.Deb

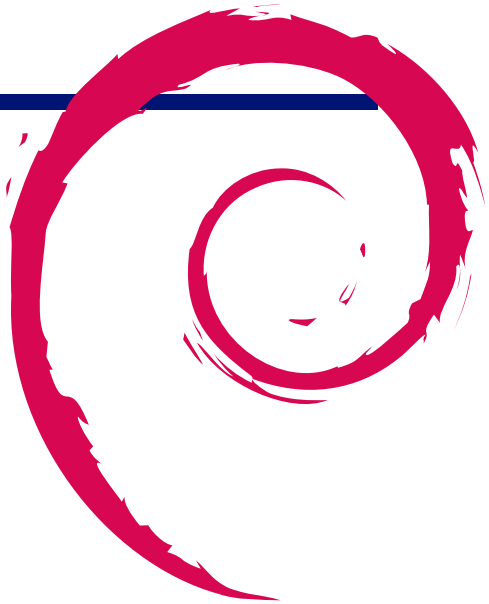
銀河系唯一のDebian専門誌

2020 年 5 月 16 日

aptly特集

デビアン勉強会

目次



1　最近の Debian 関連のミーティング報告

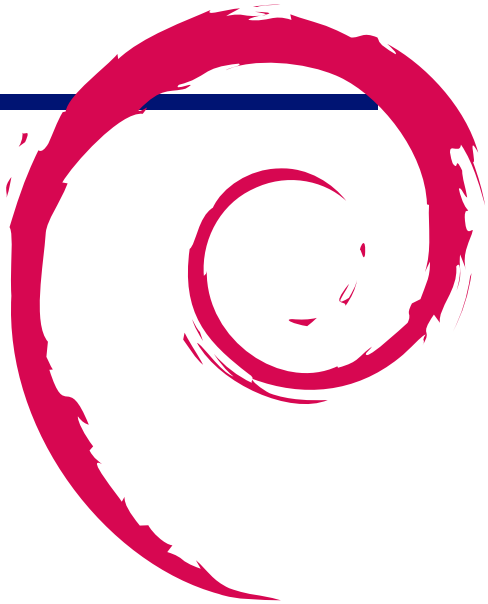
杉本　典充

1.1　2020 年 4 月度 東京エリア・関西合同 Debian 勉強会

2020 年 4 月 18 日 (土) に東京エリア Debian 勉強会と関西 Debian 勉強会の合同でオンラインによる Debian 勉強会を開催しました。参加者は 35 名でした。セミナー発表を 2 つ行いました。

セミナー「Wireguard 実践入門」という表題で西山和広さんが発表しました。Wireguard は VPN 接続するアプリケーションであり、サーバとクライアントをスター型で接続する構成の他、メッシュ型の接続も可能との説明でした。linux-5.6 より古いカーネルの場合はカーネルモジュールを DKMS でビルドして利用し、linux-5.6 以降ではカーネルにマージされているなど最新の動向の説明がありました。また、質疑応答では OpenVPN と対比する質問がありました。

セミナー「Jitsi を使ったビデオ会議サーバの作り方」という表題で杉本典充さんが発表しました。2020 年 3 月に開催した Debian 勉強会はインターネットを用いたビデオ会議形式で開催しており、そのときに利用したアプリケーションである Jitsi（ジッチー）を説明しました。Jitsi を GCP のサーバにインストールして使えるようにする手順の説明、設定のカスタマイズ方法の紹介、2020 年 3 月に開催した Debian 勉強会のときのサーバの CPU 利用率とトラフィックグラフを紹介しました。



2 事前課題

杉本　典充

今回の事前課題は以下です。

1. aptly はご存じですか

2.1 dictoss

1. 知らない

2.2 yy_y_ja_jp

1. 知らない

2.3 uwabami

1. 知っているが、使ったことはない

2.4 yosuke_san

1. 知らない

2.5 iwamatsu

1. 使ったことがある

2.6 あ (cpa119)

1. 知らない

2.7 MasanoriYoshida

1. 知らない

2.8 NOKUBI Takatsugu (knok)

1. 知らない

2.9 su_do

1. 知らない

2.10 TANIGUCHI Takaki (takaki_t)

1. 知らない

2.11 noncatalyst

1. 知らない

2.12 kozo2

1. 知らない

2.13 matoken

1. 知っているが、使ったことはない

2.14 プライニング　ノルベルト (norbu)

1. 使ったことがある

2.15 Katsuki Kobayashi (rarewin)

1. 知らない

2.16 koedoyoshida

1. 知らない

2.17 **榎真治** (enoki)

1. 知らない

2.18 daromart

1. 知っているが、使ったことはない

2.19 Incognito (cyberconn)

1. 知らない

2.20 ipv6waterstar

1. 知っているが、使ったことはない

2.21 Kazuhiro NISHIYAMA (znz)

1. 知らない

2.22 Kouhei Maeda (mkouhei)

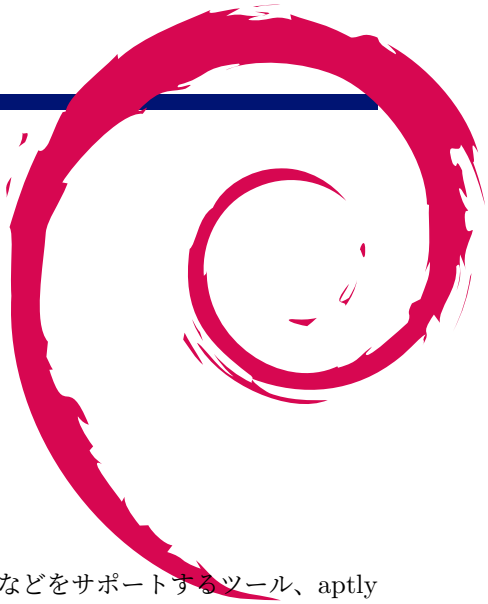
1. 知っているが、使ったことはない

2.23 kenhys

1. 知っているが、使ったことはない

2.24 gdevmjc

1. 知らない



3 aptly

岩松 信洋

Debian パッケージリポジトリのミラー作成、独自のパッケージリポジトリ作成などをサポートするツール、aptly について説明します。

3.1 aptly とは

aptly (`https://www.aptly.info/`) は Debian パッケージリポジトリの作成・管理するツールの一つです。パッケージリポジトリの作成やミラーリングを行うツールとして、dpkg-dev、apt-ftparchive、reprepro、archvsync などがありますが、これらは機能が限定的であり、いくつかのツールを組み合わせて使うことがほとんどです。aptly はこれらをまとめた機能を提供しています。

- パッケージリポジトリのフルミラーリングおよびフィルターミラーリング (アーキテクチャー、コンポーネント、パッケージ)
- パッケージリポジトリのスナップショット作成
- パッケージリポジトリの更新、マージ、リリース
- パッケージリポジトリサーバー機能
- ローカルパッケージリポジトリの作成
- パッケージリポジトリ操作用 REST API の提供
- Amazon S3 などのクラウドストレージへのアップロード機能

