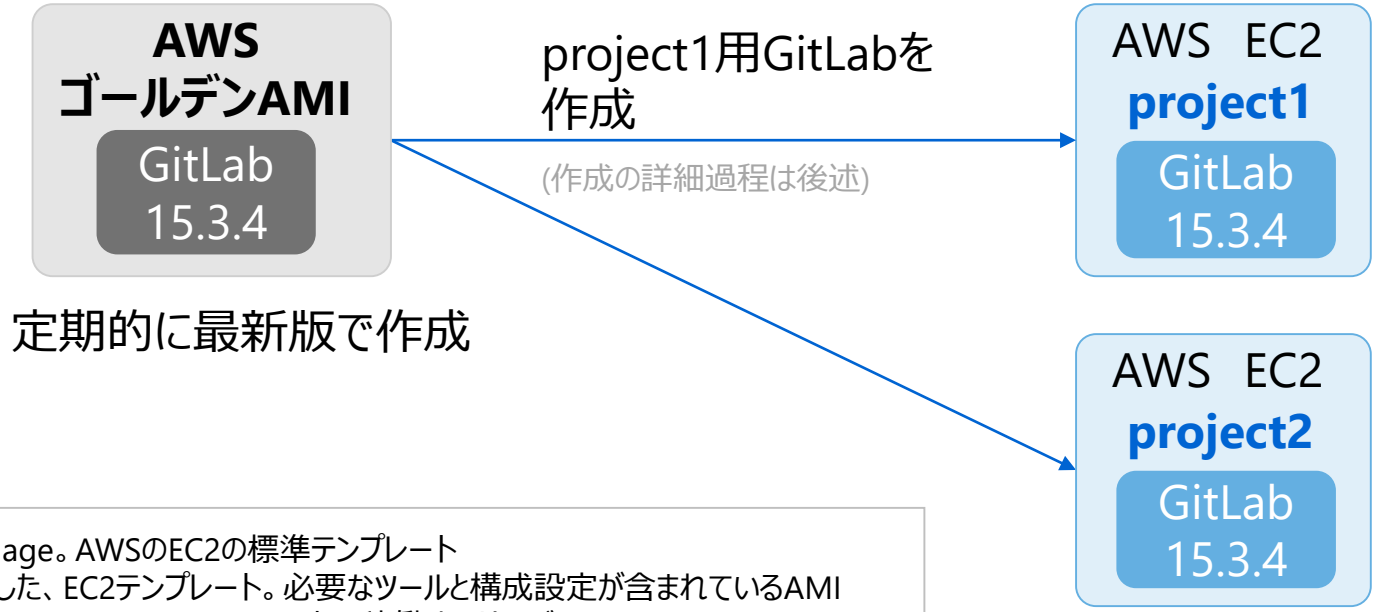
- ソフトウェア開発プロジェクト毎、ツール毎にURLを変えて提供
  - https://xxx.toshiba.co.jp/**projectX**/**redmine**/
  - https://xxx.toshiba.co.jp/**projectX**/**gitlab**/
  - https://xxx.toshiba.co.jp/**projectY**/**redmine**/
- プロジェクトのニーズに応じて、ツールバージョンや、機能の有効／無効などの設定を行う
- バックアップ等のメンテナンス時刻は、ユーザニーズや負荷などを考慮して設定している

GitLab設定例	バージョン	Registry機能	Pages機能	バックアップ時刻
開発プロジェクトX	14.7.7	有	有	24:30
開発プロジェクトY	15.3.4	無	有	25:00

© 2022 Toshiba Corporation

自動生成するシステム概要

- AWSを利用
- プロジェクト・ツール毎に、AWSに1インスタンスを構築
- 短時間で提供できるように、あらかじめ元となる環境（AMI*）を作成しておく
- AMIを元に作成するため、1ツールの環境構築は約10-20分で可能



AMI：Amazon Machine Image。AWSのEC2の標準テンプレート
ゴールデンAMI：独自に作成した、EC2テンプレート。必要なツールと構成設定が含まれているAMI
EC2インスタンス： Amazon Elastic Compute Cloud上で稼働するサーバ

