OpenTofu Day
EUROPE

# Packaging OpenTofu in pkgsrc

Leonardo Taccari

<leot@NetBSD.org>

The NetBSD Foundation

OpenTofu Day Europe 2026
March 23rd 2026
Amsterdam, Netherlands

# Outline

- Introduction
  - NetBSD
  - pkgsrc: the NetBSD package system
  - pkgsrc build phases
  - Why should we package OpenTofu and providers?
- Packaging
  - Structure of pkgsrc package
  - Go packaging
  - Requirements of OpenTofu, Terraform and providers packaging
  - A look to OpenTofu and providers in pkgsrc
- Testing migration from Terraform 0.12.x to OpenTofu 1.11.x
  - Overview for update/migration
  - Moto server-mode and Cram
  - Cram tests to exercise Terraform/OpenTofu upgrades
- Conclusions
- References

# Introduction

# whoami(1)

- Cloud architect
- NetBSD {Developer,Director}
- I love to contribute to Open Source
- I write Infrastructure as Code using OpenTofu both for work and for fun!

# My adventure with OpenTofu I

| | |
|---|---|
| September 2020 | Terraform v0.12.29 evaluation and introduction at work |
| January 2021 | Update of Terraform to v0.14.5 in pkgsrc-wip and several providers packaging |
| March 2021 | Proof of Concept of the pkgsrc Terraform/OpenTofu packaging infrastructure discussed in this talk |
| October 2023 | Packaging of OpenTofu v1.6.0-alpha2 |
| October 2023 | Making terraform-provider-* Terraform/OpenTofu agnostic |

# My adventure with OpenTofu II

| | |
|---|---|
| January 2024 | OpenTofu v1.6.0 released! |
| July 2025 | Master thesis on “Stateful policies for static code analysis of Infrastructure as Code states” |
| September 2025 | Packaging and upgrade testing of intermediate Terraform and OpenTofu releases |
| March 2026 | Request for comments and proposal to import OpenTofu packages and packaging infrastructure after pkgsrc-2026Q1 release |

# NetBSD

- Free, fast, secure, and highly portable Unix-like open source operating system
- Started as a fork from 386BSD in 1993
- As an Operating System contains common Unix userland programs, compiler (GCC or clang), debugger, X11
- Other third-party software available via pkgsrc

# pkgsrc: the NetBSD package system

- Default package manager for NetBSD and SmartOS
- Started as a fork from FreeBSD ports in 1997
- Supports a lot of Unix-like operating systems including macOS, Linux, Solaris, FreeBSD, OpenBSD and more
- More than 29000 packages available
- Quarterly stable release (e.g. pkgsrc-2026Q1) and rolling release (pkgsrc-current)
- Written in make, shell scripting (sh, AWK) and C
- NetBSD bulk builders provides binary packages
- ...we will dig in building the packages from source in this talk!
- pkgsrc-wip (pkgsrc work in progress) is a project to get more people involved in pkgsrc

# pkgsrc build phases

When building a package there are several phases:

fetch fetch source code (also known as distfiles)

patch apply pkgsrc patches and possible user's patches

build configure and build the package

package create the binary package (`.tgz` archive)

install install the binary package

### Example

Install Go 1.26.x from source:

```
$ cd pkgsrc/lang/go126
$ make install
```

# Why should we package OpenTofu and providers?

- No binaries available for NetBSD
- Install/update like any other packages
- Be more in control of the software supply chain
  (see Terraform Registry's Terms of Service change in August 2023)

# Packaging

# Structure of pkgsrc package

- Each package lives in a dedicated directory `category/package` in the pkgsrc tree, e.g. `editors/neovim`
- Files under `category/package`:
  - Makefile main package logic, how to fetch/configure/build/install package, dependencies, adjust configure/build flags
  - PLIST list files that will be installed by the package
  - distinfo contains BLAKE2s/SHA512 checksums and file size for each distfile
  - patches contains possible pkgsrc patches
  - DESCR long description of the package
  - ALTERNATIVES provides alternatives, e.g. `tofu` defaults to `tofu111`

# Go packaging

- Go packages uses `lang/go/go-module.mk` logic
- Source code of the package and all its Go modules are the distfiles
- Modules are listed via `GO_MODULES_FILES` variable
- Build is done via `go install -v ${GOFLAGS} ${GO_BUILD_PATTERN}`
- `GOFLAGS` can be used to set extra flags, e.g. `-ldflags`
- `GO_BUILD_PATTERN` by default is `./...`
- Everything installed under `${GOPATH}/bin` will be installed