コマンドが統一されており、REST API も提供されていることから、CI や CD と親和性が高く、パッケージやパッケージを使った製品のテスト等に役立ちます。またスナップショット機能があり、スナップショットをベースにパッケージリポジトリを構築するため、構成管理や組込み機器のルートファイルシステムの管理などにも利用できます。

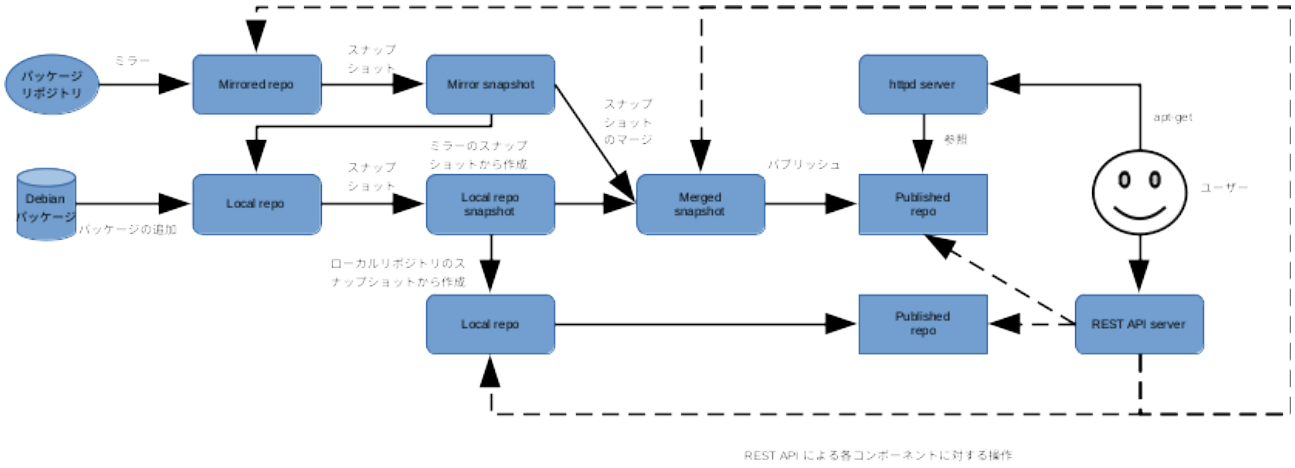
3.1.1 aptly の構成要素

aptly の構成要素としてはミラーリポジトリ、ローカルパッケージリポジトリ、スナップショットリポジトリ、パッケージリポジトリの 4 つに分類することができ、前の 3 つで提供されるパッケージ情報を使って、公開するパッケージリポジトリを作成することになります。

- ミラーリポジトリ
  ミラーリポジトリは Debian パッケージを提供しているパッケージリポジトリをミラーしたものを指します。Debian パッケージなので、Debian の公式・非公式関係なくミラーできます。また Ubuntui や PPA からもミラーを構築できます。
- ローカルパッケージリポジトリ
  ローカルパッケージリポジトリは、独自に作成した Debian パッケージ用リポジトリを管理する場合に使用し

ます。

- スナップショット
  ローカルパッケージリポジトリとミラーリポジトリのスナップショットを作成します。Debian の公式リポジトリを直接参照している場合、対処のパッケージが更新されると、古いバージョンのパッケージが対象リポジトリ参照できなくなり、同じパッケージ構成を持つマシンなどを構成することが難しくなります。このような場合、ある時点のパッケージ情報をスナップショットとして保存するなどの対策する必要があるのですが、aptly のリポジトリスナップショット機能を使うと、容易に実現できます、
- パッケージリポジトリ
  ミラーリポジトリ、ローカルパッケージリポジトリ、スナップショットは aptly で管理されるデータベース上で管理されます。これらを組みわせて作成したリポジトリデータを元にパブリッシュ(公開) することで初めてパッケージリポジトリとして利用できるようになります。



- ミラーリポジトリは他のパッケージリポジトリから作成します。
- ローカルパッケージはパッケージを追加するか、ローカルミラーから作成できます。
- 作成したローカルミラーやローカルリポジトリはそのままではパッケージリポジトリとしては利用できず、パブリッシュ(公開) する必要があります。
- ローカルミラーやローカルリポジトリはスナップショットを作成できます。
- スナップショットからスナップショットも作成でき、スナップショット同士の依存関係もメタデータとして持ちます。
- スナップショット同士でマージできます。
- パブリッシュしたパッケージリポジトリは aptly で提供される http サーバーを使用、またはその他 http サーバーを使って公開できます。

3.1.2 aptly を使用する際に注意すること

aptly ではパッケージリポジトリを公開する場合に、GnuPG による署名が必要となります (必須ではなく、オプションで署名および確認を無視できます)。aptly は一度ローカルにパッケージとメタ情報を保存し、それらを組み合わせて、パッケージリポジトリに必要なメタデータ (Release ファイルなど) を再構築します。再構築したメタデータはフルミラーであっても内容が異なるため、正しくパッケージリポジトリを運用するにはこれらに GnuPG による署名が必要になります。メタデータを含めたフルミラーを構築したい場合は archvsync (`https://salsa.debian.org/mirror-team/archvsync`) や apt-mirror パッケージを使うと良いでしょう。

またパッケージリポジトリの構築にはハードリンクを使います。よってハードリンクの制限が適用されることに注意が必要です。Amazon S3 などのクラウドストレージにパッケージリポジトリをパブリッシュする際にはこの制限はありません。

以下からは、aptly に使い方について説明します。

3.2 aptly のインストール

aptly は公式 Debian パッケージとして提供されています。執筆時の Debian パッケージバージョンは 1.3.0+ds1-4 となっています。以下のコマンドでインストールできます。

```
$ sudo apt update
$ sudo apt install aptly
```

また Upstream のバージョンは 1.4.0 となっており、GnuPG 2.x へのサポート強化などが行われています。最新バージョンを使いたい場合には、以下のようにアップストリーム が提供するリポジトリで提供されているパッケージが利用できます。 ディストリビューション部が squeeze となっていますが、一つのディストリビューションのみで提供されているので、気にしなくても良いです。

```
$ wget -qO - https://www.aptly.info/pubkey.txt | sudo apt-key add -
$ sudo sh -c "echo deb http://repo.aptly.info/ squeeze main > /etc/apt/sources.list.d/aptly.list"
$ sudo apt update
$ sudo apt install aptly
```

