amazon web services | Webinars

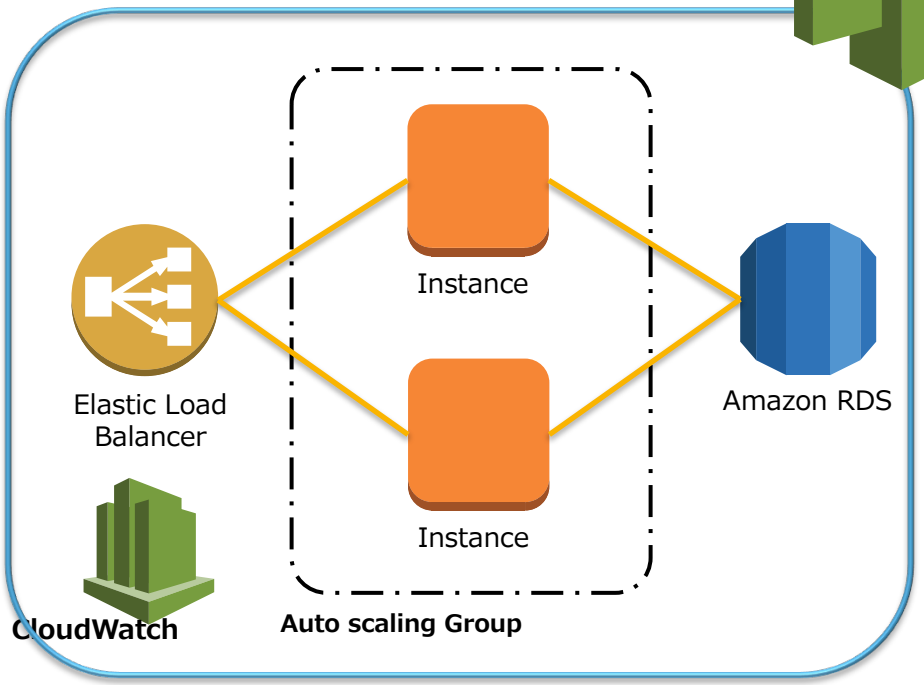
AWS ElasticBeanstalk Docker対応

アマゾン データサービス ジャパン株式会社

AWS ElasticBeanstalk：構築・デプロイの自動化サービス

WAR
ZIP

deploy!




Java


Python


PHP


.NET


Ruby


nodeJS

Docker

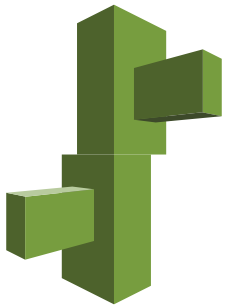
amazon web services

ElasticBeanstalkの論理構成

amazon web services

ElasticBeanstalkのDockerサポート

- ElasticBeanstalkで任意のDockerコンテナを実行可能に
  - Dockerコンテナとして構築できれば任意のDistribution・任意の言語で構築した実行環境をElasticBeanstalkにデプロイ出来る



