Build Custom AMI with Packer

엄기성 / awskrug 판교 소모임 / 2017.02.09

발표자 소개

- 이름: 엄기성(GiSeong Eom)
- 회사: 블루홀(Bluehole)
- 업무: Sysadmin, Infra Operation (2000.10 ~)

Disclaimer

- 이 슬라이드의 내용은 전적으로 발표자 개인 의견입니다.
- 소속 부서, 고용주의 정책/의견과 무관함을 미리 밝혀 둡니다.

목차

Custom AMI 사용 배경

Custom AMI란 무엇인가?

- 공개된 AMI + 독자적인 설정/패키지 설치
- 기본적으로는 생성한 AWS root 계정에서만 사용가능.
- 권한을 변경해서 외부에 공개 가능함.
- 3rd party AMI에 대해서 AWS가 무결성, 보안을 보장하지 않음

Custom AMI의 장점

- 빠른 배포
  - Launch 및 Application 배포 시간을 단축
  - AutoScaling Group에서 특히 중요

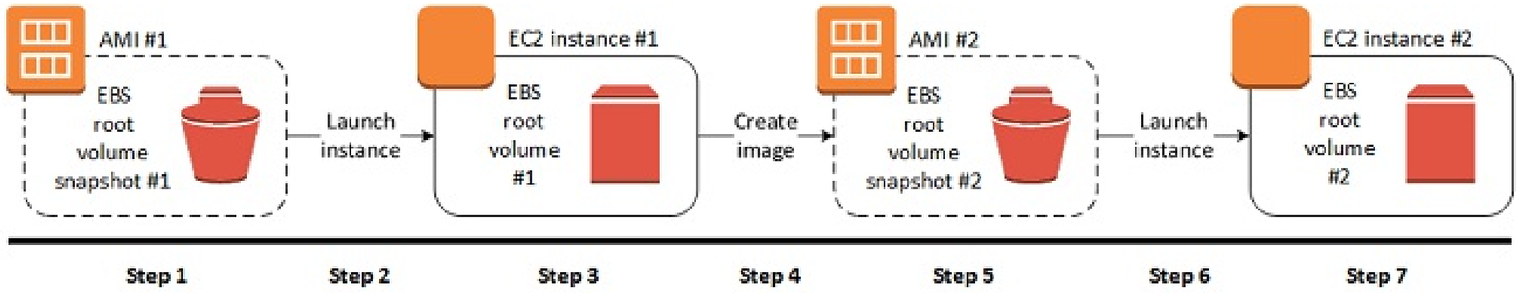
- 일관성 유지
  - 마이너 업그레이드의 경우에도 호환성 문제가 생길 수 있음
  - in-house Application도 호환성 문제 생길 수 있음

Custom AMI의 단점

- 지속적인 버전관리 부담
- 서비스 종류가 많으면, 관리부담이 선형적으로 증가
  - 이미지 관리 전략을 잘 가져가야 함
  - Thin Image vs Thick Image

Custom AMI 생성

Custom AMI 생성 과정



- 단순하게 표현하면
  1. EC2 instance 실행
  2. Customization
  3. Custom AMI 생성
  4. 전 단계의 Image로 EC2 instance 실행
  5. 반복....

AWS CLI 이용한 AMI 빌드

- Amazon 제공 AMI로 EC2 instance 실행

```
aws ec2 run-instances \
  --image-id ami-dac312b4 \
  --count 1 \
  --instance-type t2.nano \
  --key-name vagrant
```

- 새로 실행된 EC2의 public IP주소를 확인, ssh 이용해서 패키지 설치

```
aws ec2 describe-instances \
  --instance-ids i-0e97acd9de5283a9d \
  --query 'Reservations[].Instances[].[PublicIpAddress,KeyName]'

ssh -i ~/.ssh/vagrant ec2-user@52.79.174.90 \
  "sudo yum -y -q install cowsay lolcat"
```

AWS CLI 이용한 AMI 빌드 (cont'd)

- AMI Image 생성

```
aws ec2 create-image \
  --instance-id i-0e97acd9de5283a9d
  --name "exampleAMI"
```

- 방금 생성한 AMI의 Image ID 확인

```
aws ec2 describe-images \
  --owners self
  --query 'Images[].[Name,ImageId,CreationDate]'
```