# Requirements of OpenTofu, Terraform and providers packaging

- Multiple Terraform and OpenTofu versions to support upgrade/migration
- Multiple major versions of providers when there are incompatibilities
- OpenTofu and Terraform are Go packages
- Providers are Go packages (gRPC servers not depending on OpenTofu)
- OpenTofu and Terraform should be able to pick pkgsrc installed providers out of the box
- Avoid boilerplate for packaging (`terraform/provider.mk` eases provider packaging)

# A look to OpenTofu and providers in pkgsrc I

## wip/opentofu111/Makefile

```
DISTNAME=       opentofu-1.11.5
PKGNAME=        ${DISTNAME:S/opentofu/&111/}
CATEGORIES=     net
MASTER_SITES=   ${MASTER_SITE_GITHUB:=opentofu/}
GITHUB_PROJECT= opentofu
GITHUB_TAG=     v${PKGVERSION_NOREV}

MAINTAINER=     leot@NetBSD.org
HOMEPAGE=       https://opentofu.org/
COMMENT=        Open source infrastructure as code
LICENSE=        mpl-2.0

SUBST_CLASSES+=                 providers-prefix
SUBST_STAGE.providers-prefix=   pre-configure
SUBST_MESSAGE.providers-prefix= Honor providers installed via pkgsrc
SUBST_FILES.providers-prefix=   cmd/tofu/provider_source.go
SUBST_VARS.providers-prefix=    PREFIX

do-install:
	${INSTALL_PROGRAM} ${WRKDIR}/.gopath/${GOPATH_BIN}/tofu ${DESTDIR}${PREFIX}/bin/tofu111

.include "go-modules.mk"

.include "../../lang/go/go-module.mk"
.include "../../mk/bsd.pkg.mk"
```

# A look to OpenTofu and providers in pkgsrc II

### wip/opentofu111/ALTERNATIVES

```
bin/tofu @PREFIX@/bin/tofu111
```

### wip/opentofu111/PLIST

```
@comment $NetBSD$
bin/tofu111
```

### wip/opentofu111/patches/patch-provider__source.go

```
$NetBSD$

Honors providers installed via pkgsrc.

--- cmd/tofu/provider_source.go.orig    2021-03-24 18:39:36.000000000 +0000
+++ cmd/tofu/provider_source.go
@@ -160,6 +160,9 @@ func implicitProviderSource(services *di
                addLocalDir(dir)
        }

+       // Providers installed via pkgsrc
+       addLocalDir(filepath.Join("@PREFIX@", "share/terraform/plugins"))
+
        // Anything we found in local directories above is excluded from being
        // looked up via the registry source we're about to construct.
        var directExcluded getproviders.MultiSourceMatchingPatterns
```

# A look to OpenTofu and providers in pkgsrc III

wip/terraform-provider-random2/Makefile

```
DISTNAME=       terraform-provider-random-2.3.0
PKGNAME=        ${DISTNAME:S/terraform-provider-random/&2/}
CATEGORIES=     net
MASTER_SITES=   ${MASTER_SITE_GITHUB:=hashicorp/}
GITHUB_PROJECT= terraform-provider-random
GITHUB_TAG=     v${PKGVERSION_NOREV}

MAINTAINER=     pkgsrc-users@NetBSD.org
HOMEPAGE=       https://www.terraform.io/docs/providers/random/
COMMENT=        Terraform provider for randomness
LICENSE=        mpl-2.0

# Previous net/terraform-provider-random (2.x) did not have any version suffix
SUPERSEDES+=    terraform-provider-random-[0-9]*

TERRAFORM_PROVIDER_BIN=         terraform-provider-random
TERRAFORM_PROVIDER_NAMESPACE=   hashicorp
TERRAFORM_PROVIDER_TYPE=        random

TERRAFORM_PROVIDER_LEGACY_INSTALL=      yes

.include "go-modules.mk"

.include "../../lang/go/go-module.mk"
.include "../../wip/terraform/provider.mk"
.include "../../mk/bsd.pkg.mk"
```