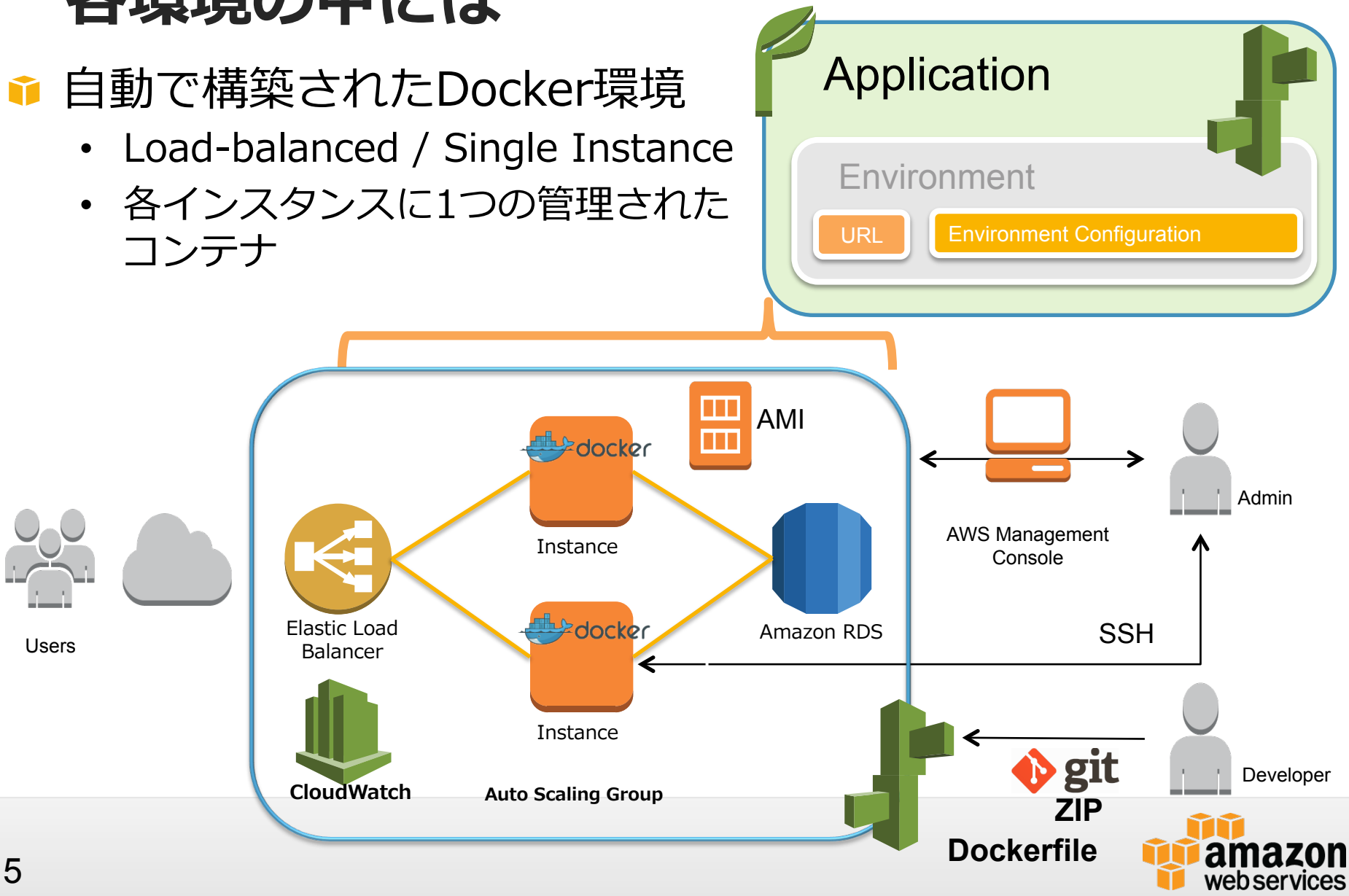
meets

- さらに、下記のような運用が可能
  - ローカルのコンテナ上でコードを開発＆テスト
  - 同じコンテナをElasticBeanstalk上の本番環境にデプロイ

