lf-common-packer

Release v0.14.1.dev0+09cb3bc

Linux Foundation Releng

Jul 04, 2023

CONTENTS

A collection of common packer scripts and baseline used by Linux Foundation projects as a central way to maintain and manage common VM configuration.

CHAPTER

ONE

RELEASE NOTES

1.1 v0.14.1

1.1.1 Known Issues

- RSA/SHA-1 was deprecated in the latest OpenSSH release 8.8 causing builds fail with the following error:

  Data could not be sent to remote host “127.0.0.1”. Make sure this host can be reached over ssh: command-line: line 0: Bad configuration option: pubkeyacceptedalgorithms

1.1.2 Bug Fixes

- Add workaroud and pass required HostKeyAlgorithms through ssh extra arguemements.

1.2 v0.14.0

1.2.1 Prelude

As of packer version 1.7.0 HCL2 is the preferred way to write Packer templates. HCL2 preserves existing workflows while leveraging HCL2’s advanced features like variable interpolation and configuration composability.

1.2.2 New Features

- Modify devstack templates for Ubuntu/Debian distributions.

1.2.3 Known Issues

- Add support for host key algorithms to work with local ssh proxy without which packer builds results in “failed to handshake” error. Workaround is to pass additional params with “extra_arguments”.

  Reference: https://github.com/hashicorp/packer-plugin-ansible/issues/140

1.2.4 Upgrade Notes

- Migrate packer templates from JSON to HCL2 format. JSON format templates are deprecated and no longer works with packer version > 1.9.x.

  Existing JSON templates can be converted to '.pkr.hcl' using:

  ```
  packer hcl2_upgrade -with-anotations <folder|filename>
  ```

  Packer version 1.9.1 will be minimum required version for packer jobs. This version requires installing the cloud specific plugin through packer config and needs to be initalize and download before running *packer build*.

  <temmplate>.pkr.hcl includes the sources and builds are defined. *.auto.pkrvars.hcl* includes variables that are loaded automatically. These variables load automatically from the same directory and are common across templates. *variables.pkr.hcl* includes variable declarations that are common across templates.

  Reference: https://developer.hashicorp.com/packer/guides/hcl/variables https://developer.hashicorp.com/packer/docs/templates/hcl_templates https://github.com/hashicorp/packer-plugin-openstack/blob/main/README.md

1.2.5 Deprecation Notes

- Support for '.json' templates will be removed from common-packer in subsequent release to give enough time for projects consuming to upgrade. All projects specific templates not available in this repository are required to convert existing '.json' to '.pkr.hcl' format.

1.3 v0.13.0

1.3.1 New Features

- Add Openstack devstack templates and provisioner to common-packer.
- Add GHA packer validation job to common-packer.

1.4 v0.12.1

1.4.1 Known Issues

- Error running "./common-packer/ansible-playbook.sh –version": exit status 1

1.4.2 Bug Fixes

- Skip ansible provisioners version check. Packer provisioners invokes the ansible version check, which can be skipped since the provisioner is invoking a custom script.

1.5 v0.12.0

1.5.1 Prelude

Install ansible and ansible-playbook using PyPI.

1.5.2 Known Issues

- **Error:**
  Problem: package ansible-5.4.0-3.el8.noarch requires (ansible-core >= 2.12.2 with ansible-core < 2.13), but none of the providers can be installed

1.5.3 Bug Fixes

- Install ansible through system packages causes dependencies conflicts on CentOS 8 platform therefore update packer templates to use ansible provisioner created through venv.

1.6 v0.11.0

1.6.1 Upgrade Notes

- Upgrade git v2.36 on CentOS 7.

1.7 v0.10.2

1.7.1 New Features

- Add support for CentOS Stream 9

1.7.2 Bug Fixes

- Import correct GPG keys for sigul and EL8

  Sigul 1.1.1 is signed by the Fedora infrastructure GPG key therefore import the Fedora infra key before installing the updated version on Sigul for CentOS8.

  Import the EL8 GPG keys without which would fail while installing several dependencies.

1.8 v0.10.0

1.8.1 New Features

- Add support for CentOS Streams 8

1.9 v0.9.2

1.9.1 Bug Fixes

- Add community.general as part of the required collections. Required for tasks defined in local-docker specific to Magma to enable and manage virtualization tools.