© 2022 Toshiba Corporation

02

大規模な、ソフトウェア開発環境の構築手順の概要

© 2022 Toshiba Corporation

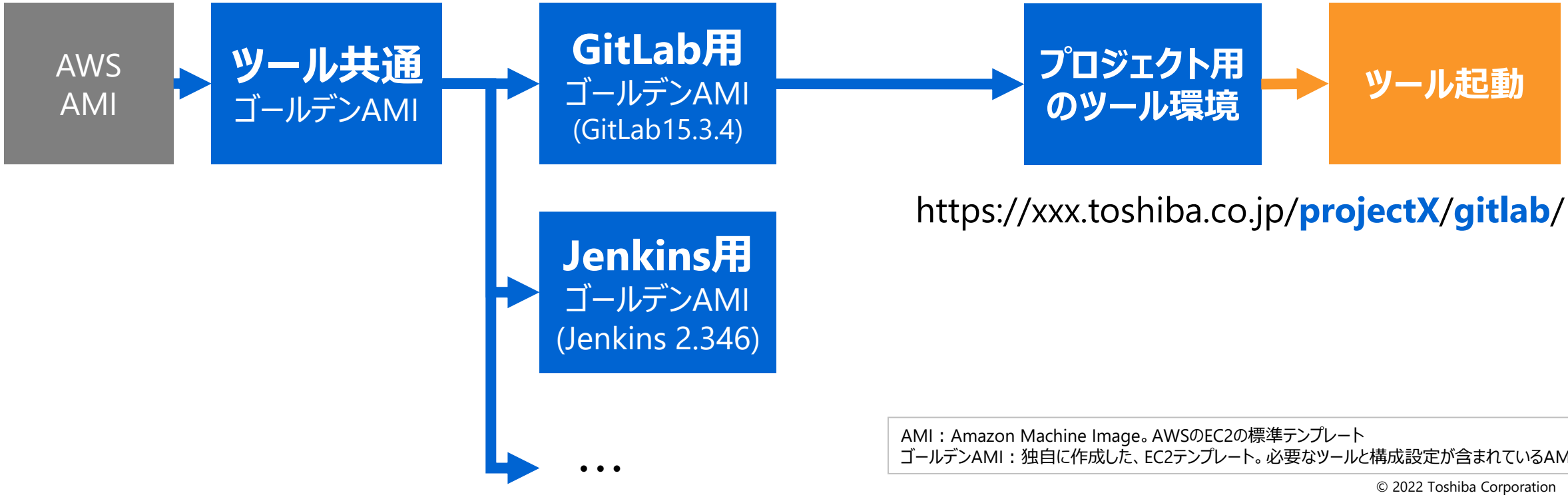
環境構築の流れ

事前準備（半年に１度更新）

利用者からの依頼時に実行

ゴールデンAMIを作成

プロジェクト用のツール環境を構築



https://xxx.toshiba.co.jp/**projectX**/**gitlab**/

AMI：Amazon Machine Image。AWSのEC2の標準テンプレート
ゴールデンAMI：独自に作成した、EC2テンプレート。必要なツールと構成設定が含まれているAMI

© 2022 Toshiba Corporation

環境構築手順の概要

4つのステップで環境を構築

ゴールデンAMIを作成

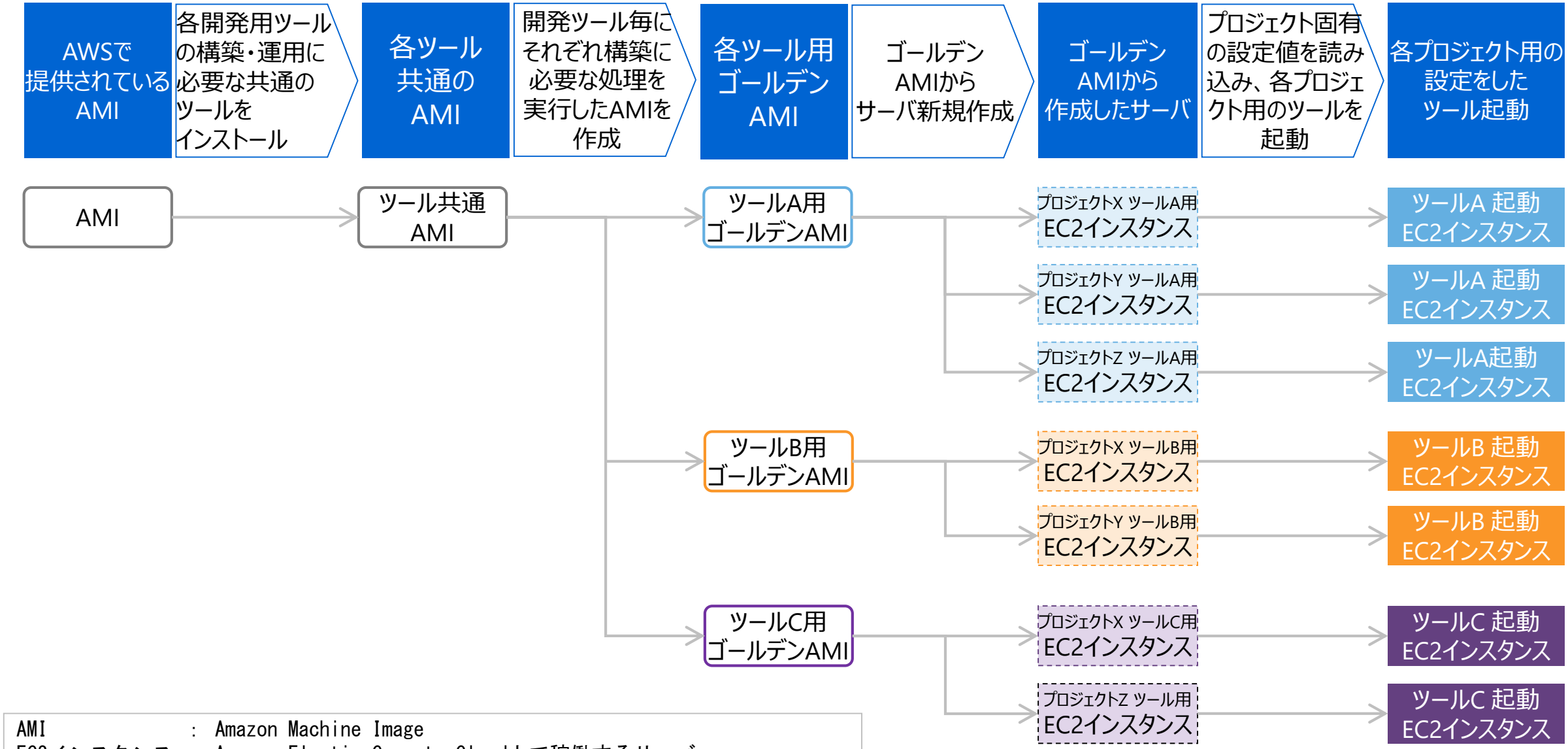
プロジェクト用のツール環境を構築

AWSで提供されているAMI	各開発用ツールの構築・運用に必要な共通のツールをインストール	各ツール共通のAMI	開発ツール毎にそれぞれ構築に必要な処理を実行したAMIを作成	各ツール用ゴールデンAMI	ゴールデンAMIからサーバ新規作成	ゴールデンAMIから作成したサーバ	プロジェクト固有の設定値を読み込み、各プロジェクト用のツールを起動	各プロジェクト用の設定をしたツール起動
ステップ1		ステップ2		ステップ3		ステップ4		
Packer, Ansible		Packer, Ansible		Terraform		Ansible		

