Custom CentOS AMI builds on AWS

Marko Bevc
About me

- Senior IT Consultant at The Scale Factory (DevOps consultancy and AWS partner)
- IT system engineering background with extensive Linux and virtualization experience
- Open source contributor and supporter
- Certifications and competencies: AWS, CKA and RHEL
- Fan of automation/simplifying things, hiking, cycling and travelling
Today’s agenda

• AMI and how to use it
• Background and motivation
• Building your own AMI
• Our solution and used tools
• Demo
• Conclusions
AMI and its lifecycle

- Machine image – formats: AMIs for EC2, VMDK/VMX files for VMware, OVF exports for VirtualBox, etc.
- AMI (Amazon Machine Image) – template for the root volume of the instance (EBS snapshot).
- Create own AMI – using existing one as base or create one.
- Sharing it with others or disseminate using market space ecosystem.
- De-registering AMI and clean-up.
Motivation for custom AMI

- System hardening
- Latest OS patches
- Configuration management bootstrapping
- Quicker instance spin-ups
- Other customisations, etc.
- ReadyScale platform
Ecosystem and tools

- CentOS/RHEL as runtime and build OS
- HashiCorp Packer for building and customising
- Kitchen Test suite with GOSS tests
  - Native provisioner: https://github.com/YaleUniversity/packer-provisioner-goss
- `aws-cli` for managing temporary SSH keys
- Code in GitHub/GitLab
- Pipeline triggered on push events
- Slack notifications
- Automated AMI publishing
- Secrets and encrypted storage
Building custom AMIs

- A machine image – single static unit that contains a pre-configured operating system and installed software for VMs
- Based on existing AMIs
- EC2 image builder (OS limitations)
- VM import (two phases: local build + import)
- Build our own from scratch in AWS
• Started with Jenkins
• Migrated to GitHub Actions (GitLab CI/CD)
• Containers as build images
• YAML configuration manifests
• Visibility and good feedback loop to repository
• Reproducible artefacts
.gitlab-ci.yml:
--------------
image: "centos:latest"

variables:
  PACKER_VER: '1.5.1'

before_script:
  - dnf install -y make unzip jq gcc redhat-rpm-config ruby ruby-devel rubygem-bundler python3-pip
  - pip3 install awscli
  - curl https://releases.hashicorp.com/packer/${PACKER_VER}/packer_${PACKER_VER}_linux_amd64.zip -o - > /tmp/packer.zip
  - unzip -d /usr/bin /tmp/packer.zip

stages:
  - test
  - build

check:
  except:
    refs:
      - master
    stage: test
    script:
      - make -v
      - aws --version
      - packer -v
      - aws s3 ls

create:
  only:
    refs:
      - master
    stage: build
    script:
      - export AWS_SG_ID=sg-fffffffffffffffff
      - make build
      - 'export AWS_DEFAULT_REGION=$(cat manifest.json | jq -r .builds[0].artifact_id | cut -d: -f1)'
      - 'export AWS_AMI_ID=$(cat manifest.json | jq -r .builds[0].artifact_id | cut -d: -f2)'
      - make test
• Hashicorp Packer is a good option (open, mature, extensible and portable)
• Open-source tool for creating identical machine images for multiple platforms from a single source configuration
• Features and attributes:
  – Integration with CD/CD system and appliance creation
  – Wide range of builders/plugins, parallel builds
  – Testability and manifest validation
  – Structured configuration
  – Simple, lightweight and performant
• Alternatives are more opinionated and have less open/pluggable architecture (LinuxKit, Boxfuse, ...)
build.json 1/2:
-------------
{
  "builders": [
    {
      "ami_name": "foundationami_centos8_hvm_ami_{{ user `BuildTime` }}",
      "ami_virtualization_type": "hvm",
      "ena_support": true,
      "instance_type": "t2.micro",
      "sriov_support": true,
      "ssh_username": "ec2-user",
      "type": "amazon-ebssurrogate",
      "source_ami_filter": {
        "filters": {
          "name": "RHEL-8.1",
          "architecture": "x86_64",
          "virtualization-type": "hvm",
          "root-device-type": "ebs"
        },
        "owners": [ "309956199498" ],
        "most_recent": true
      },
      "launch_block_device_mappings": [
        {
          "volume_type": "gp2",
          "device_name": "{{ user `DeviceName` }}",
          "delete_on_termination": true,
          "volume_size": 20
        }
      ],
      "ami_root_device": {
        "delete_on_termination": true,
        "device_name": "/dev/xvda",
        "source_device_name": "{{ user `DeviceName` }}",
        "volume_size": 20,
        "volume_type": "gp2"
      }
    }
  ]
}
{
    "run_tags": {
        "Name": "PackerBuilder C8 {{ user `BuildTime` }}"
    },
    "ami_regions": ["{{ user `build_region` }}"],
    "ami_groups": ["all"],
    "snapshot_groups": ["all"]
},