3.2.1 aptly の設定

aptly の設定は $HOME/.aptly.conf に記載されています。内容は aptly config コマンドの show サブコマンドで確認できます。もちろん普段お使いのエディタ等で確認してもかまいません。

```
$ aptly config show
{
    "rootDir": "/home/aptly/.aptly",
    "downloadConcurrency": 4,
    "downloadSpeedLimit": 0,
    "architectures": [],
    "dependencyFollowSuggests": false,
    "dependencyFollowRecommends": false,
    "dependencyFollowAllVariants": false,
    "dependencyFollowSource": false,
    "dependencyVerboseResolve": false,
    "gpgDisableSign": false,
    "gpgDisableVerify": false,
    "gpgProvider": "gpg",
    "downloadSourcePackages": false,
    "skipLegacyPool": true,
    "ppaDistributorID": "ubuntu",
    "ppaCodename": "",
    "skipContentsPublishing": false,
    "FileSystemPublishEndpoints": {},
    "S3PublishEndpoints": {},
    "SwiftPublishEndpoints": {}
}
```

詳細は省きますが、重要なのは rootDir の項目で、ここで指定されるディレクトリに aptly が管理するパッケージのメタデータやローカルパッケージリポジトリ等が格納されます。また Amazon S3 や OpenStack Swift へのアップロードを行う際には、S3PublishEndpoints や SwiftPublishEndpoints に対してアクセスキー等を設定します。各項目の詳細は Configuration (`https://www.aptly.info/doc/configuration/`) を確認してください。

3.3 GnuPG 鍵の作成とパッケージリポジトリの署名確認必要な公開鍵のインポート

aptly を使う前に、GnuPG 鍵の作成と、パッケージミラー元の GnuPG 公開鍵をインポートする必要があります。前述したように、正しくパッケージリポジトリを運用するには GnuPG による署名が必要なため、GnuPG 鍵を作成します。既に GnuPG 鍵を持っている場合も、リポジトリ管理用に別途 GnuPG 鍵を作成しておくとよいでしょう。GnuPG 鍵の作成には以下のように実行します。

```
$ gpg --gen-key

<中略>

本名: aptly
電子メール・アドレス: aptly@example.com
次のユーザ ID を選択しました:
    "aptly <aptly@example.com>"

<中略>
公開鍵と秘密鍵を作成し、署名しました。

pub   rsa3072 2020-05-01 [SC] [有効期限: 2022-05-01]
      E67BDF3221F4CDFD47F4A3639A64752708B70EFA
uid                      aptly <aptly@example.com>
sub   rsa3072 2020-05-01 [E] [有効期限: 2022-05-01]
```

次にパッケージミラー元の GnuPG 公開鍵をインポートします。パッケージリポジトリのミラー元の情報を取得する際にミラー元の署名を確認する必要があるためです。Debian のオフィシャルパッケージリポジトリを使う場合には以下のように実行します。

```
$ sudo apt install debian-archive-keyring
$ gpg --no-default-keyring --keyring /usr/share/keyrings/debian-archive-keyring.gpg --export | \
       gpg --no-default-keyring --keyring trustedkeys.gpg --import
```

次に Debian で提供されているパッケージを使っている人は、GnuPG v2 で作成した鍵を v1 にコンバートする必要があります。これは Debian のパッケージが GnuPG v1 に依存しているためです。公開鍵 E67BDF3221F4CDFD47F4A3639A64752708B70EFA をコンバートし、GnuPG v1 にインポートする方法を以下に示します。

```
$ gpg --output gpg2_export_pub.gpg --armor --export E67BDF3221F4CDFD47F4A3639A64752708B70EFA
$ gpg --output gpg2_export_sec.gpg --armor --export-secret-key E67BDF3221F4CDFD47F4A3639A64752708B70EFA
$ gpg1 --import gpg2_export_pub.gpg
gpg: 鍵 08B70EFA: 公開鍵"aptly <aptly@example.com>"をインポートしました
gpg:               処理数の合計: 1
gpg:                  インポート: 1  (RSA: 1)
gpg: 最小の「ある程度の信用」3、最小の「全面的信用」1、PGP 信用モデル
gpg: 深さ: 0  有効性:   1  署名:   0  信用: 0-, 0q, 0n, 0m, 0f, 1u
gpg: 次回の信用データベース検査は、2022-05-01 です
$ gpg1 --import --allow-secret-key-import gpg2_export_sec.gpg
gpg: 鍵 08B70EFA: 秘密鍵をインポートしました
gpg: 鍵 08B70EFA:"aptly <aptly@example.com>"変更なし
gpg:               処理数の合計: 1
gpg:                  変更なし: 1
gpg:            秘密鍵の読み込み: 1
gpg:          秘密鍵のインポート: 1
```

また作成するパッケージリポジトリにアクセスするマシンにパッケージリポジトリの公開鍵を登録することを忘れないようにしましょう。公開鍵の出力方法と出力した公開鍵を登録する方法を以下に示します。

```
$ gpg --export --armor > aptly.pub
```

```
$ sudo apt-key add aptly.pub
```

3.4 パッケージミラーリポジトリの作成

ローカルにパッケージミラーリポジトリを作成する場合、 aptly mirror コマンドの create サブコマンドを使用します。コマンドにローカルで利用する名前 (name)、リポジトリの URL、ディストリビューション、コンポーネントを指定します。例えば、

- 名前: debian-buster
- リポジトリ URL: http://deb.debian.org/debian/
- ディストリビューション: buster
- コンポーネント: main
- アーキテクチャ: すべて

上記のようなパッケージミラーリポジトリを作成する場合、以下のように実行します。実行するとリポジトリが初期化されます。

```
$ aptly mirror create debian-buster http://deb.debian.org/debian/ buster main
```

作成したパッケージミラーリポジトリの情報は list サブコマンドで確認できます。

```
$ aptly mirror list
List of mirrors:
 * [debian-buster]: http://deb.debian.org/debian/ buster

To get more information about mirror, run ‘aptly mirror show <name>‘.
```

また詳細な情報が必要な場合は show サブコマンドで確認できます。

```
$ aptly mirror show debian-buster
Name: debian-buster
Archive Root URL: http://deb.debian.org/debian/
Distribution: buster
Components: main
Architectures: amd64, arm64, armel, armhf, i386, mips, mips64el, mipsel, ppc64el, s390x
Download Sources: no
Download .udebs: no
Last update: never

Information from release file:
Acquire-By-Hash: yes
Architectures: amd64 arm64 armel armhf i386 mips mips64el mipsel ppc64el s390x
Changelogs: http://metadata.ftp-master.debian.org/changelogs/@CHANGEPATH@_changelog
Codename: buster
Components: main contrib non-free
Date: Sat, 09 May 2020 09:51:02 UTC
Description:  Debian 10.4 Released 09 May 2020

Label: Debian
Origin: Debian
Suite: stable
Version: 10.4
```

まだこの状態ではリポジトリのメタデータやパッケージがない状態のため、ミラー元から取得する必要があります。取得するには update サブコマンドを実行します。実行するとメタデータを取得し、パッケージのミラーを開始します。

```
$ aptly mirror update debian-buster
```