環境構築を自動化するためのツール（Ansible, Terraform, Packer）を使用

© 2022 Toshiba Corporation

プロジェクト用ソフトウェア開発ツールの環境立ち上げイメージ



AMI	:	Amazon Machine Image
EC2インスタンス	:	Amazon Elastic Compute Cloud上で稼働するサーバ
ゴールデンAMI	:	必要なツールと構成設定が含まれているAMI

© 2022 Toshiba Corporation

03

環境構築に使用する構成管理ツール (IaCツール)

Ansible・Terraform・Packer

© 2022 Toshiba Corporation

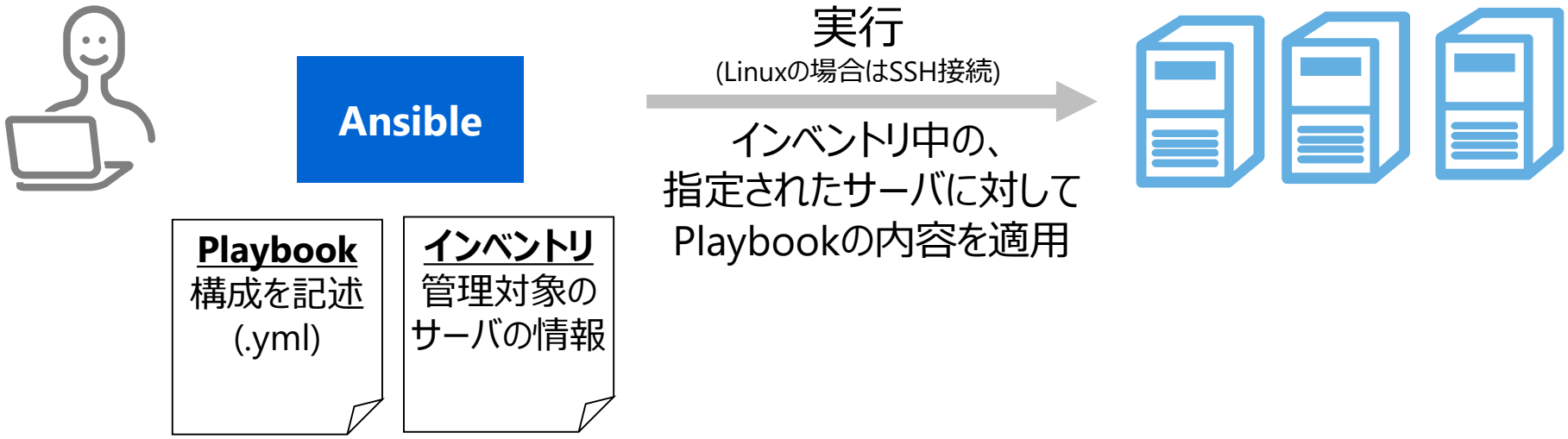
構成管理ツール (Infrastructure as Codeツール)で環境を自動構築するメリット

- 構築時間の短縮
- コード化することによる、インフラ構築手順の明確化と変更管理が可能
- 環境構築の属人性の解消
- 手動による設定ミスの大幅な削減
- 他システムでもコードを再利用可能

© 2022 Toshiba Corporation

Ansible概要 (1)

- Ansibleは、Red Hat 社が開発するオープンソースのインフラ構成管理ツール
- Pythonがインストールされているサーバに対して、ソフトウェアのインストールや設定を自動実行することが可能
- 実行するために準備が必要なファイル
  - Playbook　：構成を宣言 (例：Dockerをインストール、ファイルをコピー、ホスト名を設定…)
  - インベントリ： 管理対象のサーバの情報



© 2022 Toshiba Corporation

Ansible概要 (2)

ソフトウェア開発環境構築でのAnsibleの利用目的

サーバに対して、ソフトウェアのインストールや設定を自動実行

- AMI作成時： サーバへのソフトウェアのインストール、設定(例えば、Proxy設定など)
- 各プロジェクト用のツール起動時： サーバへの設定(ホスト名など)、ツールの設定

補足など

- 運用でも Ansible を活用。1サーバ(VM)につき1ツール(1インスタンス)の構成での運用は、Ansibleなしでは困難。
- 構成をシェルスクリプトで記述するよりもコードの可読性が高い。