AWS CLI 이용한 AMI 빌드 (cont'd)

- 확인된 Image ID 이용해서 새로운 EC2 instance 실행

```
aws ec2 run-instances \
  --image-id ami-47944429 \
  --count 1 \
  --instance-type t2.nano \
  --key-name vagrant

ssh -i ~/.ssh/vagrant ec2-user@52.79.186.35 \
  "sudo cowsay \`uname -a\` | lolcat"
```

AWS CLI 이용한 AMI 빌드 (cont'd)

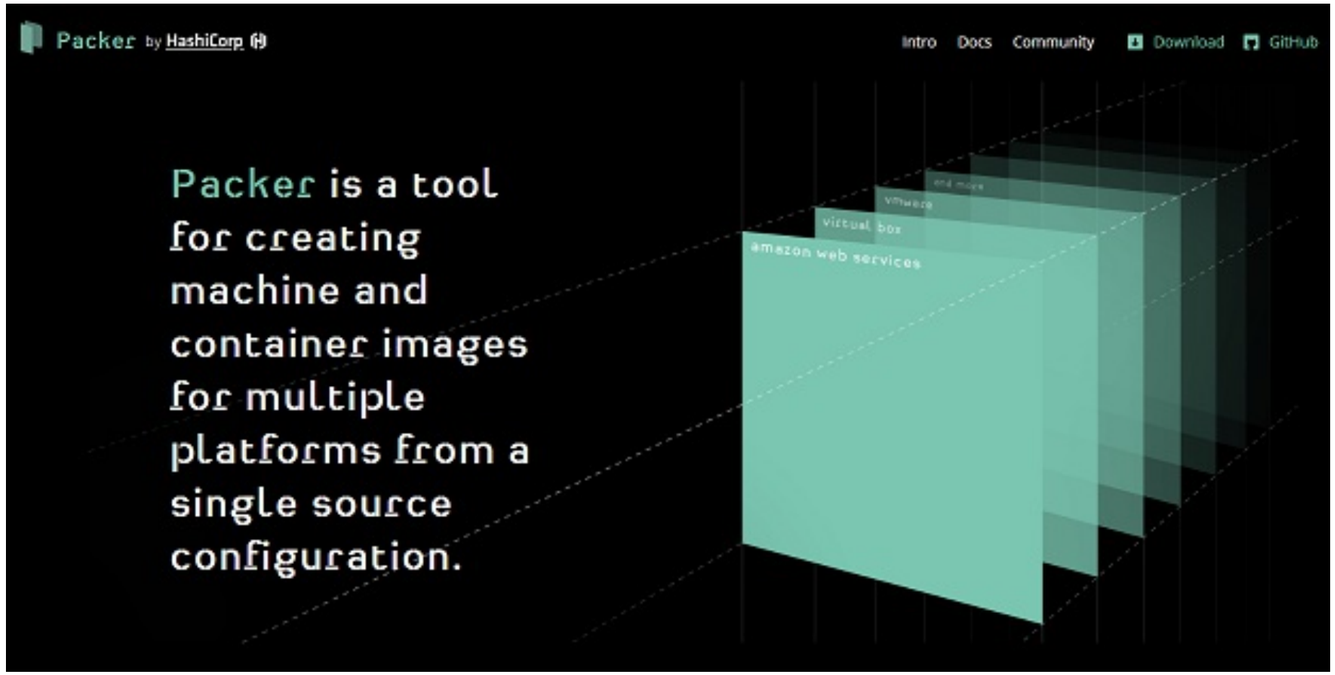
작업 정리하면

1. public AMI 검색
2. EC2 instance 실행
3. EC2 instance 정보 조회
    - AMI customization 작업
4. AMI Image 생성
5. AMI 정보 조회 (Image Id)
6. AMI ID를 이용해서 EC2 instance 실행

다수의 스크립트 작성 **&** 유지보수 부담 - 대안이 필요하다고 판단

Packer 소개

Packer



- https://packer.io
- 최신 버전: **0.12.2**
- Hashicorp에서 개발, 공개한 오픈소스 도구 (go 언어로 개발)
- VM image build에 최적화
- 소스 `Template` = `Builder` + `Provisioner` 로 구성