1.10 v0.9.1

1.10.1 Bug Fixes

- Using ansible_facts does not match/return the minor versions of the Repoid. For CentOS 8.2.2004 and earlier versions uses repoid as 'PowerTools' while CentOS 8.3.2011 and later versions uses repoid as 'powertools'. To handle this, check the repo file name under /etc/yum.repos.d/ and enable the correct repository.

  https://wiki.centos.org/Manuals/ReleaseNotes/CentOS8.2011#Yum_repo_file_and_repoid_changes

1.11 v0.9.0

1.11.1 New Features

- Feat: Add support for Docker builder on arm64

  The original change e24c07369afd514abdf3efb0f596f772261412ed missed updating arm64 var files, while the builder templates were updated. This breaks the packer verify jobs.

  Error:
  required variable not set: docker_source_image'

- Add support for Docker builder on Windows

  The original change e24c07369afd514abdf3efb0f596f772261412ed missed updating Windows var files, while the builder templates were updated. This breaks the packer verify jobs.

  Error:

  required variable not set: docker_source_image'

1.11.2 Bug Fixes

- All vars files have been updated to properly include the AMI product code filter. This is needed to properly pass global-jjb verification.

1.12 v0.8.0

1.12.1 New Features

- Docker image builds are now supported by the packer templates.

1.12.2 Upgrade Notes

- Requires first upgrading global-jjb to version v0.57.0 to pull in support for selecting a packer-builder in the packer-merge jobs, otherwise existing project packer-merge job builds *may* fail without the global-jjb updates if builds are run on a Jenkins node that does not support Docker.
- Projects using AWS must ensure that the packer-merge jobs are updated to set *packer-builder* to aws.

  Example:

```
- project:
    name: packer-builder-jobs
    jobs:
      - gerrit-packer-merge

    project: releng/builder
    project-name: builder
    branch: master
    archive-artifacts: "**/*.log"

    build-node: centos7-builder-2c-1g
    build-timeout: 90
    cron: "00 H 1 * *"

    platforms:
      - centos-7
      - centos-8

    packer-builder: aws
    templates: builder
    update-cloud-image: true
```

1.13 v0.7.6

1.13.1 Upgrade Notes

- Upgrade lf-standard-* flavors to v3

  v3 flavors guarantees jobs spin on newer hardware that are faster and cost-efficient than the v2 flavors.

1.14 v0.7.1

1.14.1 Known Issues

- Remove availability zone from Openstack templates. This was causing errors, and is unnecessary (there is only one AZ for these images).

1.15 v0.6.1

1.15.1 Bug Fixes

- Updated Ubuntu 18.04 image as the previous base image was uploaded in qcow2 format. This caused timeouts in packer builds. New image is in raw format.

  Replaces: "LF - Ubuntu 18.04 LTS (2020-07-28)"

  With: "LF - Ubuntu 18.04 LTS (2019-12-11)"

1.16 v0.6.0

1.16.1 New Features

- CentOS 7 builds will now include the Sigul client by default. This package is now in use by most projects, and is being downloaded whenever it is needed. This has particularly been a problem due to frequent connection issues with the kojipkgs servers that host the sigul package.

1.16.2 Upgrade Notes

- The CentOS 7 2003 base image is uploaded on the cloud provider. Switch the packer var files to build from the latest CentOS 7.6 2003 base image.

  https://cloud.centos.org/centos/7/images/CentOS-7-x86_64-GenericCloud-2003.raw.tar.gz

1.17 v0.5.2

1.17.1 Bug Fixes

- Fix failure to install Git 2 from the IUS repo. While the patch https://gerrit.linuxfoundation.org/infra/c/releng/common-packer/+/62244 sets up the IUS repo, it fails to replace git with git from the IUS repo due to the selected package git2u being non-existant in the repo. This update instead installs git224 from the IUS repo.

1.18 v0.5.0

1.18.1 Upgrade Notes

- Add CentOS 8.x base image.

1.19 v0.4.2

1.19.1 Bug Fixes

- Use netselect to choose a package mirror to install python-minimal in a reliable manner.

  apt{-get} does not refresh the package mirrors (for packer builds run within Jenkins), therefore fails with "E: Unable to locate package python-minimal" while installing python-minimal.