# A look to OpenTofu and providers in pkgsrc IV

## wip/terraform-provider-random2/PLIST

```
@comment $NetBSD$
${PLIST.legacy}bin/terraform-provider-${TERRAFORM_PROVIDER_TYPE}_v${TERRAFORM_PROVIDER_VERSION}
share/terraform/plugins/${OPENTOFU_PROVIDER_HOSTNAME}/${TERRAFORM_PROVIDER_NAMESPACE}/${
     TERRAFORM_PROVIDER_TYPE}/${TERRAFORM_PROVIDER_VERSION}/${GO_PLATFORM}/terraform-provider-${
     TERRAFORM_PROVIDER_TYPE}
share/terraform/plugins/${TERRAFORM_PROVIDER_HOSTNAME}/${TERRAFORM_PROVIDER_NAMESPACE}/${
     TERRAFORM_PROVIDER_TYPE}/${TERRAFORM_PROVIDER_VERSION}/${GO_PLATFORM}/terraform-provider-${
     TERRAFORM_PROVIDER_TYPE}
```

# Testing migration from Terraform 0.12.x to OpenTofu 1.11.x

# Overview for update/migration

- Shortest upgrade/migration path is: Terraform 0.12.x → Terraform 0.13.x → Terraform 0.14.x → Terraform 1.x → OpenTofu 1.6.x → OpenTofu 1.11.x
- It would be nice to not worry about cloud providers bill :)

# Moto server-mode and Cram

- Moto is a Python library to mocks AWS Services
  - Its server-mode works well with OpenTofu and aws provider
- Cram is a functional testing framework for command line applications
  - It is like writing text snippets of interactive shell sessions

# Cram tests to exercise Terraform/OpenTofu upgrades I

requirements.t test to check installed Terraform/OpenTofu packages

```
#!/usr/bin/env cram

This test Terraform/OpenTofu upgrades in pkgsrc and goes from Terraform 0.12.x
to OpenTofu 1.11.x.

First we check that we have all needed Terraform versions...

We set CHECKPOINT_DISABLE variable because otherwise Terraform phones home and
check if there are new versions available via hashicorp/go-checkpoint:

  $ export CHECKPOINT_DISABLE=yes

We check all Terraform versions:

  $ terraform012 version
  Terraform v0.12.* (glob)
  $ terraform013 version
  Terraform v0.13.* (glob)
  $ terraform014 version
  Terraform v0.14.* (glob)
  $ terraform15 version
  Terraform v1.5.* (glob)
  on * (glob)
```

# Cram tests to exercise Terraform/OpenTofu upgrades II

## requirements.t test to check installed Terraform/OpenTofu packages (cont.)

```
We also check that we have OpenTofu installed:

  $ tofu16 version
  OpenTofu v1.6.* (glob)
  on * (glob)
  $ tofu110 version
  OpenTofu v1.10.* (glob)
  on * (glob)
  $ tofu111 version
  OpenTofu v1.11.* (glob)
  on * (glob)
```

### Example

Running Cram tests:

```
$ ./requirements.t
.
# Ran 1 tests, 0 skipped, 0 failed.
```

# Cram tests to exercise Terraform/OpenTofu upgrades III

### `aws-vpc/versions.tf` live module code

```
terraform {
  required_version = ">= 0.12.31"

  required_providers {
    aws = ">= 3.28, <= 4"
  }
}
```

### `aws-vpc/main.tf` live module code

```
provider "aws" {
  access_key                  = "test"
  secret_key                  = "test"
  region                      = "us-east-1"
  skip_credentials_validation = true
  skip_metadata_api_check     = true
  skip_requesting_account_id  = true

  endpoints {
    ec2 = "http://localhost:5000"
    iam = "http://localhost:5000"
  }
}
```

# Cram tests to exercise Terraform/OpenTofu upgrades IV

## aws-vpc/main.tf live module code (cont.)

```
module "vpc" {
  source  = "terraform-aws-modules/vpc/aws"
  version = "3.7.0"

  name = "testing-aws-vpc"
  cidr = "10.0.0.0/16"

  azs             = ["us-east-1a", "us-east-1b", "us-east-1c"]
  private_subnets = ["10.0.1.0/24", "10.0.2.0/24", "10.0.3.0/24"]
  public_subnets  = ["10.0.101.0/24", "10.0.102.0/24", "10.0.103.0/24"]

  enable_dns_hostnames = true
  enable_nat_gateway   = true

  tags = {
    Terraform = "true"
  }
}
```