Packer (cont'd)

Template

- 다양한 Machine 이미지를 단일 소스에서 생성할 수 있다.
- 파일 포맷: `.json`
- 필수 구성요소: **builders**

Packer (cont'd)

Builder

- 특정 플랫폼의 VM Image 생성을 도와주는 Packer 컴포넌트
- Amazon EC2 (AMI), Azure, Google, Hyper-V, Docker 등 18개의 Builder 제공
- Amazon EC2 (AMI)는 가장 처음부터 지원되었던 Builder이고, 안정적임

```
  "builders": [
    {
      "type": "amazon-ebs",
      "access_key": "...",
      "secret_key": "..."
    }
]
```

Packer (cont'd)

Provisioner

- Customization 작업을 도와주는 Pakcer 컴포넌트
- PowerShell, shell script, Chef, Puppet 등을 사용할 수 있다.

```
"provisioners": [
  {
    "type": "windows-shell",
    "scripts": [
        "./scripts/run-sysprep.cmd"
    ]
  }
]
```

AMI build with Packer

example.json

```
{
  "builders": [
    {
      "type": "amazon-ebs",
      "source_ami": "ami-dac312b4",
      "instance_type": "t2.nano",
      "ssh_username": "ec2-user",
      "ami_name": "packer-{{timestamp}}",
      "region": "ap-northeast-2"
    }
  ],

  "provisioners": [
    {
      "type": "shell",
      "inline": [
        "sudo yum -y -q install cowsay lolcat",
        "sudo cowsay `uname -a`"
       ]
    }
  ]
}
```

AMI build with Packer (cont'd)

Packer validate

- example.json 파일의 , 누락된 것을 확인

```
giseong.eom@GISEONG-PC MINGW64 ~/Documents/_devroot/dev/packer/aws
$ packer validate example.json
Failed to parse template: Error parsing JSON: invalid character '"' after object key:value pair
At line 13, column 4 (offset 245):
   12:
   13:   "
         ^


giseong.eom@GISEONG-PC MINGW64 ~/Documents/_devroot/dev/packer/aws
$ packer validate example.json
Template validated successfully.
```

AMI build with Packer (cont'd)

Packer build

```
giseong.eom@GISEONG-PC MINGW64 ~/Documents/_devroot/dev/packer/aws
$ packer build example.json
amazon-ebs output will be in this color.

==> amazon-ebs: Prevalidating AMI Name...
    amazon-ebs: Found Image ID: ami-dac312b4
==> amazon-ebs: Creating temporary keypair: packer_58906036-ca4b-fb5c-4ba0-72f63c84e1e6
==> amazon-ebs: Creating temporary security group for this instance...
==> amazon-ebs: Authorizing access to port 22 the temporary security group...
==> amazon-ebs: Launching a source AWS instance...
    amazon-ebs: Instance ID: i-08c17b2ab5c3a511f
==> amazon-ebs: Waiting for instance (i-08c17b2ab5c3a511f) to become ready...
==> amazon-ebs: Adding tags to source instance
==> amazon-ebs: Waiting for SSH to become available...
==> amazon-ebs: Connected to SSH!
==> amazon-ebs: Provisioning with shell script: T:\Temp\packer-shell329749811
```

AMI build with Packer (cont'd)

Packer build

```
==> amazon-ebs: Provisioning with shell script: T:\Temp\packer-shell329749811
    amazon-ebs:  _____________________________________
    amazon-ebs: / Linux ip-172-31-5-90                \
    amazon-ebs: | 4.4.41-36.55.amzn1.x86_64 #1 SMP Wed  |
    amazon-ebs: | Jan 18 01:03:26 UTC 2017 x86_64 x86_64 |
    amazon-ebs: \ x86_64 GNU/Linux                    /
    amazon-ebs:  -------------------------------------
    amazon-ebs:  \   ^__^
    amazon-ebs:  \  (oo)\_______
    amazon-ebs:     (__)\       )\/\
    amazon-ebs:         ||----w |
    amazon-ebs:         ||     ||
==> amazon-ebs: Stopping the source instance...
==> amazon-ebs: Waiting for the instance to stop...
==> amazon-ebs: Creating the AMI: packer-1485856822
    amazon-ebs: AMI: ami-e922f387
==> amazon-ebs: Waiting for AMI to become ready...
==> amazon-ebs: Terminating the source AWS instance...
==> amazon-ebs: Cleaning up any extra volumes...
==> amazon-ebs: Deleting temporary security group...
==> amazon-ebs: Deleting temporary keypair...
Build 'amazon-ebs' finished.

==> Builds finished. The artifacts of successful builds are:
--> amazon-ebs: AMIs were created:

ap-northeast-2: ami-e922f387
```