1.20 v0.3.1

1.20.1 Bug Fixes

- The EC2 (aws) template had an extra configuration option that was added but had not been properly tested. This option is removed allowing aws based templates to properly validate and build.
- Pygments release 2.4.0 which added a python requires that excludes all versions of Python < 3.5. The LFCI default 3 version is 3.4 so causes build failure.
- Request-2.22.0 does not work with python-3.4.9, so pin requests to v2.21.0 to address the tox failures.
- Ansible supports lists passed to package install that can avoid using with_items. Using with_items makes multiple calls to the packages manager slowing down the performance.

1.21 v0.3.0

1.21.1 New Features

- More control over Openstack images is now allowed. This includes selection of cloud, availability zone, if volumes should be created and at what size and format.

1.21.2 Upgrade Notes

- Packer 1.3.2 is now required to support Openstack block storage and disk format requirements
- The CentOS 7.6 1811 base image is uploaded on the cloud provider. Switch the packer var files to build from the latest CentOS 7.6 1811 base image.

1.22 v0.1.0

1.22.1 Upgrade Notes

- Requires Global JJB v0.26.0 minimum.

CHAPTER TWO

REQUIREMENTS

- Ansible 2.9.27 or later
- Packer 1.9.1 or later

Install Ansible via pip in a virtualenv to build images.

```
virtualenv -p $(which python3) ~/venv/.ansible
source ~/venv/.ansible/bin/activate
pip3 install ansible~=2.9.27
```

CHAPTER

THREE

INSTALL COMMON PACKER

Deploy common-packer in the ci-management repository's packer directory as a submodule. Installing, upgrading, and rolling back changes is simple via the versioned git tags.

1. Choose a common packer version to install

```
COMMON_PACKER_VERSION=v0.1.0
```

2. Clone common-packer into ci-management repo

```
cd packer/
git submodule add https://github.com/lfit/releng-common-packer common-packer

# Checkout the version of common-packer you wish to deploy
cd common-packer
git checkout $COMMON_PACKER_VERSION
```

3. Commit common-packer version to the ci-managment repo

```
cd ../..
git add packer/common-packer
git commit -sm "Install common-packer $COMMON_PACKER_VERSION"
```

4. Push the patch to ci-management for review

```
git review
```

CHAPTER

FOUR

COMMON PACKER USAGE

To use any provisioning script available from the common-packer repository, the calling template must appropriately reference the full path to the script. In most cases this is 'provision/$SCRIPT' which is now 'common-packer/provision/$SCRIPT'

To use any of the provided templates, the template should have a symlink into the calling project's templates directory. This is because our common-packer job scripts operate on the templates available in this directory. Any template, will also look for local customization out of the local repository's provisioning directory via local-$TEMPLATE.yaml playbook.

Distribution specific vars are now provided in 'common-packer/vars/$DISTRO'. Path to them as normal and they will already contain the correct strings. For a new project make sure the base_image name is available in the cloud system.

4.1 Setup packer template

This setups up a builder image for use in a project. Repeat for any other templates provided by common-packer as necessary.

```
# Instructions assume the working directory is the ci-management repo root
cd packer
mkdir provision templates
ln -rs common-packer/templates/builder.pkr.hcl templates/builder.pkr.hcl
cp common-packer/provision/local-builder.yaml provision/local-builder.yaml
```

4.2 Example template design and run

In most cases the 'builder' template unmodified is all that the project should need to run their code builds. If a project has a custom package that they must build into a custom builder type then design the new template with the following parameters:

1. Execute the common-packer/provision/install-python.sh script
2. Execute the common-packer/provision/baseline.yaml Ansible playbook
3. Execute a local playbook
4. Execute the system reseal Ansible role

Steps 2-4 are actually all contained inside of the local playbook. The following examples for `docker template` and `provisioning script` show how they import the existing baseline playbook into the local playbook to reduce duplication in code.