上記では指定したディストリビューションのコンポーネントをすべてミラーします。特定のアーキテクチャのみや、ミラーしたいパッケージを指定したい場合には、オプションを使うことで制御できます。以下ではよく利用されるオプションを以下に示します。

- アーキテクチャを指定する: -architectures
- パッケージをフィルターする: -filter
- フィルターで指定されたパッケージに依存するパッケージもミラーする: -filter-with-deps
- ソースパッケージもミラーする: -with-sources

アーキテクチャを amd64 と arm64、ミラーするパッケージを busybox、依存するパッケージとソースパッケージもミラーする場合には以下のように実行します。

```
$ aptly mirror create -architectures=amd64,arm64 -filter='busybox' \
    -filter-with-deps -with-sources busybox-mirror http://deb.debian.org/debian/ buster main
```

実行し、取得したパッケージは $HOME/.aptly ディレクトリ以下に保存されます。ファイルの配置は git リポジトリの git object のような配置になっています（sha265sum 値）。

```
.aptly
├──── db
│   ├──── 000004.ldb
│   ├──── 000007.ldb
│   ├──── 000008.log
│   ├──── CURRENT
│   ├──── LOCK
│   ├──── LOG
│   └──── MANIFEST-000009
└──── pool
    ├──── 1b
    │   └──── 00
    │       └──── f7cef567645a7e695caf6c1ad395_gcc-8-base_8.3.0-6_amd64.deb
    ├──── 1e
    │   └──── 32
    │       └──── ea742bddec4ed5a530ee2f423cdf_busybox_1.30.1-4_amd64.deb
<省略>
```

作成したミラーで提供されるパッケージを確認するには aptly mirror コマンドの search サブコマンド使用します。検索するミラー名と検索用のクエリーを指定し、実行します。以下に パッケージ名が busy 、アーキテクチャが arm64 であるパッケージを検索するには以下のように実行します。検索用クエリーを指定しない場合は提供されるパッケージが全て出力されます。

```
$ aptly mirror search busybox-mirror 'Name (% busy*), $Architecture (arm64)'
busybox_1:1.30.1-4_arm64
busybox-static_1:1.30.1-4_arm64
```

3.5 ローカルパッケージリポジトリの作成

ローカルパッケージリポジトリは、独自で作成したパッケージをリポジトリで管理したい場合などに使用します。まず管理するためのリポジトリを作成する必要があります。リポジトリを作成するには、aptly repo コマンド の create サブコマンドに作成するリポジトリ名を指定して実行します。このリポジトリ名は Debian パッケージで利用される ディストリビューション名 (unstable, buster など) と紐付けられています。

```
$ aptly repo create my-repo
```

またローカルリポジトリを作成する場合、スナップショットからも作成できます。

```
$ aptly repo create my-repo from snapshot snapshot-name
```

3.5.1 ローカルパッケージリポジトリへのパッケージの追加

create サブコマンドで作成した段階では、ローカルパッケージリポジトリにまだパッケージが登録されていない状態なので、パッケージを追加します。追加する方法は以下の方法があります。

add サブコマンドを使ったパッケージの追加

add サブコマンドは 特定の debian バイナリパッケージ、または dsc (Debian source packages control file) を元にソースパッケージを追加したい場合に利用します。パッケージを指定してローカルパッケージリポトリに追加したい場合は、add サブコマンドにパッケージファイルのパスを指定して実行します。

```
$ aptly repo add my-repo pool/liblz4-1_1.9.2-2_amd64.deb
Loading packages...
[+] liblz4-1_1.9.2-2_amd64 added
```

またソースパッケージを追加したい場合には dsc ファイルを指定して実行します。

```
$ aptly repo add my-repo pool/lz4_1.9.2-2.dsc
Loading packages...
[+] lz4_1.9.2-2_source added
```

特定のディレクトリにあるパッケージすべて追加したい場合には対象のディレクトリを指定すればよいです。

```
$ aptly repo add my-repo pool
Loading packages...
[+] liblz4-1-dbgsym_1.9.2-2_amd64 added
[+] liblz4-1_1.9.2-2_amd64 added
[+] liblz4-dev_1.9.2-2_amd64 added
[+] liblz4-tool_1.9.2-2_all added
[+] lz4-dbgsym_1.9.2-2_amd64 added
[+] lz4_1.9.2-2_source added
[+] lz4_1.9.2-2_amd6 added
```

add コマンドを使った場合、ファイルのサイズやファイルハッシュ値の確認は行われないため注意が必要です。より安全にパッケージをリポジトリに追加したい場合、次で説明する include サブコマンドを使います。

include サブコマンドを使ったパッケージの追加

ローカルパッケージリポジトリへのパッケージの追加には add サブコマンドの他に include サブコマンドも利用可能です。add サブコマンドとの違いは処理の対象が .changes ファイルである事です。.changes ファイルを指定して実行するには以下のように実行します。実行すると、.changes ファイルに記載されているパッケージが登録されます。特に指定しない場合、実行後にはパッケージファイル等は削除されてしまうので、削除したくない場合には-no-remove-files オプションを指定する必要があります。また.changes に GnuPG による署名を行っておらず、署名の有無を無視したい場合には -accept-unsigned オプションを指定する必要があります。

```
Format: 1.8
Date: Tue, 12 Nov 2019 07:58:47 +0900
Source: lz4

<中略>

Checksums-Sha1:
 44348438b55dc32132df8039513a57c4c1cd87a5 1073 lz4_1.9.2-2.dsc
 3790bc3c9e6d4e26e5063855cf847f5af91d8420 12712 lz4_1.9.2-2.debian.tar.xz
 ce4d29ed984de507203d917e671c1894c9778e8c 323008 liblz4-1-dbgsym_1.9.2-2_amd64.deb
 aed6d84f4636d37cd94b90e9ce0320dee853eca6 57148 liblz4-1_1.9.2-2_amd64.deb
 018ef0f8bf34411b4b928072f5c1f2d1abf8103f 76752 liblz4-dev_1.9.2-2_amd64.deb
 bf32ee9eee03ef69d6b0526bed92dcc95af57883 5088 liblz4-tool_1.9.2-2_all.deb
 39f1aa1950030608ad528a4f4fe951b37e0e4956 410216 lz4-dbgsym_1.9.2-2_amd64.deb
 dec576464d1c5e4b2b69e59ab29208f4e6d741eb 6648 lz4_1.9.2-2_amd64.buildinfo
 689287ecea701b76f99c5d30040ca584bdc58fde 84076 lz4_1.9.2-2_amd64.deb

<省略>
```

```
$ aptly repo include -repo my-repo pool/lz4_1.9.2-2_amd64.changes
Loading repository my-repo for changes file lz4_1.9.2-2_amd64.changes...
[+] liblz4-1-dbgsym_1.9.2-2_amd64 added
[+] liblz4-1_1.9.2-2_amd64 added
[+] liblz4-dev_1.9.2-2_amd64 added
[+] liblz4-tool_1.9.2-2_all added
[+] lz4-dbgsym_1.9.2-2_amd64 added
[+] lz4_1.9.2-2_source added
[+] lz4_1.9.2-2_amd64 added
```