- 최종 산출물(build artifact): **ami-e922f387**

AMI build with Packer (cont'd)

주의사항

- .json 파일 편집이 용이한 편집기를 사용할 것
- provision 단계에서 한 번이라도 실패(return code)하면 build 실패

Tip

source_ami

- 예제 소스에서는 `source_ami` 가 하드코딩되어 있다.
- Amazon AMI는 계속 업데이트되므로 AMI ID도 변경된다.

```
{
  "builders": [
    {
      ".... 중복코드 생략 ..."
      "source_ami": "ami-dac312b4",
    }
  ],
```

유지보수 이슈 발생!!!

source_ami (cont'd)

- `source_ami_filter` 를 사용한다.

example2.json

```
  "builders": [
    {
      ".... 중복코드 생략 ..."
      "source_ami_filter": {
        "filters": {
          "virtualization-type": "hvm",
          "architecture": "x86_64",
          "name": "amzn-ami-hvm-*",
          "block-device-mapping.volume-type": "gp2",
          "root-device-type": "ebs"
        },
        "owners": ["amazon"],
        "most_recent": true
      },
    }
  ],
```

source_ami (cont'd)

- build 하면 자동으로 이미지 검색되는 것 확인

```
$ packer build example2.json
amazon-ebs output will be in this color.

==> amazon-ebs: Prevalidating AMI Name...
    amazon-ebs: Found Image ID: ami-dac312b4
==> amazon-ebs: Creating temporary keypair: packer_509bd5c9-54f5-a639-61a2-4271a2436a05
==> amazon-ebs: Creating temporary security group for this instance...
==> amazon-ebs: Authorizing access to port 22 the temporary security group...
==> amazon-ebs: Launching a source AWS instance...
    amazon-ebs: Instance ID: i-02e46c8aaf4583ad4
==> amazon-ebs: Waiting for instance (i-02e46c8aaf4583ad4) to become ready...
==> amazon-ebs: Adding tags to source instance
==> amazon-ebs: Waiting for SSH to become available...
```

variables

- 예제 소스에서 `instance_type`, `ssh_username`, `region` 이 하드코딩.
- Production 환경은 가변적이므로 사용 불가.

```
  "builders": [
    {
      ".... 중복코드 생략 ..."
      "instance_type": "t2.nano",
      "ssh_username": "ec2-user",
      "region": "ap-northeast-2"
    }
  ],
```

유지보수 이슈 발생!!!

variables (cont'd)

- Shell 환경변수, 사용자 변수(variables)를 적극 이용.

example3.json

```
{
  "variables": {
    "aws_region": "{{env `AWS_DEFAULT_REGION`}}"
  },

  "builders": [
    {
      ".... 중복코드 생략 ..."
      "instance_type": "{{user `aws_ec2_type`}}",
      "ssh_username": "{{user `aws_ec2_user`}}",
      "region": "{{user `aws_region`}}"
    }
  ],
}
```

variables (cont'd)

- 사용자 변수에 저장될 내용은 별도 파일로 분리

packer-var.json

```
{
    "aws_ec2_type": "t2.nano",
    "aws_ec2_user": "ec2-user"
}
```

variables (cont'd)

- packer build 도움말
- `-var-file` 옵션을 사용할 수 있다.

```
$ packer build -h
Usage: packer build [options] TEMPLATE

  Will execute multiple builds in parallel as defined in the template.
  The various artifacts created by the template will be outputted.

Options:

  -color=false               Disable color output (on by default)
  -debug                     Debug mode enabled for builds
  -except=foo,bar,baz        Build all builds other than these
  -only=foo,bar,baz          Build only the specified builds
  -force                     Force a build to continue if artifacts exist, deletes existing artifacts
  -machine-readable          Machine-readable output
  -on-error=[cleanup|abort|ask] If the build fails do: clean up (default), abort, or ask
  -parallel=false            Disable parallelization (on by default)
  -var 'key=value'           Variable for templates, can be used multiple times.
  -var-file=path             JSON file containing user variables.
```