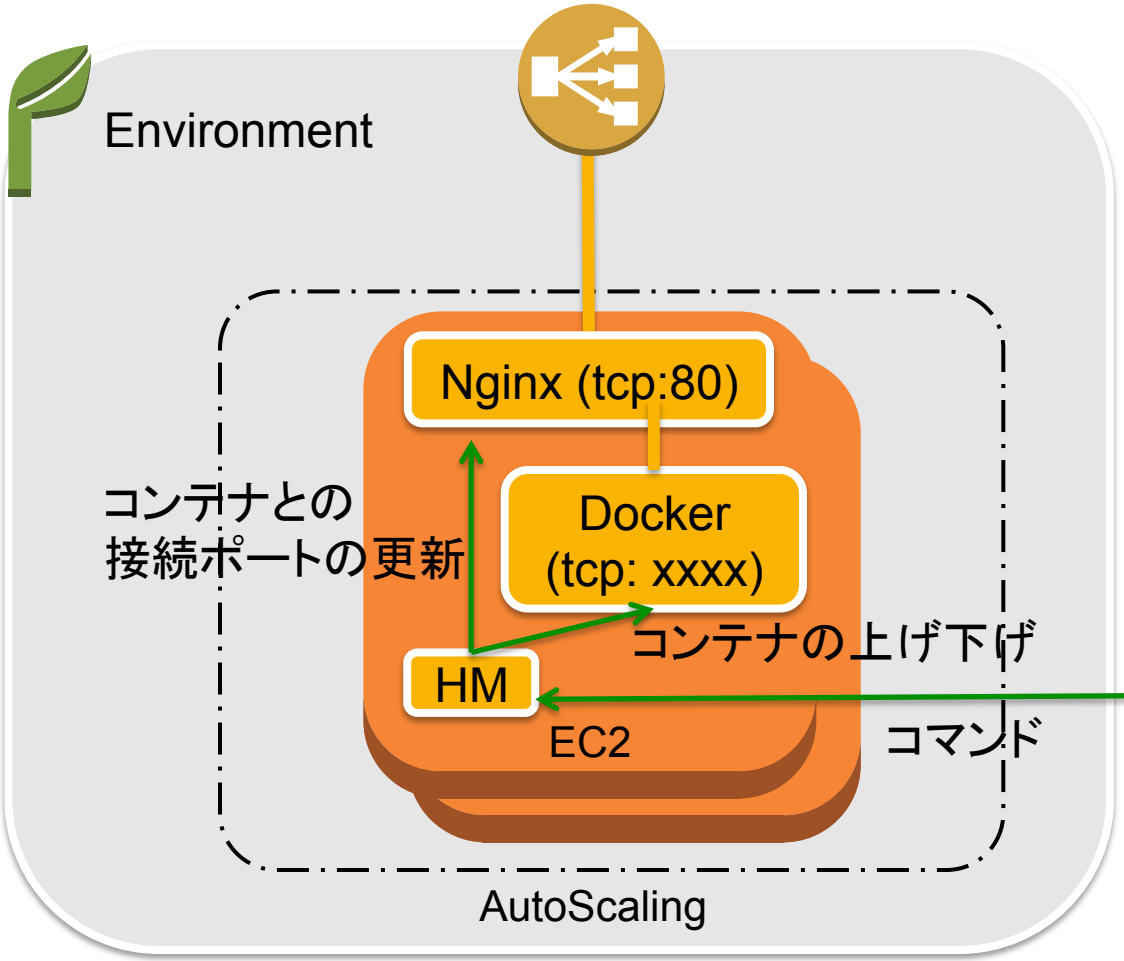
amazon web services

各環境の中には

- 自動で構築されたDocker環境
  - Load-balanced / Single Instance
  - 各インスタンスに1つの管理されたコンテナ



各インスタンスの中には



<u>各インスタンス</u>
・マネージされたコンテナが1つ
・Host Manager (HM)がコマンドを実行

コンテナの設定記述の仕方

- 次の2種類の記述方式を利用可能
  - Dockerfile
    - 一般的なコンテナの構築に使われるDockerfile
    - AWS特有ではないため、どこでも作成・編集・構築が可能
    - コンテナの構築方法を指示するファイルなのでコンテナ起動オプションを指定したい場合は不十分
  - Dockerrun.aws.json
    - ElasticBeanstalk独自の形式
    - 使用するコンテナイメージ、コンテナから参照するボリュームなどを指定
    - コンテナの構築方法自体は指定できない（要Dockerfileと併用）

Dockerfileの例

- いわゆる普通のDockerfile
  - EXPOSEで指定したポートがnginxとの接続に使われる
    注：nginx経由で外部に開けるポートは1つだけ

```
FROM ubuntu:12.04

RUN apt-get update
RUN apt-get install -y nginx zip curl

RUN echo "daemon off;" >> /etc/nginx/nginx.conf
RUN curl -o /usr/share/nginx/www/master.zip -L https://codeload.github.com/gabrielecirulli/2048/zip/master
RUN cd /usr/share/nginx/www/ && unzip master.zip && mv 2048-master/* . && rm -rf 2048-master master.zip

EXPOSE 80

CMD ["/usr/sbin/nginx", "-c", "/etc/nginx/nginx.conf"]
```

- その他注意点
  - FROMとEXPOSEは必須
  - CMDとENTRYPOINTのどちらかは必須

Dockerrun.aws.jsonの例

```
{
  "AWSEBDockerrunVersion": "1",
  "Authentication": {
    "Bucket": "my-bucket",
    "Key": "mydockercfg"
  },
  "Image": {
    "Name": "janedoe/image",
    "Update": "true"
  },
  "Ports": [
    {
      "ContainerPort": "1234"
    }
  ],
  "Volumes": [
    {
      "HostDirectory": "/var/app/mydb",
      "ContainerDirectory": "/etc/mysql"
    }
  ],
  "Logging": "/var/log/nginx"
}
```