上記では実行する際に、-repo オプションでリポジトリ名を指定しています。これは changelog で指定されているディストリビューション名 (unstable など) がリポジトリ名と合致しない場合にエラーになるためです。

またディレクトリを指定する場合には、.changes が格納されているデレクトリを指定します。

```
$ aptly repo include -repo my-repo pool
Loading repository my-repo for changes file lz4_1.9.2-2_amd64.changes...
[+] liblz4-1-dbgsym_1.9.2-2_amd64 added
[+] liblz4-1_1.9.2-2_amd64 added
[+] liblz4-dev_1.9.2-2_amd64 added
[+] liblz4-tool_1.9.2-2_all added
[+] lz4-dbgsym_1.9.2-2_amd64 added
[+] lz4_1.9.2-2_source added
[+] lz4_1.9.2-2_amd64 added
```

指定したディレクトリに .changes ファイルがない場合や、.changes の内容が異なる場合には処理されません。

```
$ aptly repo include -repo my-repo -no-remove-files -accept-unsigned pool
[!] unable to process file lz4_1.9.2-2_amd64.changes: size mismatch: expected 12712 != obtained 12568
[!] Some files were skipped due to errors:
  pool/lz4_1.9.2-2_amd64.changes
ERROR: some files failed to be added
```

ローカルミラーからの追加

aptly mirror コマンドによって作成したローカルミラーで提供されているパッケージもローカルリポジトリに追加できます。この場合 import サブコマンドを利用します。

例えば、ローカルミラー busybox-mirror にある busybox で始まるパッケージを my-repo ローカルリポジトリに追加する場合には以下のように実行します。

```
$ aptly repo import busybox-mirror my-repo busybox
Loading packages...
[o] busybox_1:1.30.1-4_source imported
[o] busybox-static_1:1.30.1-4_amd64 imported
[o] busybox_1:1.30.1-4_arm64 imported
[o] busybox_1:1.30.1-4_amd64 imported
[o] busybox-static_1:1.30.1-4_arm64 imported
```

Package Queries(`https://www.aptly.info/doc/feature/query/`) にパッケージ名の指定方法等に関する説明があるので興味のある方は参照してください。

3.5.2 ローカルパッケージリポジトリの削除

ローカルパッケージリポジトリの情報を削除したい場合には、drop サブコマンドに削除したいリポジトリ名を指定します。

```
$ aptly repo drop my-repo
```

3.5.3 その他ローカルリポジトリに関する操作

- repo list
  作成されたリポジトリのリストを出力します。
- repo copy
  リポジトリにあるパッケージを指定したリポジトリにコピーします。コピー先のリポジトリは先に作成しておく必要がある点に注意が必要です。

  ```
  $ aptlu repo create my-repo-next
  $ aptly repo copy my-repo my-repo-next 'Name (%lib*)'
  Loading packages...
  [o] liblz4-tool_1.9.2-2_all copied
  [o] liblz4-1_1.9.2-2_amd64 copied
  [o] liblz4-1-dbgsym_1.9.2-2_amd64 copied
  [o] liblz4-dev_1.9.2-2_amd64 copied
  ```

- repo move
  リポジトリにあるパッケージを指定したリポジトリに移動します。コピー先のリポジトリは先に作成しておく必要がある点に注意が必要です。
- repo remove
  指定したパッケージをリポジトリから削除します。
- repo search
  リポジトリで提供されているパッケージを検索します。
- repo edit
  リポジトリに関する情報を修正します。
- repo rename
  リポジトリ名を変更します。

3.6 スナップショット

aptly mirror update コマンドを実行した場合、ミラー元のパッケージ情報等と完全に同期するため、ミラー元で削除されたパッケージはローカルミラーでも削除され、更新されたパッケージも古いものは残っていない状態になってしまいます。またローカルパッケージリポジトリでも運用によっては上記と那じようなことが起こる可能性もあります。このような事が起きても困らないようにするために、スナップショットを作成し、パッケージの構成を維持できるよう管理します。

3.6.1 スナップショットの作成と確認

スナップショットを作成するには、aptly snapshot コマンドの create サブコマンドを使用します。例えば、ローカルミラー buster-mirror のスナップショットを buster-mirror-2020512 として作成するには以下のように実行します。

```
$ aptly snapshot create buster-mirror-2020512 from mirror buster-mirror
```

スナッショットは ローカルリポジトリからも作成できます。ローカルリポジトリ my-repo のスナップショットを my-repo-20200512 として作成するには以下のように実行します。

```
$ aptly snapshot create imy-repo-20200510 from repo my-repo
```

現在作成されているスナップショットを確認するには list サブコマンド、スナップショットの情報を確認するにはは show サブコマンドを使います。

```
$ aptly snapshot list
List of snapshots:
 * [busybox-buster-mirror-20200512]: Snapshot from mirror [busybox-buster-mirror]: http://deb.debian.org/debian/ buster
 * [my-repo-20200512]: Snapshot from local repo [my-repo]

To get more information about snapshot, run ‘aptly snapshot show <name>‘.
$ aptly snapshot show my-repo-20200512
Name: my-repo-20200512
Created At: 2020-05-15 07:06:32 UTC
Description: Snapshot from local repo [my-repo]
Number of packages: 12
Sources:
  my-repo [local]
```

3.6.2 スナップショットのマージ

作成したスナップショットをマージし、新しいスナップショットを作成するには merge サブコマンドを使います。作成したいスナップショット名にマージするスナップショットを指定します。

```
$ aptly snapshot merge my-product-release-20200512 my-repo-20200512 busybox-buster-mirror-20200512

Snapshot my-product-release-20200512 successfully created.
You can run ’aptly publish snapshot my-product-release-20200512’ to publish snapshot as Debian repository.
$ aptly snapshot list
List of snapshots:
 * [busybox-buster-mirror-20200512]: Snapshot from mirror [busybox-buster-mirror]: http://deb.debian.org/debian/ buster
 * [my-product-release-20200512]: Merged from sources: ’my-repo-20200512’, ’busybox-buster-mirror-20200512’
 * [my-repo-20200512]: Snapshot from local repo [my-repo]

To get more information about snapshot, run ‘aptly snapshot show <name>‘.

$ aptly snapshot show my-product-release-20200512
Name: my-product-release-20200512
Created At: 2020-05-15 07:20:35 UTC
Description: Merged from sources: ’my-repo-20200512’, ’busybox-buster-mirror-20200512’
Number of packages: 15
Sources:
  my-repo-20200512 [snapshot]
  busybox-buster-mirror-20200512 [snapshot]
```

3.6.3 スナップショットの検証

スナップショットの状態によってはパッケージの依存関係が維持できてない場合があります。これはパッケージの追加し忘れや、ローカルミラーリポジトリを構築した際に -filter-with-deps オプションを付けていなかったことなどが