Example docker template:

```
{
  "variables": {
    "ansible_roles_path": ".galaxy",
    "arch": "x86_64",
    "base_image": null,
    "cloud_network": null,
    "cloud_user_data": null,
    "cloud_region": "ca-ymq-1",
    "vm_use_block_storage": "true",
    "vm_volume_size": "20",
    "vm_image_disk_format": "",
    "distro": null,
    "docker_source_image": null,
    "flavor": "v2-highcpu-1",
    "ssh_user": null,
    "ssh_proxy_host": ""
  },
  "builders": [
    {
      "name": "openstack",
      "image_name": "ZZCI - {{user `distro`}} - docker - {{user `arch`}} - {{isotime \
↪"20060102-150405.000\"}}",
      "instance_name": "{{user `distro`}}-docker-{{uuid}}",
      "source_image_name": "{{user `base_image`}}",
      "type": "openstack",
      "region": "{{user `cloud_region`}}",
      "networks": ["{{user `cloud_network`}}"],
      "user_data_file": "{{user `cloud_user_data`}}",
      "ssh_username": "{{user `ssh_user`}}",
      "ssh_proxy_host": "{{user `ssh_proxy_host`}}",
      "flavor": "{{user `flavor`}}",
      "metadata": {
        "ci_managed": "yes"
      },
      "use_blockstorage_volume": "{{user `vm_use_block_storage`}}",
      "volume_size": "{{user `vm_volume_size`}}",
      "image_disk_format": "{{user `vm_image_disk_format`}}"
    },
    {
      "name": "docker",
      "type": "docker",
      "image": "{{ user `docker_source_image` }}",
      "commit": true,
      "changes": ["ENTRYPOINT [\"\"]", "CMD [\"\"]"]
    }
  ],
  "provisioners": [
    {
      "type": "shell",
      "scripts": ["common-packer/provision/install-python.sh"],
      "execute_command": "chmod +x {{ .Path }}; if [ \"$UID\" == \"0\" ]; then {{ .Vars }
```

(continues on next page)

(continued from previous page)

```
↪} '{{ .Path }}'; else {{ .Vars }} sudo -E '{{ .Path }}'; fi"
    },
    {
      "type": "shell-local",
      "command": "./common-packer/ansible-galaxy.sh {{user `ansible_roles_path`}}"
    },
    {
      "type": "ansible",
      "command": "./common-packer/ansible-playbook.sh",
      "skip_version_check": true,
      "playbook_file": "provision/local-docker.yaml",
      "ansible_env_vars": [
        "ANSIBLE_NOCOWS=1",
        "ANSIBLE_PIPELINING=True",
        "ANSIBLE_ROLES_PATH={{user `ansible_roles_path`}}",
        "ANSIBLE_CALLBACK_WHITELIST=profile_tasks",
        "ANSIBLE_STDOUT_CALLBACK=debug"
      ]
    }
  ]
}
```

Example provisioning script:

```
---
- import_playbook: baseline.yaml

- hosts: all
  become_user: root
  become_method: sudo

  pre_tasks:
    - include_role: name=lfit.system-update

  roles:
    - { role: lfit.docker-install, mtu: 1458 }

  post_tasks:
    - name: System Reseal
      script: system-reseal.sh
      become: true
```

4.3 Install Roles from Ansible Galaxy

Common-packer contains a script *ansible-galaxy.sh* which runs *ansible-galaxy install -r requirements.yaml* from the common-packer repo to install common-packer role dependencies. In the local ci-management/packer directory a project can provide it's own requirements.yaml to pull in roles before running a Packer build.

4.4 Local testing of common-packer

For developers of common-packer who would like to be able to locally test from the common-packer repo, the common-packer repository already contains a symlink to itself which allows one to test the templates in the common-packer templates standalone.

CHAPTER

FIVE

LF NETWORK

Note: This doc is relevant to LF staff who have access to the VPN network.

To bootstrap an image inside the LF network the packer configuration contains an `ssh_proxy_host` variable to connect to a SOCKS5 proxy through a known system such as Jenkins to connect to the VM instance on the network.

5.1 Connect through the LF network

1. Connect to the VPN
2. Set `ssh_proxy_host` to **127.0.0.1** in vars/cloud-env.json
3. Create a SOCKS 5 proxy on port 1080

   ```
   ssh -fND 1080 vex-yul-odl-jenkins-2.ci.codeaurora.org
   ```

   Replace the server `vex-yul-odl-jenkins-2.ci.codeaurora.org` with a known server for the relevant project.
4. Run packer as usual

Note: If forwarding your ssh-agent and you have more than one ssh key, you may see this error message:

```
Failed to connect to the host via ssh: Warning: Permanently
  added '[127.0.0.1]:44502' (RSA) to the list of known hosts.
Received disconnect from 127.0.0.1 port 44502:2: too many
  authentication failures
packet_write_wait: Connection to 127.0.0.1 port 44502: Broken pipe
muxclient: master hello exchange failed
Failed to connect to new control master
```

To resolve this start the ssh SOCKS proxy with your agent first, stop your agent, then run packer.