プライベートレポジトリを使用する際に必要
S3バケット上に置いた認証情報ファイルを指定

使用すべきイメージ

コンテナがEXPOSEするポートを指定
（ElasticBeanstalkが使用するのはContainerPortのみ）

コンテナにマウントすべき
EC2インスタンスのディレクトリを指定

コンテナ内のログディレクトリを指定

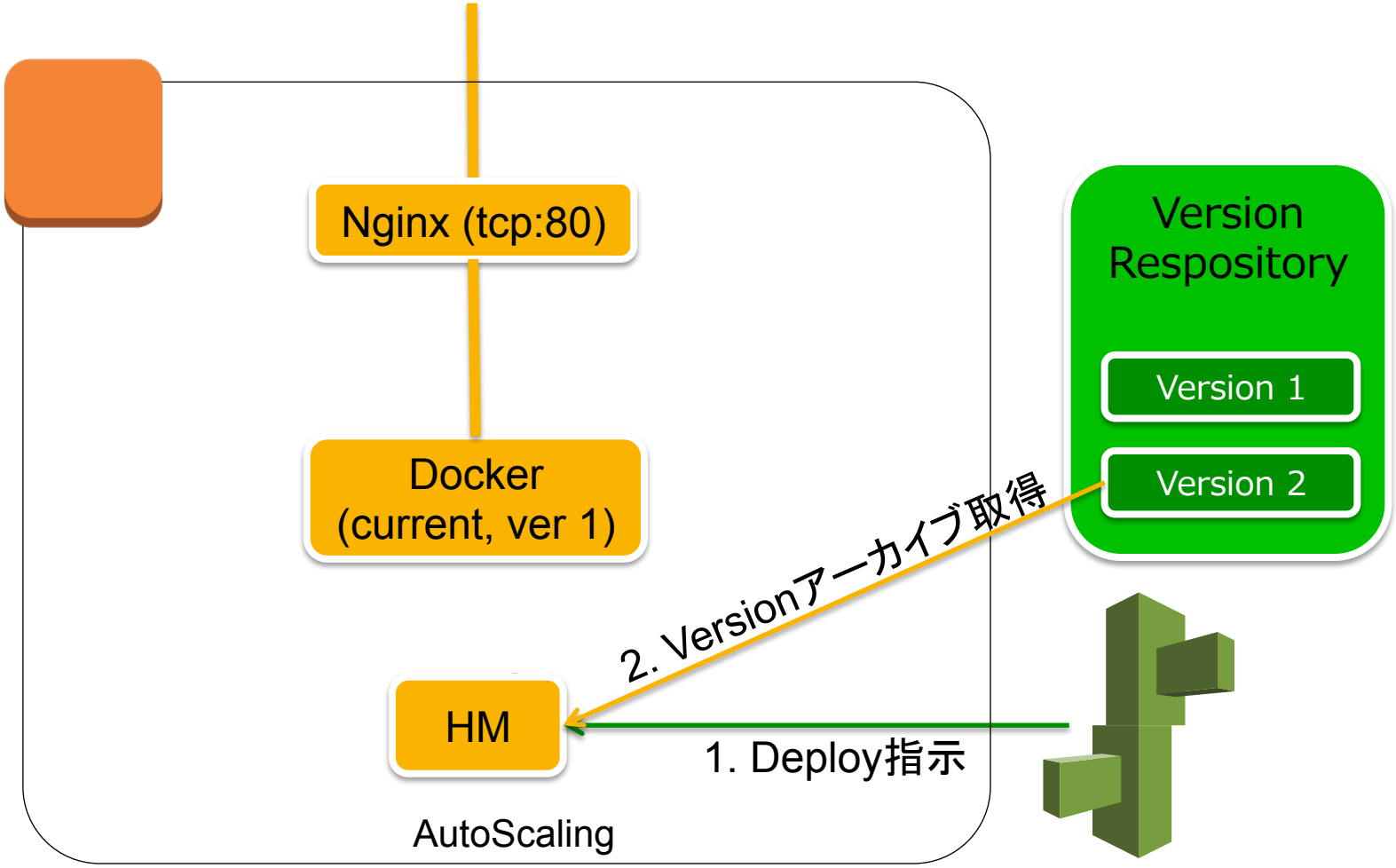
amazon web services

デプロイの仕方

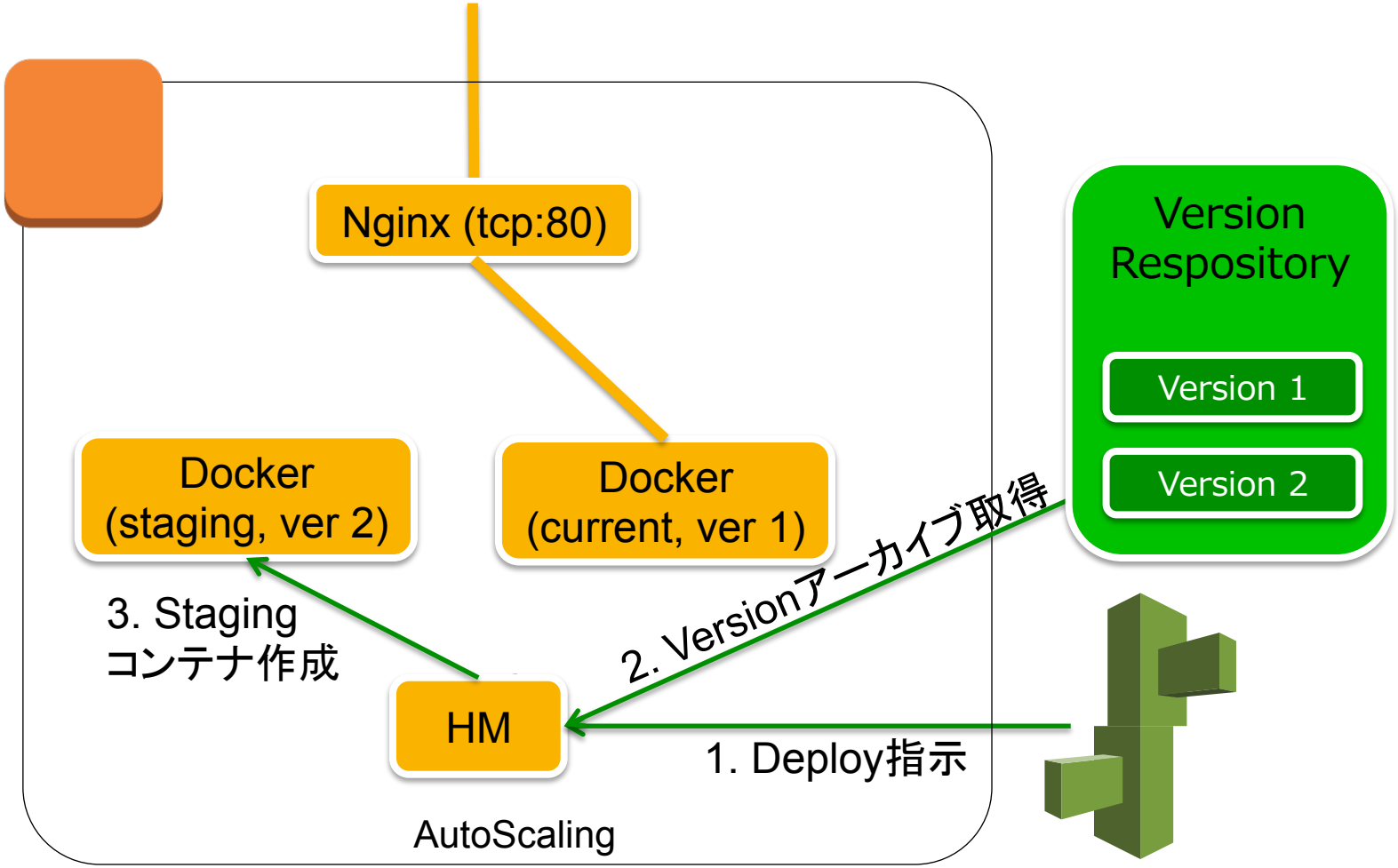
- デプロイ時のファイル形式も2種類
  - 単一ファイル
    - Dockerfile / Dockerrun.aws.jsonのみのデプロイの場合
    - Pros: アーカイブを作る必要がなく簡便
    - Cons: ElasticBeanstalkの環境カスタマイズやリソースファイルのデプロイ等は出来ない
  - ZIPアーカイブ / gitでデプロイ
    - Dockerfile / Dockerrun.aws.jsonあるいは両方を含むアーカイブ
    - Pros: Dockerfileでイメージを構築して起動オプションを指定して起動したり、環境カスタマイズを行ったりすることが可能
    - Cons: アーカイブを作らないといけない