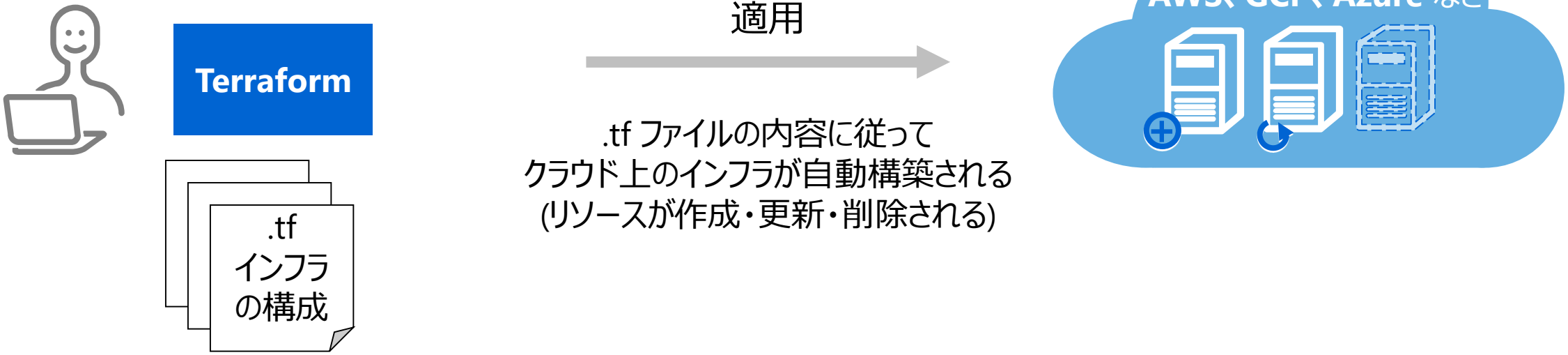
参考URL

- https://docs.ansible.com/
- https://ja.wikipedia.org/wiki/Ansible_(%E3%82%BD%E3%83%95%E3%83%88%E3%82%A6%E3%82%A7%E3%82%A2)

© 2022 Toshiba Corporation

Terraform概要 (1)

- Terraformは、HashiCorp社によって開発されているオープンソースのインフラ構成管理ツール
(クラウドプロビジョニングツール)
- 複数のクラウドプロバイダに対応している (AWS、GCP、Azure、 …)
- インフラの構成を定義し、インフラを自動構築できる



© 2022 Toshiba Corporation

Terraform概要 (2)

ソフトウェア開発環境構築でのTerraformの利用目的

- 使用するAWSの**一部の**リソース管理
  (作成・変更・削除・AWSコンソールによる変更の検出)
  - EC2インスタンス(サーバ)
  - EBS(ボリューム)
  - セキュリティグループ (一部)
  - S3(Amazon Simple Storage Service)バケット (一部)

  AnsibleでもEC2インスタンスを作成することは可能だが(作成していたこともあった)、
  Terraform は リーソースの変更・削除 などの管理が可能。
- リソースの変更履歴の管理

参考URL
- https://www.terraform.io/
- https://en.wikipedia.org/wiki/Terraform_(software)

© 2022 Toshiba Corporation

Packer概要 (1)

- Packerは、HashiCorp社によって開発されている、マシンイメージを作成するためのオープンソースツール
- Packer 実行時の処理内容の概要
  1. マシンイメージを作成するためのサーバ(仮想マシン)を新規作成
  2. 作成したサーバ(仮想マシン)にソフトウェアのインストールや設定を実施
     Ansible・shell などインストール・設定方法を指定可能
  3. マシンイメージ作成・登録
  4. 作成したサーバを削除

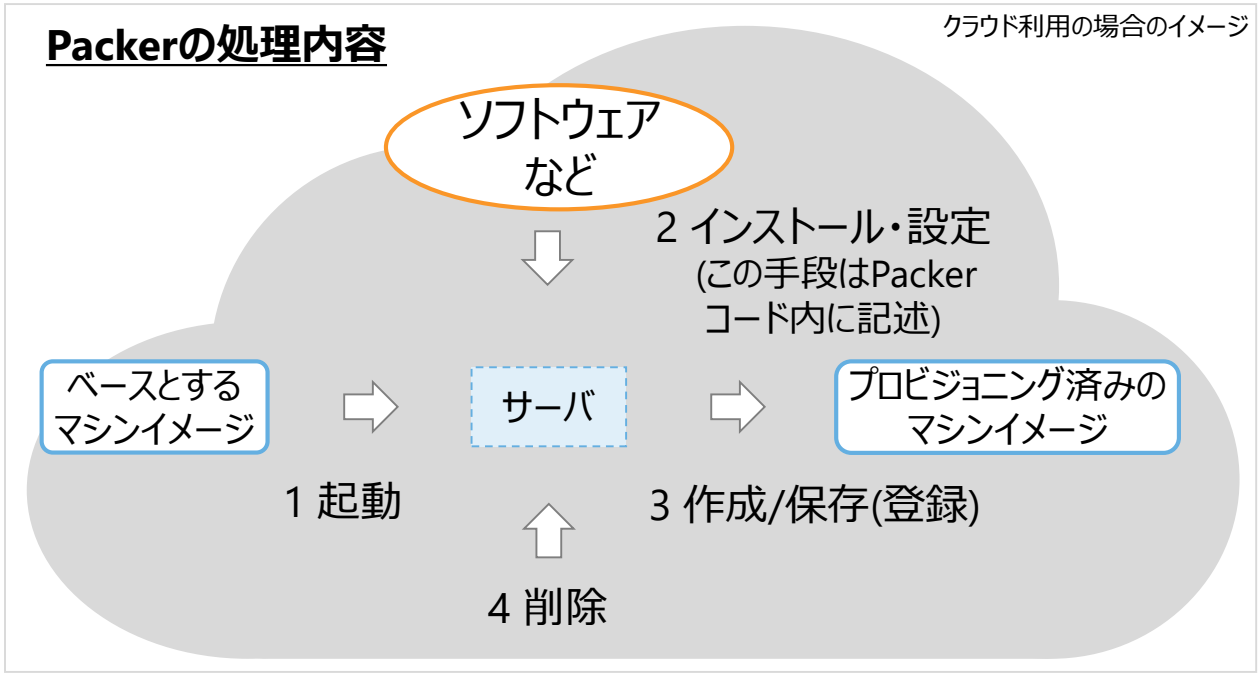
ソフトウェア開発環境構築時のPacker の利用目的
AMIを作成

参考URL
- https://www.packer.io/intro

© 2022 Toshiba Corporation

Packer概要 (2)