原因となります。このような状態を確認するために、スナップショット内容を検証するサブコマンド verify が用意されています。以下では、-filter-with-deps を付けずに busybox から始まるパッケージのミラーを作成し、そのスナップショットを検証した結果です。このスナップショットでは libc6 (¿= 2.28) が不足していることがわかります。

```
$ aptly mirror create -architectures=amd64 -filter='busybox' \
    busybox-buster-mirror-without-dep http://deb.debian.org/debian/ buster main
$ aptly mirror update busybox-buster-mirror-without-dep
$ aptly snapshot create busybox-buster-mirror-without-dep-v1 from mirror busybox-buster-mirror-without-dep
$ aptly snapshot verify busybox-buster-mirror-without-dep-v1
Loading packages...
Verifying...
Missing dependencies (1):
  libc6 (>= 2.28) [amd64]
```

このような場合、足りないパッケージを提供するリモートパッケージリポジトリを apt-line に追加するなどで対応できますが、このリモートパッケージリポジトリの内容が変更される可能性も考えられます。根本的に問題を回避するためには、同じスナップショット内で提供できるようにしたほうがよいでしょう。スナップショットのベースとなったミラーを -filter-with-deps 付加した後更新し、再度スナップショットを作成するか、他のスナップショットで必要とするパッケージが提供されているなら、次で説明する pull サブコマンドを使って、他のスナップショットからパッケージを取り込むこともできます。

```
$ aptly mirror edit -filter-with-deps busybox-buster-mirror-without-dep
$ aptly mirror show busybox-buster-mirror | grep '^Filter With Deps'
Filter With Deps: yes
$ aptly mirror update busybox-buster-mirror
```

他のスナップショットからパッケージを取り込み、新しいスナップショットを作成する

既存のスナップショットに、他のスナップショットで提供されるパッケージを取り込み、新しいスナップショットを作成するには、pull サブコマンドを使います。ベースとするスナップショット busybox-buster-mirror-without-dep-v1 に スナップショット libc-dev-mirror-20200512 で提供される libc6 パッケージを取り込み、スナップショット busybox-buster-mirror-v1 を作成するには、以下のように実行します。

```
$ aptly snapshot pull busybox-buster-mirror-without-dep-v1 libc-dev-mirror-20200512 busybox-buster-mirror-v1 libc6
Dependencies would be pulled into snapshot:
    [busybox-buster-mirror-without-dep-v1]: Snapshot from mirror [busybox-buster-mirror-without-dep]: \
        http://deb.debian.org/debian/ buster
from snapshot:
    [libc-dev-mirror-20200512]: Snapshot from mirror [libc-dev-mirror]: http://deb.debian.org/debian/ buster
and result would be saved as new snapshot busybox-buster-mirror-v1.
Loading packages (8)...
Building indexes...
[+] gcc-8-base_8.3.0-6_amd64 added
[+] libc6_2.28-10_amd64 added
[+] libgcc1_1:8.3.0-6_amd64 added

Snapshot busybox-buster-mirror-v1 successfully created.
You can run 'aptly publish snapshot busybox-buster-mirror-v1' to publish snapshot as Debian repository.
```

新しく作成されたスナップショット busybox-buster-mirror-v1 を確認すると、libc6 パッケージと libc6 に依存するパッケージが取り込まれ、verify サブオプションの結果も問題ないことがわかります。

```
$ aptly snapshot show -with-packages busybox-buster-mirror-v1
Name: busybox-buster-mirror-v1
Created At: 2020-05-15 18:13:45 UTC
Description: Pulled into 'busybox-buster-mirror-without-dep-v1' with 'libc-dev-mirror-20200512' as source, \
        pull request was: 'libc6'
Number of packages: 5
Sources:
  busybox-buster-mirror-without-dep-v1 [snapshot]
  libc-dev-mirror-20200512 [snapshot]
Packages:
  busybox_1:1.30.1-4_amd64
  busybox-static_1:1.30.1-4_amd64
  gcc-8-base_8.3.0-6_amd64
  libc6_2.28-10_amd64
  libgcc1_1:8.3.0-6_amd64
$ aptly snapshot verify busybox-buster-mirror-v1
Loading packages...
Verifying...
All dependencies are satisfied.
```

スナップショットの削除

スナップショットを削除する場合、drop サブコマンドを使用します。削除するスナップショットが他のスナップショットでマージされている場合には削除されません。-force オプションで強制的に削除もできますが、リポジトリの整合性がなくなる可能性があるので、マージ先のスナップショットを削除してから、削除するようにします。下記の実行例では、スナップショット my-repo-20200512 は スナップショット my-product-release-20200512 に依存されているため、削除できません。

```
$ aptly snapshot drop my-repo-20200512
Snapshot ‘my-repo-20200512‘ was used as a source in following snapshots:
 * [my-product-release-20200512]: Merged from sources: ’my-repo-20200512’, ’busybox-buster-mirror-20200512’
ERROR: won’t delete snapshot that was used as source for other snapshots, use -force to override
```

もしスナップショット my-repo-20200512 を削除したい場合には、依存されている my-product-release-20200512 を先に削除する必要があります。

```
$ aptly snapshot drop my-product-release-20200512
Snapshot ‘my-product-release-20200512‘ has been dropped.
$ aptly snapshot drop my-repo-20200512
Snapshot ‘my-repo-20200512‘ has been dropped.
```

スナップショットに関する他の機能

- diff
  スナップショット間の差分を出力します。
- search
  指定したスナップショットで提供されているパッケージを検索します。
- rename
  スナップショット名を変更します。

3.7 リポジトリの公開 (パブリッシュ)

これまでローカルミラーやローカルリポジトリ、スナップショットについて説明してきましたが、これまでの状態ではまだ apt-line として参照できるパッケージリポジトリにはなっていないため、apt コマンド等でパッケージのダウロードやインストールはできません。publish コマンドを用いてパッケージリポジトリを公開する必要があります。

公開するパッケージリポジトリはスナップショットとローカルパッケージリポジトリから作成できますが、後者は推奨されません。この理由として、ローカルパッケージリポジトリはスナップショットとは異なり、内容が変更される可能性があるためです。以下ではスナップショットをパッケージリポジトリとして公開する方法を説明します。