柔軟性とコンテナ記述ファイルのバージョン管理も考慮すると
gitで管理してgit aws.pushでデプロイがおすすめ

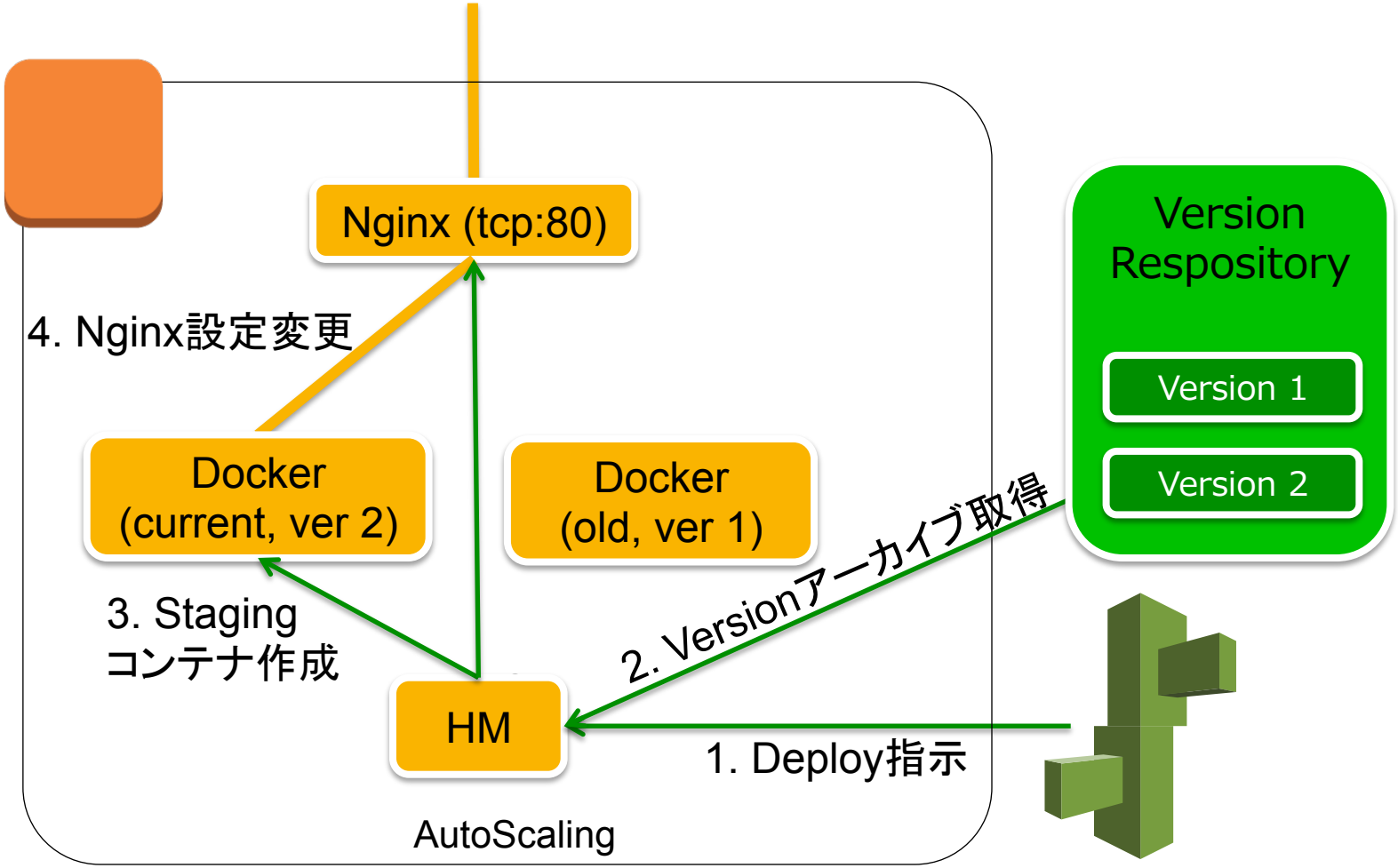
デプロイ時の動作 (1/4)