Packerの処理内容



ビルド
(例：packer build packer_code_example.json)

Packer

.json

【 packer_code_example.json 】

```
{
  "builders": [{
     "type": "amazon-ebs",
     "region": "ap-northeast-1",
      ...
  }]

  "provisioners": [{
     "type": “ansible",
     “playbook_file : ”common.yml“,
      ...
  }]
}
```

マシンイメージを作成するために起動するサーバ(仮想マシン)の情報
AWSの例。

実行手段と具体的な処理内容を記述。
Ansibleの指定例。
(この場合、マシンイメージを作成するためのサーバへのソフトウェアインストール・設定などの処理はAnsibleで実施)

shellの例

```
"provisioners": [{
   "type": “shell",
   "inline": [
      “sudo apt-get install –y …
       ...
 }]
```

プロビジョニング：必要に応じてネットワークやコンピュータの設備などのリソースを提供できるよう予測し、準備しておくこと

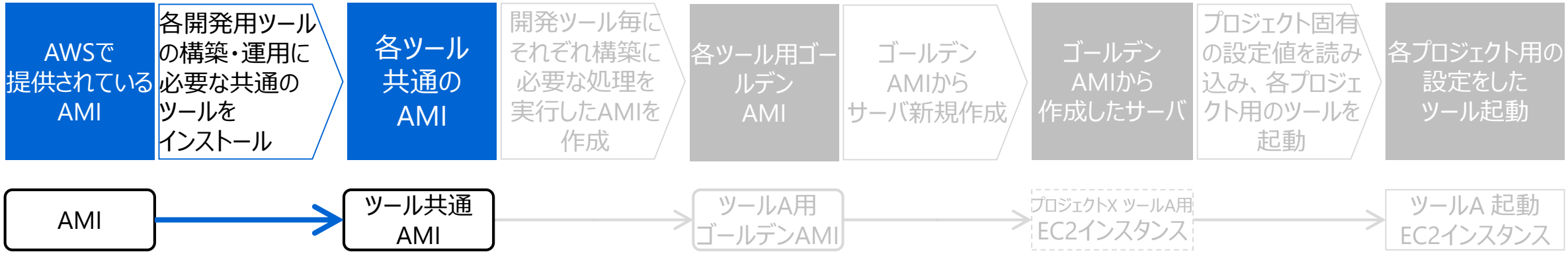
© 2022 Toshiba Corporation

04

ソフトウェア開発環境の構築方法の詳細

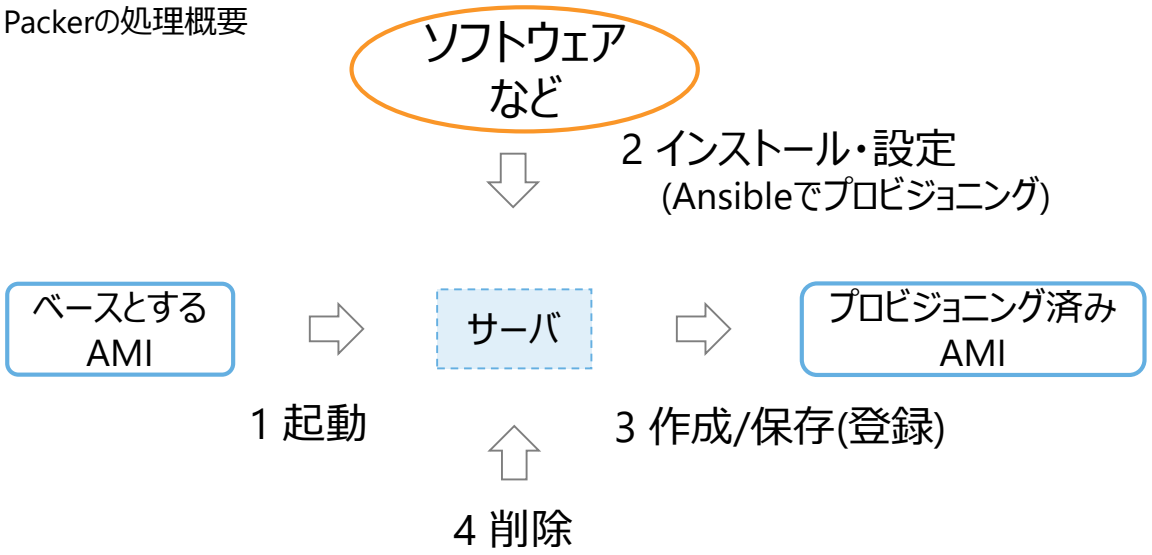
© 2022 Toshiba Corporation

構築方法 (ステップ1) ～Packer・Ansible～



利用ツール： Packer、 Ansible

Packerの処理概要



【 packer_code_example_common.json 】