Bonus

If you would like to be able to ssh directly to a dynamic system inside of the LF Network add the following to `~/.ssh/config`:

```
Host 10.30.18*
  ProxyCommand ssh vex-yul-acumos-jenkins-2.ci.codeaurora.org nc %h 22
  StrictHostKeyChecking no
  UserKnownHostsFile /dev/null
```

Replace the server `vex-yul-acumos-jenkins-2.ci.codeaurora.org` with a known server for the relevant project.

CHAPTER

SIX

UPLOAD LINUX CLOUD IMAGE FOR PACKER JOBS

The following instructions provide details on how to update a base image in OpenStack cloud.

1. Ask an administrator for the required ~/.config/openstack/ files 1. Install prerequisites

```
pip install lftools[openstack]
yum/apt install qemu-img
```

2. Fetch the image file in .img format
3. Vexxhost requires images to be uploaded in RAW format so QCOW2 images need to be converted to RAW:

```
qemu-img convert -f qcow2 -O raw bionic-server-cloudimg-amd64.img bionic-server-
↪cloudimg-amd64-raw.img
```

3. upload it to the cloud in question

```
lftools openstack --os-cloud "cloudname" image upload --disk-format raw local-
↪server-cloudimg-arm64-raw.img "Ubuntu 18.04 LTS [2020-08-04]"
```

4. Update common-packer to use this new image.

eg edit: vars/ubuntu-18.04.json once that is merged tag common packer with the new version

```
git tag -s v0.6.2 -m  "common-packer v0.6.2 release"
git push origin v0.6.2
```

5. pull common packer changes into global-jjb of your project

```
cd packer/common-packer/
git checkout git checkout v0.6.2
```

6. once your change is merged re-run one of the packer merge jobs it will use the new base image.

CHAPTER

SEVEN

UPDATE PACKER IMAGES TO REFLECT CHANGES IN LF ANSIBLE GALAXY ROLES

Workflow:

Change is merged to an ansible role: https://gerrit.linuxfoundation.org/infra/c/ansible/roles/lf-recommended-tools/+/16671

Now we want the images in our "umbrella-project's" openstack cloud to have these changes:

Find the packer merge jobs in our umbrella project's jenkins. Trigger this job for each builder you want to be able to update.

https://jenkins.umbrella-name.org/search/?q=packer-merge

> ci-management-packer-merge-centos-7-docker ci-management-packer-merge-centos-7-builder ci-management-packer-merge-ubuntu-18.04-docker ci-management-packer-merge-ubuntu-18.04-builder

Trigger a merge job for each builder that we want to update. https://jenkins.umbrella-name.org/view/ci-management/job/ci-management-packer-merge-centos-7-builder/

Or if you dont have trigger:

you can run a remerge via comment on a change (anyone can do this) to for example: umbrella-project/ci-management/packer/vars/centos-7.json

example: https://gerrit.onap.org/r/c/ci-management/+/89661/1/packer/vars/centos-7.json

and that will trigger both builds: ci-management-packer-merge-centos-7-docker ci-management-packer-merge-centos-7-builder

When the job is complete, you will see some info in the Build history

Which will look like this:

```
Image: ZZCI - CentOS 7 - builder - x86_64 - 20190910-180457.538
```

Or there is an openstack command for admins.

```
openstack --os-cloud="odlci" image list
```

Take this information and update a file in your ci-managment repo umbrella-project/ci-management/jenkins-config/clouds/openstack/UMBRELLA-PROJECT-VEX/ for example: centos7-builder-2c-1g.cfg

```
IMAGE_NAME=ZZCI - CentOS 7 - builder - 20181115-0246
LABELS=centos7-basebuild-4c-4g
HARDWARE_ID=v3-standard-4
```

In this case you would also want to update centos7-builder-4c-4g.cfg

Replace the IMAGE_NAME with the string retreived from the ci-management-packer-merge-centos-7-builder job. Once that is merged the new image will be used when that builder is spawned.

Some info on the jenkins side of this about the ci-management-packer-merge jobs: https://docs.releng.linuxfoundation.org/projects/global-jjb/en/latest/jjb/lf-ci-jobs.html?highlight=jenkins-config#jenkins-configuration-verify

CHAPTER

EIGHT

INDICES AND TABLES

- genindex
- modindex
- search