# Cram tests to exercise Terraform/OpenTofu upgrades V

## upgrades-aws-vpc.t test to exercise upgrades

```
#!/usr/bin/env cram

This initialize and test the complete upgrade path from Terraform 0.12.x to
OpenTofu 1.11.x for aws-vpc module.

Enable job control and kill moto_server if tests are interrupted or exit for
any reasons.

  $ set -m
  $ trap ’echo cleanup >&2; kill -9 %+ 2>/dev/null; trap -; wait’ \
  > ALRM EXIT HUP INT TERM

Start moto_server in the background:

  $ moto_server 2>/dev/null &

Double-check that moto_server is running by checking its dashboard:

  $ curl --fail --head --max-time 10 --no-show-headers \
  >      --retry-connrefused --retry 3 --silent \
  > http://localhost:5000/moto-api/

Copy the entire module to a temporary directory so we do not need to worry and
clean .terraform and terraform.tfstate files:

  $ cp -r "${TESTDIR}/aws-vpc" "${CRAMTMP}/aws-vpc"
  $ cd "${CRAMTMP}/aws-vpc"
```

# Cram tests to exercise Terraform/OpenTofu upgrades VI

```

Set CHECKPOINT_DISABLE environment variable so Terraform does not phone home:

  $ export CHECKPOINT_DISABLE=yes

Initialize the project with Terraform 0.12.x and create the resources:

  $ terraform012 init >/dev/null
  $ terraform012 plan -detailed-exitcode >/dev/null
  [2]
  $ terraform012 apply -auto-approve >/dev/null

Upgrade to Terraform 0.13.x by following
<https://developer.hashicorp.com/terraform/language/v1.1.x/upgrade-guides/0-13>...
We ensure that there are no pending changes to do:

  $ terraform012 plan -detailed-exitcode >/dev/null

Run 0.13upgrade command so that provider requirements are rewritten to have
explicit source locations:

  $ terraform013 0.13upgrade -yes >/dev/null

Rewrite providers in the state too and re-init:

  $ terraform013 state replace-provider -auto-approve -- -/aws registry.terraform.io/hashicorp/aws >/dev/null
```

# Cram tests to exercise Terraform/OpenTofu upgrades VII

```
  $ terraform013 state replace-provider -auto-approve -- -/random registry.terraform.io/hashicorp/random >/dev/null
  $ terraform013 init >/dev/null

Run apply to complete the upgrade

  $ terraform013 apply -auto-approve >/dev/null

Upgrade to Terraform 0.14.x by following
<https://developer.hashicorp.com/terraform/language/v1.1.x/upgrade-guides/0-14>...
We ensure that there are no pending changes to do:

  $ terraform013 plan -detailed-exitcode >/dev/null

Rewrite providers in the state too and re-init:

  $ terraform014 init >/dev/null

Run apply to complete the upgrade

  $ terraform014 apply -auto-approve >/dev/null

Upgrade to Terraform 1.5 by following
<https://developer.hashicorp.com/terraform/language/v1.5/upgrade-guides/>...
We ensure that there are no pending changes to do:

  $ terraform014 plan -detailed-exitcode >/dev/null

```

# Cram tests to exercise Terraform/OpenTofu upgrades VIII

```
Rewrite providers in the state too and re-init:

  $ terraform15 init >/dev/null

Run apply to complete the upgrade

  $ terraform15 apply -auto-approve >/dev/null

Upgrade to OpenTofu 1.6 by following
<https://opentofu.org/docs/intro/migration/migration-guide/>...
We ensure that there are no pending changes to do:

  $ terraform15 plan -detailed-exitcode >/dev/null

Re-init:

  $ tofu16 init >/dev/null

Run apply to complete the upgrade:

  $ tofu16 apply -auto-approve >/dev/null

Upgrade to OpenTofu 1.10 by following
<https://opentofu.org/docs/intro/migration/migration-guide/>...
We ensure that there are no pending changes to do:

  $ tofu16 plan -detailed-exitcode >/dev/null

```

# Cram tests to exercise Terraform/OpenTofu upgrades IX

```
Re-init:

  $ tofu110 init >/dev/null

Run apply to complete the upgrade:

  $ tofu110 apply -auto-approve >/dev/null

Upgrade to OpenTofu 1.11 by following
<https://opentofu.org/docs/intro/migration/migration-guide/>...
We ensure that there are no pending changes to do:

  $ tofu110 plan -detailed-exitcode >/dev/null

Re-init:

  $ tofu111 init >/dev/null

Run apply to complete the upgrade:

  $ tofu111 apply -auto-approve >/dev/null

Stop moto_server and restore to default trap handlers:

  $ kill %+
  $ trap -
```

# Thanks

- Thanks to Benny Siegert and Taylor R Campbell!

# Conclusions

- We have introduced NetBSD and pkgsrc
- We have seen how to package Terraform/OpenTofu and providers in pkgsrc
- We have seen how a complete migration path from Terraform 0.12.x to OpenTofu 1.11.x can be tested via `moto_server` and Cram
- All this work should be available in pkgsrc-2026Q2!