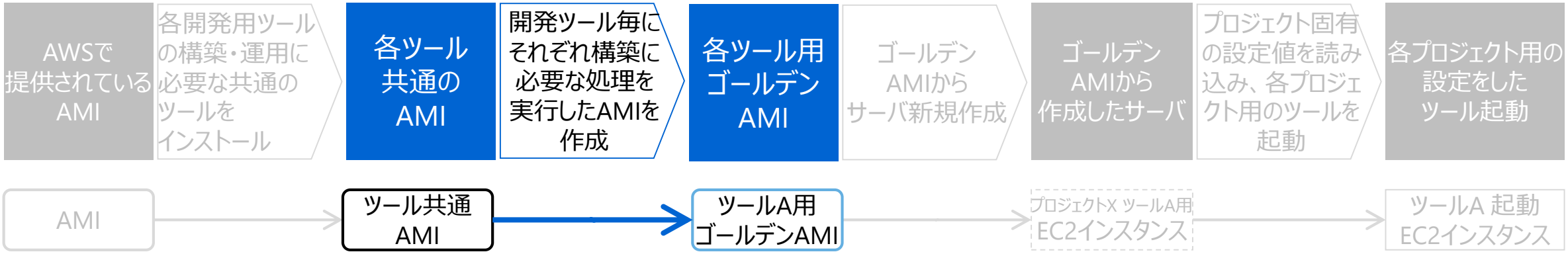
```
{
  "builders": [{
     "type": "amazon-ebs",
     "region": "ap-northeast-1",
      ...
   }]

  "provisioners": [{
     "type": “ansible",
     “playbook_file : ”common.yml“,
      ...
   }]
}
```

AMIを作成するために起動するサーバの情報

Ansibleで、Playbook common.yml を適用。
common.yml：
Docker、AWS CLIなど全てのツールの起動・運用に必要な共通の道具をインストール

© 2022 Toshiba Corporation

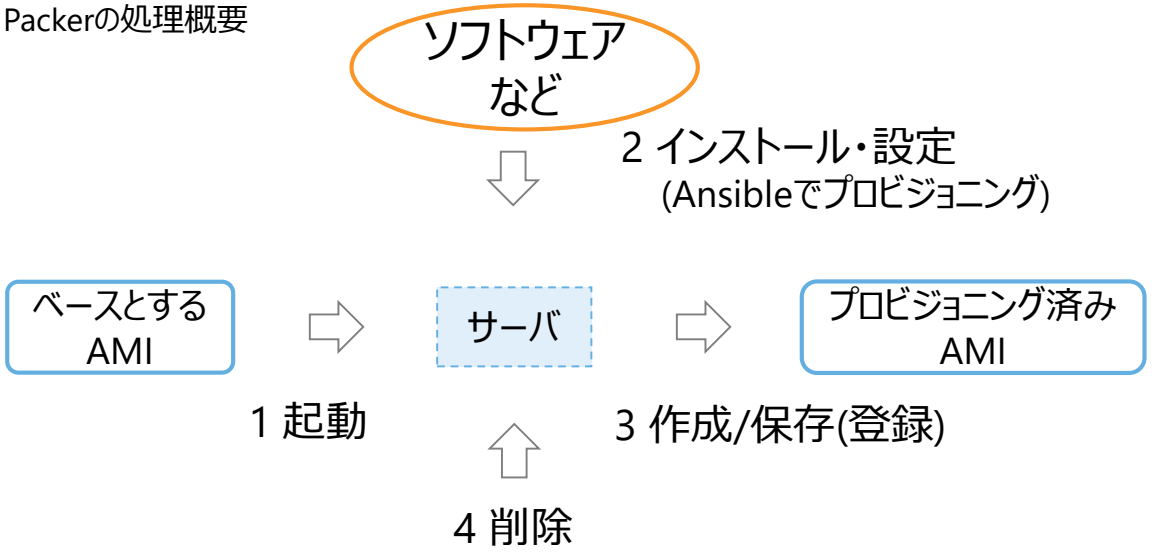
構築方法 (ステップ2) ～Packer・Ansible～



AMIを作成するステップなので、基本的に前スライドと同様。playbook_file で指定するYAMLファイルにそれぞれツール毎に必要な処理を記述。

利用ツール： Packer、 Ansible

Packerの処理概要



【 packer_code_example_gitlab.json 】 GitLabの例

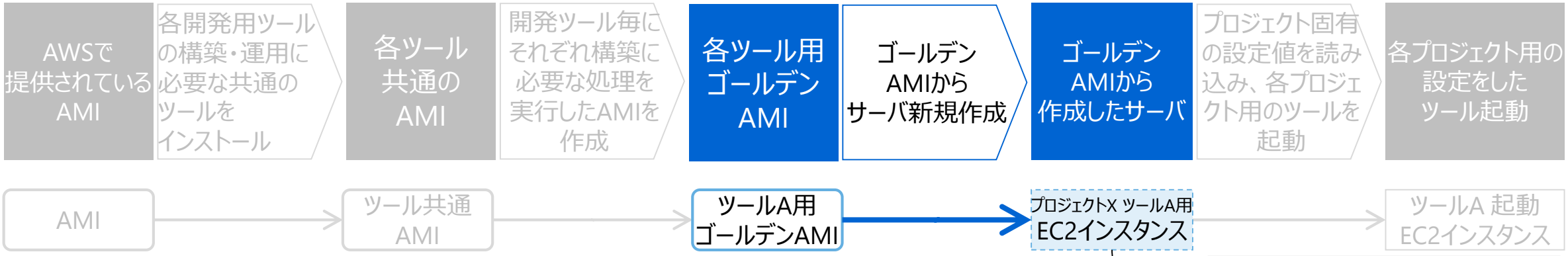
```
{
  "builders": [{
     "type": "amazon-ebs",
     "region": "ap-northeast-1",
      ...
  }]

  "provisioners": [{
     "type": “ansible",
     “playbook_file : ”gitlab.yml“,
      ...
  }]
}
```