"provisioners": [
    {
        "inline": "mkdir {{ user `BuildDirectory` }}",
        "type": "shell"
    },
    {
        "destination": "{{ user `BuildDirectory` }}",
        "source": "files",
        "type": "file"
    },
    {
        "destination": "{{ user `BuildDirectory` }}/build.sh",
        "source": "build.sh",
        "type": "file"
    },
    {
        "inline": "sudo {{ user `BuildDirectory` }}/build.sh {{ user `DeviceName` }} {{ user `BuildDirectory` }}",
        "type": "shell"
    }
],

"post-processors": [
    {
        "type": "manifest",
        "output": "manifest.json",
        "strip_path": true
    }
]
build.sh:

---------

RELEASE="http://mirror.centos.org/centos/8/BaseOS/x86_64/os/Packages/centos-release-8.1-1.1911.0.8.el8.x86_64.rpm"

# Create arrays of packages to manage
mapfile UNNECESSARY < "${BUILD_DIRECTORY}/files/packages_remove"
mapfile REQUIRED < "${BUILD_DIRECTORY}/files/packages_install"

# Partition EBS volume
parted -a optimal "${DEVICE}" < "${BUILD_DIRECTORY}/files/parted.conf"
partprobe
 # Show Partitions
parted -l "${DEVICE}"
 # Format main partition as XFS
mkfs.xfs -L root "${DEVICE}3"
mkdir -p "${ROOTFS}"
mount "${DEVICE}3" "${ROOTFS}"
 # Format swap
mkswap "${DEVICE}2"

### Basic CentOS Install
rpm --root=${ROOTFS} --initdb
rpm --root="${ROOTFS}" --nodeps "${RELEASE}"
rm -rf /etc/yum.repos.d/*
dnf install -y http://mirror.centos.org/centos/8/BaseOS/x86_64/os/ Packages/centos-repos-8.1-1.1911.0.8.el8.noarch.rpm
dnf install -y http://mirror.centos.org/centos/8/BaseOS/x86_64/os/ Packages/centos-repos-8.1-1.1911.0.8.el8.x86_64.rpm
# Install base system
dnf --installroot=${ROOTFS} --nogpgcheck -y groupinstall core
dnf --installroot=${ROOTFS} install -y http://mirror.centos.org/centos/8/BaseOS/x86_64/os/ Packages/centos-repos-8.1-1.1911.0.8.el8.x86_64.rpm

# Add required deps
# Yum S3 iam repo
cp "${BUILD_DIRECTORY}/files/repos/yum-plugin-s3-iam.repo" "${ROOTFS}/etc/yum.repos.d/yum-plugin-s3-iam.repo"

dnf --installroot=${ROOTFS} --nogpgcheck -y install "$[REQUIRED[@]/$'\n'/]"
# Remove unnecessary packages
dnf --installroot=${ROOTFS} -C -y remove "$[UNNECESSARY[@]/$'\n'/]" --setopt="clean_requirements_on_remove=1"

...
Testing AMIs using GOSS

- Opensource & written in Golang
- YAML based serverspec alternative
- Test generation from current state
- Fast and ‘small’ binary footprint
- Wide community support and contributions
- Testing limitations: only Linux + related package/services
file:
  /tmp:
    exists: true
    mode: "1777"
    owner: root
    group: root
    filetype: directory
    contains: []
  /tmp/bootstrapped:
    exists: true
    contains: ["**bootstrapped**"]
  /usr/bin/aws:
    exists: true
    mode: "0755"
    owner: root
    group: root
    filetype: file
    contains: []
user:
  ec2-user:
    exists: true
    uid: 1000
    gid: 1000
    groups:
      - ec2-user
    home: /home/ec2-user
process:
  sshd:
    running: true
port:
  tcp:22:
    listening: true
    ip:
      - '0.0.0.0'
---
package:
  centos-release:
    installed: true
    versions:
      - '8.1'
addr:
  tcp://google.com:22:
    reachable: false
    timeout: 1000
  tcp://google.com:443:
    reachable: true
    timeout: 1000
command:
  pip3 -V:
    exit-status: 0
    stdout:
      - 'pip 9.0.3 from /usr/lib/python3.6/site-packages (python 3.6)'
    stderr: []
    timeout: 10000
  aws --version:
    exit-status: 0
    stdout:
      - 'aws-cli/1.17.6'
    stderr: []
    timeout: 10000
mount:
  /dev:
    exists: true
    opts:
      - rw
      - nosuid
    source: devtmpfs
    filesystem: devtmpfs
Time for DEMO!
Conclusions & takeaways

- Simple, quick, portable and reproducible workflow
- Automation, flexibility and high deployment cadence
- Wide support for various AWS builders (amazon-ebs\ instance|chroot|ebssurrogate)
- Machine readable/generated outputs – pipeline stages and testing
- Kitchen Testing EC2 driver sometimes breaks
- Native provisioner (no aws-cli and keys generation)
- Missing CentOS 8.x base AMI*
"Make things as simple as possible, but not simpler.” A. Einstein

- Twitter: @_MarkoB
- GitHub / GitLab: @mbevc1
- LinkedIn: https://www.linkedin.com/in/marko-bevc-245271118/
- Resources:
  - https://github.com/aelsabbahy/goss
  - https://github.com/ahelal/kitchen-goss