# References I

Alistair Crooks, Hubert Feyrer, The pkgsrc Developers.
The pkgsrc guide.
https://www.NetBSD.org/docs/pkgsrc/.

OpenTofu.
Opentofu.
https://opentofu.org/.

HashiCorp.
Terms of use - terraform registry.
https://registry.terraform.io/terms.

Steve Pulec.
Moto - mock aws services.
https://github.com/getmoto/moto.

# References II

bitheap.
Cram: It's test time.
https://bitheap.org/cram/.

Leonardo Taccari.
Updating terraform/opentofu packages and providers: request for review and testing.
https://mail-index.NetBSD.org/tech-pkg/2026/03/04/msg032042.html.

# Thanks!

# Questions?

https://sched.co/2DY5x

in: Leonardo Taccari

✉: leot@NetBSD.org

m: @iamleot@mastodon.sdf.org

# Extra

# terraform/provider.mk I

wip/terraform/provider.mk

```
# $NetBSD$
#
# This file implements common logic for Terraform providers in pkgsrc.
#
# Package-settable variables:
#
# TERRAFORM_PROVIDER_HOSTNAME (optional)
#       Hostname of the Terraform registry that distributes the provider.
#
#       Default: registry.terraform.io
#
# OPENTOFU_PROVIDER_HOSTNAME (optional)
#       Hostname of the OpenTofu registry that distributes the provider.
#
#       Default: registry.opentofu.org
#
# TERRAFORM_PROVIDER_BIN
#       Binary name of the provider that will be installed under
#       TERRAFORM_PROVIDER_DIR.
#
# TERRAFORM_PROVIDER_NAMESPACE
#       Organizational namespace within the Terraform registry.
#
# TERRAFORM_PROVIDER_TYPE
#       Short name for the platform or system the provider manages.
```

# terraform/provider.mk II

```
#
# TERRAFORM_PROVIDER_VERSION (optional)
#       Version of the provider.
#
#       Default: ${PKGVERSION_NOREV}
#
# TERRAFORM_PROVIDER_DIR (optional)
#       Directory where the provider will be installed.
#
#       Default: share/terraform/plugins/${TERRAFORM_PROVIDER_HOSTNAME}/${TERRAFORM_PROVIDER_NAMESPACE}/${TERRAFORM_PROVIDER_TYPE}/${TERRAFORM_PROVIDER_VERSION}/${GO_PLATFORM}
#
# OPENTOFU_PROVIDER_DIR (optional)
#       Directory where the provider will be installed.
#
#       Default: share/terraform/plugins/${OPENTOFU_PROVIDER_HOSTNAME}/${TERRAFORM_PROVIDER_NAMESPACE}/${TERRAFORM_PROVIDER_TYPE}/${TERRAFORM_PROVIDER_VERSION}/${GO_PLATFORM}
#
# TERRAFORM_PROVIDER_LEGACY_INSTALL (optional)
#       Install the provider under ${PREFIX}/bin too so that can be used by
#       Terraform 0.12.x.
#
#       Default: no
#
# Keywords: opentofu terraform
#

TERRAFORM_PROVIDER_HOSTNAME?=   registry.terraform.io
```