AMIを作成するために起動するサーバの情報

Ansibleで、Playbook gitlab.yml を適用。
gitlab.yml：Docker イメージのpullなど、GitLabの起動に必要な処理を実施

© 2022 Toshiba Corporation

構築方法 (ステップ3) ～Terraform～



利用ツール： Terraform

GitLabのEC2インスタンスの例

```
【 X_gitlab_sample.tf 】
module “projectX_gitlab" {
  source = "./gitlab/"
  private_ip = "12.34.56.78"
  tags = {
     Name      = “projectX-gitlab"
     Owner     = “X"
     Ver       = “15.3.5“
     Registry  = “true”
     Pages     = “false”
     Hour      = “12”
     Min       = “10”
   }
...
}
```

GitLab共通のEC2インスタンスに関する情報はモジュールで定義。

プロジェクト毎の設定値をタグ情報として、EC2インスタンスに設定。

程よい範囲でAWSリソースをTerraform管理する
(全てをTerraform管理しない。全運用担当者に影響するリソースはTerraform管理しない)

EC2インスタンスのタグ情報

Name	= “projectX-git;ab"
Owner	= “X"
Ver	= “15.3.4”
Registry	= “true”
Pages	= “false”
Hour	= “12”
Min	= “10”

AWSコンソール画面から確認できるサーバの情報のイメージ

Name	Owner	Ver	Registry	Pages	Hour	Min	...
projectX-gitlab	X	15.3.4	true	false	12	10	...
projectY-gitlab	Y	15.3.4	true	true	12	20	...
projectZ-gitlab	Z	14.7.7	true	true	12	30	...
projectX-jenkins	X	10.3			15	10	...
....	...	...	...	...	...	...	...

タグにツールの設定情報を設定することで、Webブラウザから各サーバの設定情報一覧を確認可能

© 2022 Toshiba Corporation

構築方法 (ステップ3) ～Terraform コードのイメージ～

利用ツール： Terraform

GitLabの例

【 X_gitlab_sample.tf 】

```
module “projectX_gitlab" {
  source = "./gitlab/"
  private_ip = "12.34.56.78"
  tags = {
     Name      = “projectX-gitlab"
     Owner     = “X"
     Ver       = “15.3.5“
     Registry  = “true”
     Pages     = “false”
     Hour      = “12”
     Min       = “10”
   }
...
}
```

Terraform 使用前は、ツール毎に異なる情報をAnsibleの変数で管理

GitLab共通のEC2インスタンスに関する情報はモジュールで定義。

プロジェクト毎の設定値をタグ情報として、EC2インスタンスに設定。

程よい範囲でAWSリソースをTerraform管理する

【./gitlab/main.tf イメージ 】

```
variable "instance_type" {
  default = "t3.large"
}
```

基本的にプロジェクト毎に同じ値を設定するときはdefault値を設定。
異なる値を設定する場合は、X_gitlab_sample.tf で指定。

```
variable "tags" {
  type = map(string)
  default = {
    Name   = ""
    Owner  = ""
    Ver    = "“
    ...
  }
}
.....
```

プロジェクト毎に異なる値を設定するときは、default値なしで、必ずX_gitlab_sample.tf で指定。

```
resource "aws_instance" "gitlab" {
  ami                    = data.aws_ami.base-image.id
  instance_type          = var.instance_type
  private_ip             = var.private_ip
  iam_instance_profile   = “TOSHIBA_Instances"
  subnet_id              = "subnet-01234567"
  key_name               = “toshiba-key"
  availability_zone      = "ap-northeast-1c“
  ....
```

異なる値が設定される項目は変数にする。

共通の設定値は直接指定。
運用中に変数化することも…

© 2022 Toshiba Corporation

構築方法 (ステップ4) ～Ansible～

AWSで提供されているAMI → 各開発用ツールの構築・運用に必要な共通のツールをインストール → 各ツール共通のAMI → 開発ツール毎にそれぞれ構築に必要な処理を実行したAMIを作成 → 各ツール用ゴールデンAMI → ゴールデンAMIからサーバ新規作成 → ゴールデンAMIから作成したサーバ → プロジェクト固有の設定値を読み込み、各プロジェクト用のツールを起動 → 各プロジェクト用の設定をしたツール起動

利用ツール： Ansible 、(AWS CLI)

- AWS CLIを使用して、EC2インスタンスのタグ情報から、これから起動するツールの固有の設定値を取得し、動的にインベントリを作成
- インベントリ情報をもとに、プロジェクト用の設定値をサーバおよびツールに設定し、ツールを起動

プロジェクトX ツールA用 EC2インスタンス → ツールA 起動 EC2インスタンス

EC2インスタンスのタグ情報

Name	= “projectX-git;ab"
Owner	= “X"
Ver	= “15.3.4”
Registry	= “true”
Pages	= “false”
Hour	= “12”
Min	= “10”

AWS CLI でインベントリを作成する。
これにより、タグの値をAnsibleの変数として扱うことができるようになる。

【インベントリイメージ 】

```
projectX-gitlab  Owner=X Ver=15.3.4 Registry=true Pages=false .....
projectY-gitlab  Owner=Y Ver=15.3.4 Registry=true Pages=true .....
projectX-gitlab  Owner=X Ver=14.7.7 Registry=true Paages=true .....
```