3.7.1 スナップショットからパッケージリポジトリを作成する

スナップショット my-product-release-20200512 をパッケージリポジトリとして公開する場合、snapshot サブコマンドにスナップショット名を指定して実行します。実行すると GnuPG 署名するにパスフレーズを要求されます。

```
$ aptly publish snapshot my-product-release-20200512
Loading packages...
Generating metadata files and linking package files...
Finalizing metadata files...
Signing file 'Release' with gpg, please enter your passphrase when prompted:

次のユーザの秘密鍵のロックを解除するには
パスフレーズがいります:"aptly <aptly@example.com>"
3072 ビット RSA 鍵, ID 08B70EFA 作成日付は 2020-05-10

Clearsigning file 'Release' with gpg, please enter your passphrase when prompted:

次のユーザの秘密鍵のロックを解除するには
パスフレーズがいります:"aptly <aptly@example.com>"
3072 ビット RSA 鍵, ID 08B70EFA 作成日付は 2020-05-10

Snapshot my-product-release-20200512 has been successfully published.
Please setup your webserver to serve directory '/home/aptly/.aptly/public' with autoindexing.
Now you can add following line to apt sources:
  deb http://your-server/ buster main
  deb-src http://your-server/ buster main
Don't forget to add your GPG key to apt with apt-key.

You can also use `aptly serve` to publish your repositories over HTTP quickly.
```

パッケージリポジトリは $(HOME)/.aptly/public 以下に作成されます。対象ディレクトリを apt-line で参照できるように設定すると、apt コマンド等からパッケージを取得できるようになります。また上記出力にあるように apt-line で設定する内容をディストリビューション名が buster になっています。これは スナップショット my-product-release-20200512 のベースが Debian ミラーの buster ディストリビューションだったためです。ディストリビューション名を変更したい場合には、-distribution オプションを使います。

```
$ aptly publish snapshot -distribution="my-product" my-product-release-20200512
<省略>

Now you can add following line to apt sources:
  deb http://your-server/ my-product main
  deb-src http://your-server/ my-product main
```

3.7.2 パッケージリポジトリの更新

パッケージリポジトリ内容を、新しく作成したスナップショットに更新したい場合、switch サブコマンドを使います。以下では、zlib1g パッケージを含んだスナップショットをマージした スナップショット my-product-release-20200513 に更新しています。

```
$ aptly mirror drop zlib-buster-mirror
$ aptly mirror create -architectures=amd64 -filter='zlib1g' -filter-with-deps zlib-buster-mirror \
      http://deb.debian.org/debian/ buster main
$ aptly mirror update zlib-buster-mirror
$ aptly snapshot create zlib-buster-mirror-20200512 from mirror zlib-buster-mirror
$ aptly snapshot merge my-product-release-20200513 my-repo-20200512 busybox-buster-mirror-20200512 zlib-buster-mirror-20200512
$ aptly snapshot diff my-product-release-20200512 my-product-release-20200513
  Arch   | Package | Version in A | Version in B
+ amd64  | zlib1g  | -            | 1:1.2.11.dfsg-1
$ aptly publish switch my-product my-product-release-20200513
Loading packages...
Generating metadata files and linking package files...
Finalizing metadata files...
Signing file 'Release' with gpg, please enter your passphrase when prompted:

<省略>
```

3.7.3 公開されているパッケージリポジトリの確認

公開されているパッケージリポジトリを確認するには list サブコマンドを使います。これにより公開されているパッケージリポジトリがどのスナップショット・ローカルパッケージリポジトリに依存しているのか確認できます。

```
$ aptly publish list
Published repositories:
  * ./buster [amd64, source] publishes {main: [my-product-release-20200512]: Merged from sources: \
      'my-repo-20200512', 'busybox-buster-mirror-20200512'}
  * ./my-product [amd64, source] publishes {main: [my-product-release-20200513]: Merged from sources: \
      'my-repo-20200512', 'busybox-buster-mirror-20200512', 'zlib-buster-mirror-20200512'}
```

パッケージリポジトリでサポートしているアーキテクチャなども確認したい場合には show サブコマンドを使います。

```
$ aptly publish show my-product
Prefix: .
Distribution: my-product
Architectures: amd64 source
Sources:
  main: my-product-release-20200513 [snapshot]
```

3.7.4 パッケージリポジトリの削除

パッケージリポジトリを削除するには drop サブコマンドを使います。実行するとパッケージリポジトリ出力先ディレクトリも削除されます。

```
$ aptly publish drop buster
$ aptly publish list
Published repositories:
  * ./my-product [amd64, source] publishes {main: [my-product-release-20200513]: Merged from sources: \
        'my-repo-20200512', 'busybox-buster-mirror-20200512', 'zlib-buster-mirror-20200512'}
```

3.7.5 パッケージリポジトリへのアクセス

パブリッシュしたパッケージリポジトリで提供されるパッケージを利用するには、以下の 3 つの方法があります。

1. File プロトコルを使用して、.aptly/public/ を apt-line に追加する。
2. aptly で提供される http サーバーを立ち上げ、http 経由で取得する。
3. 別途 HTTP サーバーを立ち上げ、.aptly/public/を参照するよう設定し、http 経由で取得する。

各々の方法を以下に説明します。

File プロトコルを使う

File プロトコルを使う場合には、apt-line に パッケージリポジトリを指定します。この場合、パッケージリポジトリを提供しているマシン以外からはアクセスできない点に注意が必要です。

```
$ sudo sh -c "echo deb file:///home/aptly/.aptly/public/ my-product main" >> /etc/apt/sources.list.d/my-product.list
$ sudo apt update
```

aptly で提供される http サーバー使う

aptly で提供される http サーバーを使うには、aptly serve コマンドを使います。実行すると パッケージリポジトリへ 8080 番ポート経由でアクセスできるようになります。

```
$ aptly serve
Serving published repositories, recommended apt sources list:

# ./my-product [amd64, source] publishes {main: [my-product-release-20200513]: Merged from sources: 'my-repo-20200512', \
        'busybox-buster-mirror-20200512', 'zlib-buster-mirror-20200512'}
deb http://ryzen7:8080/ my-product main
deb-src http://ryzen7:8080/ my-product main

Starting web server at: :8080 (press Ctrl+C to quit)...
```

その後アクセスするマシンに apt-line を追加することで、パッケージリポジトリにアクセスできるようになります。

```
$ sudo sh -c "echo deb http://ryzen7:8080/ my-product main" > /etc/apt/sources.list.d/my-product.list
$ sudo apt update
```

アクセスするためのポートやホスト名を指定したい場合には -listen オプションを使います。ローカルループバックアドレス 127.0.0.1 の 8888 番ポートとして起動したい場合には以下のように実行します。

```
$ aptly serv -listen=127.0.0.1:8888
```

別途 パッケージリポジトリ用 http サーバーを立ち上げる

nginx を例に http サーバーの設定方法を説明します。nginx をインストールし、/etc/nginx/sites-available/apt を作成します。

```
$ sudo apt-get install nginx
$ sudo vi /etc/nginx/sites-available/apt
```

```
server {
    listen 80;
    listen [::]:80;

    server_name aptly.exaple.com;
    root /var/www-apt;
    allow all;
    autoindex on;

    # Full access for everybody for the stable debian repo
    location /public {
        root /home/aptly/.aptly;
        allow all;
    }

    # Allow access to the top level to be able to download the GPG key
    location / {
        allow all;
    }
}
```