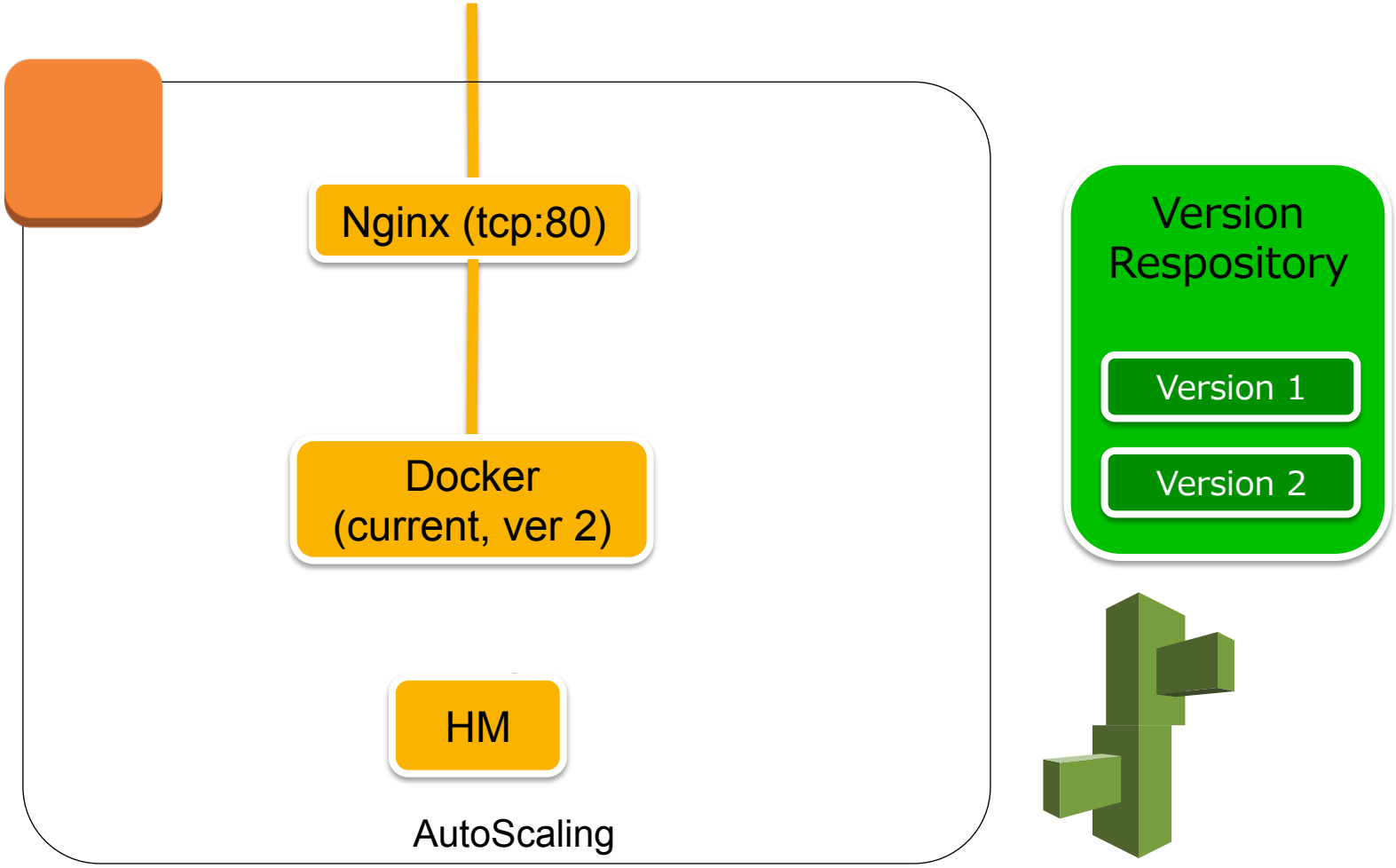
デプロイ時の動作 (2/4)