# terraform/provider.mk III

```
TERRAFORM_PROVIDER_VERSION?=    ${PKGVERSION_NOREV}
TERRAFORM_PROVIDER_DIR?=        share/terraform/plugins/${TERRAFORM_PROVIDER_HOSTNAME}/${
     TERRAFORM_PROVIDER_NAMESPACE}/${TERRAFORM_PROVIDER_TYPE}/${TERRAFORM_PROVIDER_VERSION}/${
     GO_PLATFORM}

OPENTOFU_PROVIDER_HOSTNAME?=    registry.opentofu.org
OPENTOFU_PROVIDER_DIR?=         share/terraform/plugins/${OPENTOFU_PROVIDER_HOSTNAME}/${
     TERRAFORM_PROVIDER_NAMESPACE}/${TERRAFORM_PROVIDER_TYPE}/${TERRAFORM_PROVIDER_VERSION}/${
     GO_PLATFORM}

TERRAFORM_PROVIDER_LEGACY_INSTALL?=     no

PLIST_SUBST+=           TERRAFORM_PROVIDER_HOSTNAME=${TERRAFORM_PROVIDER_HOSTNAME}
PLIST_SUBST+=           TERRAFORM_PROVIDER_NAMESPACE=${TERRAFORM_PROVIDER_NAMESPACE}
PLIST_SUBST+=           TERRAFORM_PROVIDER_TYPE=${TERRAFORM_PROVIDER_TYPE}
PLIST_SUBST+=           TERRAFORM_PROVIDER_VERSION=${TERRAFORM_PROVIDER_VERSION}

PLIST_SUBST+=           OPENTOFU_PROVIDER_HOSTNAME=${OPENTOFU_PROVIDER_HOSTNAME}

PRINT_PLIST_AWK+=       { sub("${TERRAFORM_PROVIDER_HOSTNAME}", "$${TERRAFORM_PROVIDER_HOSTNAME}") }
PRINT_PLIST_AWK+=       { sub("${TERRAFORM_PROVIDER_NAMESPACE}", "$${TERRAFORM_PROVIDER_NAMESPACE}") }
PRINT_PLIST_AWK+=       { gsub("${TERRAFORM_PROVIDER_TYPE}", "$${TERRAFORM_PROVIDER_TYPE}") }
PRINT_PLIST_AWK+=       { sub("${TERRAFORM_PROVIDER_VERSION}", "$${TERRAFORM_PROVIDER_VERSION}") }

PRINT_PLIST_AWK+=       { sub("${OPENTOFU_PROVIDER_HOSTNAME}", "$${OPENTOFU_PROVIDER_HOSTNAME}") }

PRINT_PLIST_AWK+=       /bin/ { sub("^", "$${PLIST.legacy}") }

```

# terraform/provider.mk IV

```
PLIST_VARS+=    legacy
.if ${TERRAFORM_PROVIDER_LEGACY_INSTALL:tl} == "yes"
PLIST.legacy=   yes
.endif

INSTALLATION_DIRS+=     ${TERRAFORM_PROVIDER_DIR}
INSTALLATION_DIRS+=     ${OPENTOFU_PROVIDER_DIR}
.if ${TERRAFORM_PROVIDER_LEGACY_INSTALL:tl} == "yes"
INSTALLATION_DIRS+=     bin
.endif


#
# This should be overwritten via ‘do-install:’ target, however
# lang/go/go-{module,package}.mk already overwrite them.
#
.if !target(pre-install)
pre-install:
        ${RUN} \
        if [ -f ${WRKDIR}/.gopath/bin/${TERRAFORM_PROVIDER_BIN} ]; then \
                ${INSTALL_PROGRAM} ${WRKDIR}/.gopath/bin/${TERRAFORM_PROVIDER_BIN} \
                    ${DESTDIR}${PREFIX}/${TERRAFORM_PROVIDER_DIR} ; \
                ${LN} -sf ${PREFIX}/${TERRAFORM_PROVIDER_DIR}/${TERRAFORM_PROVIDER_BIN} \
                    ${DESTDIR}${PREFIX}/${OPENTOFU_PROVIDER_DIR}/${TERRAFORM_PROVIDER_BIN} ; \
                { [ -d ${WRKDIR}/.gopath ] && chmod -R +w ${WRKDIR}/.gopath || ${TRUE} ; } ; \
                ${RM} -rf ${WRKDIR}/.gopath/bin ; \
        elif [ -f ${WRKDIR}/bin/${TERRAFORM_PROVIDER_BIN} ]; then \
                ${INSTALL_PROGRAM} ${WRKDIR}/bin/${TERRAFORM_PROVIDER_BIN} \
```

# terraform/provider.mk V

```
                    ${DESTDIR}${PREFIX}/${TERRAFORM_PROVIDER_DIR} ; \
                ${LN} -sf ${PREFIX}/${TERRAFORM_PROVIDER_DIR}/${TERRAFORM_PROVIDER_BIN} \
                    ${DESTDIR}${PREFIX}/${OPENTOFU_PROVIDER_DIR}/${TERRAFORM_PROVIDER_BIN} ; \
                ${RM} -rf ${WRKDIR}/bin ; \
                ${RM} -rf ${WRKDIR}/pkg ; \
        fi
.  if ${TERRAFORM_PROVIDER_LEGACY_INSTALL:tl} == "yes"
                ${LN} -sf ${PREFIX}/${TERRAFORM_PROVIDER_DIR}/${TERRAFORM_PROVIDER_BIN} \
                    ${DESTDIR}${PREFIX}/bin/${TERRAFORM_PROVIDER_BIN}_v${TERRAFORM_PROVIDER_VERSION} ;
.  endif
.endif
```