- packer validate & build

```
packer validate -var-file=packer-var.json example3.json
packer build    -var-file=packer-var.json example3.json
```

ami_regions

- 예제 소스에서는 단일 region에서만 사용할 것을 전제함.
- 멀티 리젼 환경이 대부분인 요즘 문제가 된다.

```
"builders": [
  {
    ".... 중복코드 생략 ..."
    "region": "{{user `aws_region`}}"
  }
],
```

유지보수 이슈 발생!!!

ami_regions (cont'd)

- ami_regions 설정을 이용해서 빌드 후, 다중 리젼에 복사

example4.json

```
{

  "builders": [
    {
      ".... 중복코드 생략 ..."
      "region": "{{user `aws_region`}}",
      "ami_regions": [
        "ap-northeast-1",
        "ap-northeast-2"
      ]
    }
  ],
}
```

ami_regions (cont'd)

- build 완료된 다음, 이미지 복사 진행되는 것을 확인

```
==> amazon-ebs: Creating the AMI: packer-1486611868
    amazon-ebs: AMI: ami-5c9a4a32
==> amazon-ebs: Waiting for AMI to become ready...
==> amazon-ebs: Copying AMI (ami-5c9a4a32) to other regions...
    amazon-ebs: Copying to: ap-northeast-1
    amazon-ebs: Avoiding copying AMI to duplicate region ap-northeast-2
    amazon-ebs: Waiting for all copies to complete...
==> amazon-ebs: Terminating the source AWS instance...
==> amazon-ebs: Cleaning up any extra volumes...
==> amazon-ebs: No volumes to clean up, skipping
==> amazon-ebs: Deleting temporary security group...
==> amazon-ebs: Deleting temporary keypair...
Build 'amazon-ebs' finished.

==> Builds finished. The artifacts of successful builds are:
--> amazon-ebs: AMIs were created:

ap-northeast-1: ami-90c083f7
ap-northeast-2: ami-5c9a4a32
```

미사용 AMI 삭제

- AMI ID에 연결된 snapshot Id를 먼저 확인

```
aws ec2 describe-images \
 --image-ids ami-5c9a4a32 \
 --query 'Images[].BlockDeviceMappings[].Ebs[].SnapshotId'\
 --output text

snap-0b0f0d168161b67aa
```

- AMI를 de-register

```
aws ec2 deregister-image --image-id ami-5c9a4a32
```

- Snapshot 삭제

```
aws ec2 delete-snapshot --snapshot-id snap-0b0f0d168161b67aa
```

미사용 AMI 삭제 (cont'd)

- AWS CLI 호출하는 bash functions 사용

```
$ aws-ami-delete -l ami-e922f387
-----------------------------
|       DescribeImages      |
+---------------------------+
|  packer-1485856822        |
|  None                     |
|  hvm                      |
|  2017-01-31T10:01:16.000Z |
|  ami-e922f387             |
|  snap-0449dcb725155a67b   |
+---------------------------+

giseong.eom@GISEONG-PC MINGW64 ~/Documents/_devroot/dev/packer/aws
$ aws-ami-delete -d ami-e922f387
Deleting ami-e922f387...done
Deleting snap-0449dcb725155a67b...done
```

Amazon AMI Update Notification

Amazon Linux AMI Notifications (2016.09.1 Release 부터 지원됨)

`arn:aws:sns:us-east-1:137112412989:amazon-linux-ami-updates`

Amazon Windows AMI Notifications

`arn:aws:sns:us-east-1:801119661308:ec2-windows-ami-update`

Any Questions?

발표자료/예제소스 다운로드

References

- 패커(Packer)로 도커(Docker) 이미지 빌드 및 AMI 자동 빌드 시스템 구축
- Packer
  - PACKER
  - PACKER TERMINOLOGY
- Hashicorp
  - DevOps Defined
- AWS Docs
  - Creating an Amazon EBS-Backed Linux AMI
  - Subscribing to Windows AMI Notifications
  - Amazon Linux AMI 2016.09 Release Notes