Ansibleで起動に必要なファイルを生成/サーバを設定

X用GitLab　EC2インスタンス

設定ファイルに基づいてツールを起動

```
url=http://xxx.toshiba.co.jp/X/gitlab/
registyr=true
pages=false
```

GitLab

```
# cron設定
10 12 * * * maintenance.sh
```

© 2022 Toshiba Corporation

05

おわりに

© 2022 Toshiba Corporation

まとめ

- Ansible、Terraform、Packerを使用して、複数のソフウェア開発環境を効率よく自動構築・管理する方法の一例を紹介

  留意事項

  - 前提が一つ違うと最適解は変わってくる
  - 新しいツールが次々に出てくるので、その時々で最適なものを選択する必要がある

- 開発環境の構築に限らず、設定内容が一部異なる環境を複数構築するケースでも同様の仕組みを適用可能

© 2022 Toshiba Corporation

課題

- バージョンアップ
  - 一度構築した環境のOSのバージョンアップ
  - プラグインを扱うツールのバージョンアップ
- サーバ数・利用者数・データ量の増加による、最適なシステム構成・運用方法の変化への対応
- 新しいサービスや新しいツールの採用
- 利用申し込みから環境構築完了のユーザへの連絡までの完全自動化
- 利用者数増加による問合せ件数の増加

© 2022 Toshiba Corporation

～環境構築する際のその他の留意事項など～

- 最適な構成・運用方法は、ソフトウェア開発環境の規模によって異なる
- 程よい匙加減で自動化する必要がある
  どこまでの範囲を自動化するのか、事象の発生頻度と変更頻度も考慮した上で、吟味が必要
- コードは程よい粒度でモジュール化する必要がある
  分割しすぎて、可読性が低下することがある
- 担当者はゆるやかに流動していくので、基本的にはシンプルで誰でもメンテナンスできるような方法にすることが望ましい
- ツール側もバージョンアップすると、コードの記述方法が変更になることがあるのでバージョンアップするタイミングは要検討

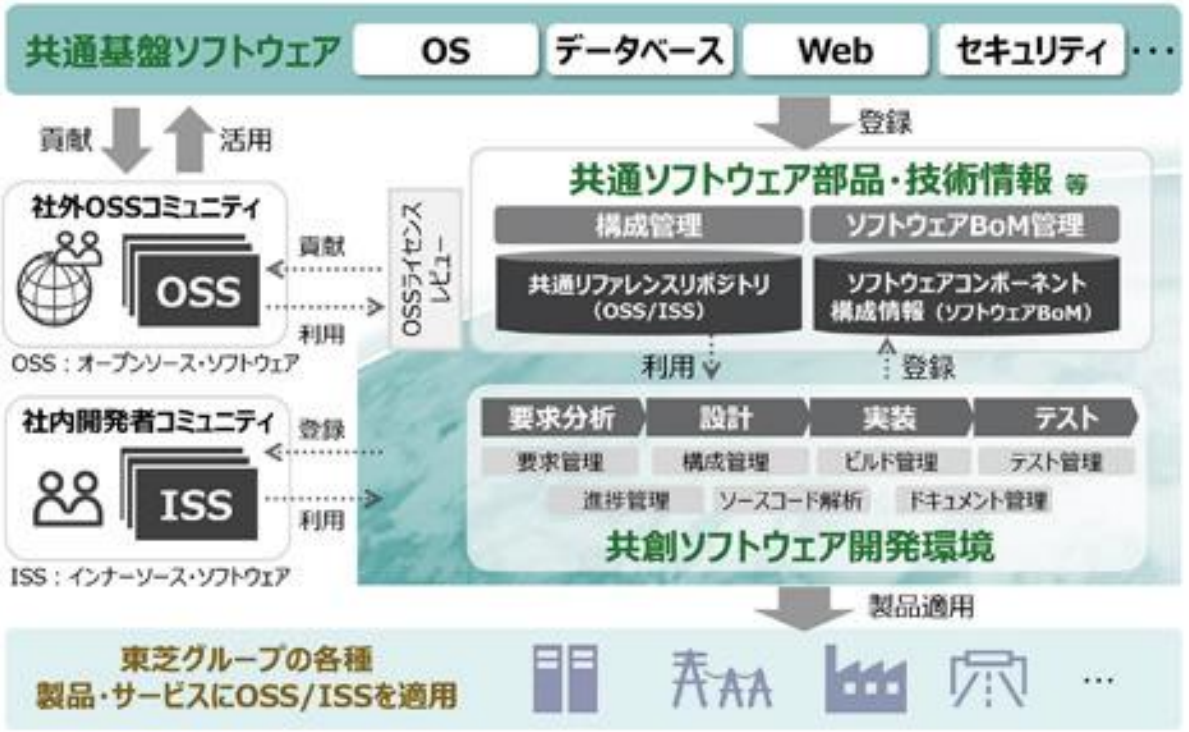
© 2022 Toshiba Corporation

仲間を募集しています！

株式会社 **東芝 ソフトウェア技術センター** では、

- OSSを中心とした共通基盤ソフトウェアに関する技術
- ソフトウェア開発環境、開発プロセス構築に関する技術
- ソフトウェア開発の効率化と信頼性向上に関する技術

に興味のある人を募集しております。



より詳しい情報をWEBで公開していますので是非ご覧ください。
https://www.global.toshiba/jp/technology/corporate/swc.html

© 2022 Toshiba Corporation

TOSHIBA