/etc/nginx/sites-enabled/apt にシンボリックリンクを作成し、nginx を再起動します。

```
$ sudo ln -s /etc/nginx/sites-available/apt /etc/nginx/sites-enabled/apt
$ sudo systemctl restart nginx
```

その後アクセスするマシンに apt-line を追加することで、パッケージリポジトリにアクセスできるようになります。

```
$ sudo sh -c "echo deb http://ryzen/public/ my-product main" > /etc/apt/sources.list.d/my-product.list
$ sudo apt update
```

3.8 REST API

aptly は REST API が提供されており、API 用サーバーを立ち上げることで利用できるようになります。API サーバーの起動は aptly api コマンドの serve サブオプションを実行します。

```
$ aptly api serve
Starting web server at: :8080 (press Ctrl+C to quit)...
[GIN-debug] [WARNING] Now Gin requires Go 1.6 or later and Go 1.7 will be required soon.
...
```

起動した後は、サーバーに対して API を実行できるようになります。repo、snapshot、publish、package、graph に関する API が提供されており、mirror や db に対する API はまだ提供されていません。API の詳細は Web サイトの API (`https://www.aptly.info/doc/api/`) に記載されています。ここでは API の使い方をいくつか紹介します。

- ローカルパッケージリポジトリで提供されているパッケージ一覧を取得する。

```
$ curl http://localhost:8080/api/repos/my-repo/packages
["Pamd64 lz4 1.9.2-2 76bbfff77a824848","Pamd64 lz4-dbgsym 1.9.2-2 e71a271edd2c351","Psource lz4 1.9.2-2 91576aff056e5141",\
        "Pall liblz4-tool 1.9.2-2 8db5a921d2f58813","Pamd64 liblz4-1 1.9.2-2 30f3b73d9c877f6b",\
        "Pamd64 liblz4-1-dbgsym 1.9.2-2 7a9c4cd5844174f8","Pamd64 liblz4-dev 1.9.2-2 9f649d10440c4c7f"]
```

- スナップショットに関する情報を取得する。

```
$ curl http://localhost:8080/api/snapshots
[{"Name":"busybox-buster-mirror-20200512","CreatedAt":"2020-05-15T19:29:25.376281727Z",\
        "Description":"Snapshot from mirror [busybox-buster-mirror]: http://deb.debian.org/debian/ buster","Origin":"Debian",\
        "NotAutomatic":"","ButAutomaticUpgrades":""},
        {"Name":"busybox-buster-mirror-without-dep-v1","CreatedAt":"2020-05-15T18:09:04.530501686Z",\
        "Description":"Snapshot from mirror [busybox-buster-mirror-without-dep]: http://deb.debian.org/debian/ buster",
        "Origin":"Debian","NotAutomatic":"","ButAutomaticUpgrades":""},
        {"Name":"my-product-release-20200512","CreatedAt":"2020-05-15T21:51:09.772154566Z",\
        "Description":"Merged from sources: 'my-repo-20200512', 'busybox-buster-mirror-20200512'","Origin":"",
        "NotAutomatic":"", "ButAutomaticUpgrades":""}, \
        {"Name":"my-product-release-20200513","CreatedAt":"2020-05-15T19:38:33.373419548Z", \
        "Description":"Merged from sources: 'my-repo-20200512', 'busybox-buster-mirror-20200512',
        'zlib-buster-mirror-20200512'", "Origin":"","NotAutomatic":"","ButAutomaticUpgrades":""}, \
        {"Name":"my-repo-2* Connection #0 to host localhost left intact 0200512","CreatedAt":"2020-05-15T19:29:26.185504839Z", \
        "Description":"Snapshot from local repo [my-repo]","Origin":"","NotAutomatic":"","ButAutomaticUpgrades":""}, \
        {"Name":"zlib-buster-mirror-20200512","CreatedAt":"2020-05-15T19:29:25.5108828555Z", \
        "Description":"Snapshot from mirror [zlib-buster-mirror]: http://deb.debian.org/debian/ buster","Origin":"Debian"," \
        NotAutomatic":"","ButAutomaticUpgrades":""}]
```

- ローカルパッケージリポジトリ my-repo のスナップショット を my-repo-devel として作成する

```
$ curl -X POST -H 'Content-Type: application/json' --data '{"Name":"my-repo-devel"}' \
        http://localhost:8080/api/repos/my-repo/snapshots
{"Name":"my-repo-devel","CreatedAt":"2020-05-16T09:44:46.985707235+09:00", \
        "Description":"Snapshot from local repo [my-repo]","Origin":"","NotAutomatic":"","ButAutomaticUpgrades":""}
```

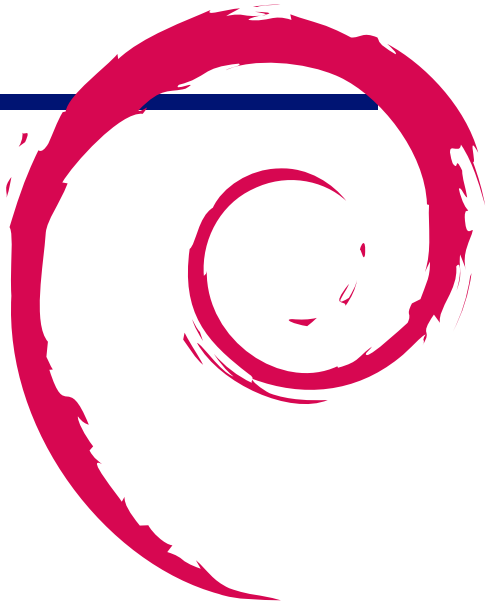
3.8.1 REST API を扱うソフトウェア

REST API は直接扱いづらいため、Python や Ruby などのプログラミング言語で操作したいと思う人がほとんどだと思います。aptly を扱うツールや API ラッパーを紹介します。

- Python
  - aptly-api-client / `https://github.com/gopythongo/aptly-api-client`
  - python-aptly / `https://github.com/tcpcloud/python-aptly`
  - , pyaptly / `https://github.com/adfinis-sygroup/pyaptly`
- Ruby
  - aptly_cli / `https://github.com/sepulworld/aptly_cli`
  - aptly-simple / `https://github.com/serge-name/aptly-simple`
  - aptlier / `https://github.com/3ofcoins/aptlier`
- shell
  - nitrux-repository-util / `(https://github.com/Nitrux/nitrux-repository-util`

3.9 まとめ

Debian パッケージリポジトリの統合ツールである aptly の使い方を紹介しました。API の機能はまだ足りてない部分も多く、使いづらいところはありますが、dpkg-dev や apt-ftparchive 使いながら CI/CD 環境を構築されている方はもちろん、会社のインフラ整備や Debian/Ubuntu をつかった製品開発をされている方や Debian パッケージメンテナにも有用なツールです。これを機会に aptly へのきりかえを検討してみませんか。



4 メモ

Debian 勉強会資料

2020 年 5 月 16 日　　初版第 1 刷発行

東京エリア Debian 勉強会 （編集・印刷・発行）