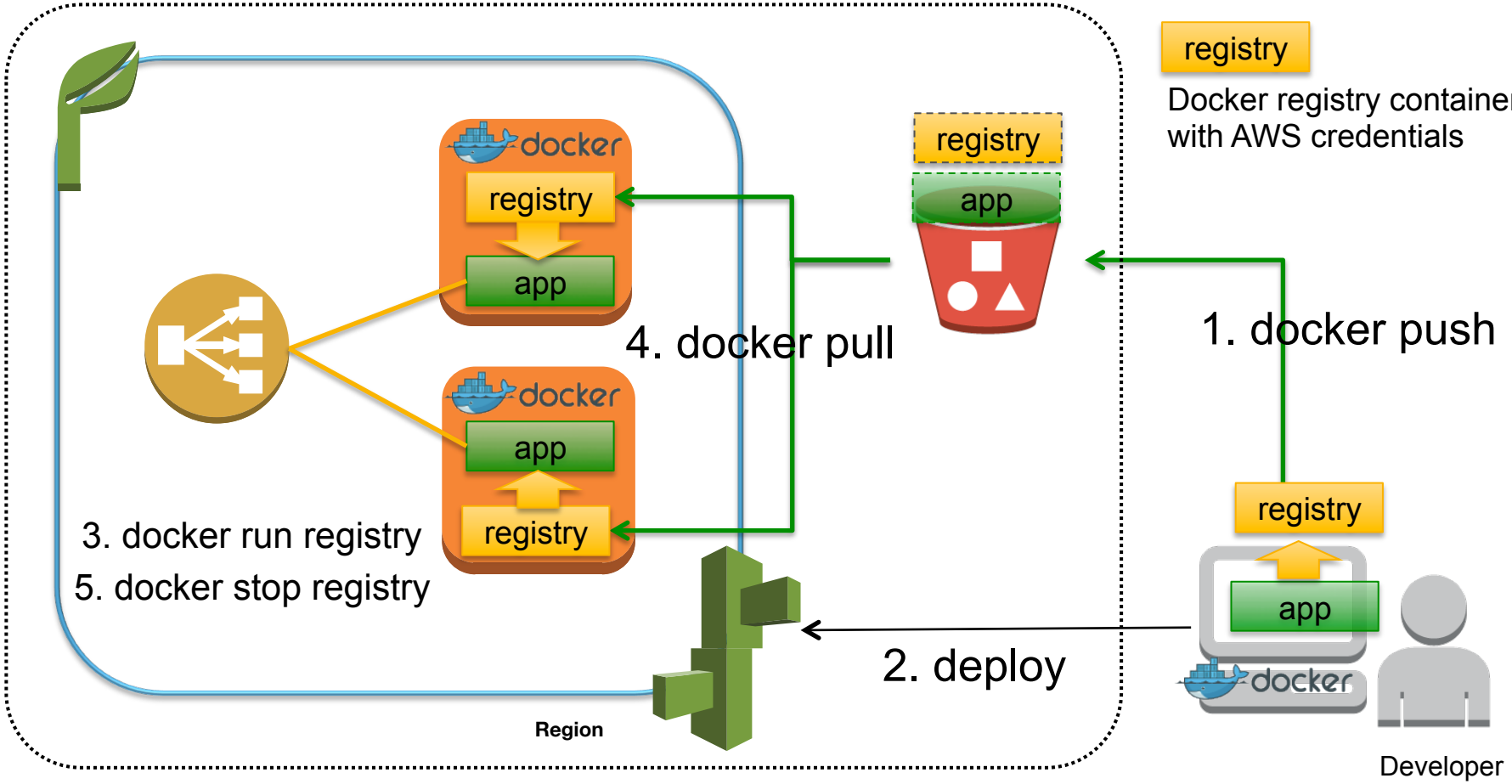
デプロイ時の動作 (3/4)



デプロイ時の動作 (4/4)



S3をプライベートレポジトリとして利用



詳細: http://aws.typepad.com/sajp/2014/06/eb-docker-private-repo.html