pulpdist Documentation

Release 0.1.1

Nick Coghlan

November 11, 2016

Contents

PulpDist is a set of Pulp plugins and an associated Django application that together allow a network of Pulp servers to be used as a filtered mirroring network with robust access control mechanisms.

The project is in a usable state for the specific task of filtered mirroring with rsync, but still has quite a few rough edges. In particular, it still relies on the alpha version of the plugin APIs in Pulp v1 rather than using the updated version that are coming in Pulp v2.

Contents:

CHAPTER 1

PulpDist Architecture

PulpDist uses the Pulp repository management utilities to manage arbitrary directory trees (note that the underlying Pulp features it uses are under active development, so it still has quite a long way to go before it can be considered ready for production use beyond a very narrow set of use cases).

Each site in the mirror network has its own Pulp server. These servers handle the actual data transfers involved in the mirror network using a number of custom *Pulp plugins*.

The status of these transfers can then be monitored using a central *web application* which uses OAuth to retrieve information from each Pulp server in the network. An instance of the web app may also be run at each site with a Pulp mirror server for local status monitoring.

CHAPTER 2

PulpDist Web Application

The PulpDist web application is a Django-based web service that can be set up to monitor a network of Pulp servers.

Current iterations focus primarily on status monitoring, leaving configuration tasks to command line scripting tools. Longer term, some configuration tasks may be permitted through forms in the web application.

This page focuses on deployment of a single PulpDist web app instance with a colocated Pulp server. Other configurations are of course possible - the two communicate solely through the Pulp REST API (Note: the PulpDist web app does not yet cache results received from the Pulp server, nor has it been optimised to make full use of server side batch queries and filtering, so expect poor performance from the current version if the two aren't running on the same web server and abysmal performance if they aren't at least on the same LAN).

2.1 Deployment

The first step in deploying a standard PulpDist web application with a colocated Pulp server is:

```
$ sudo yum install pulpdist-plugins pulpdist-httpd
```

(Note: prebuilt RPMs are not yet available from the public repo. See *Building the PulpDist RPMs*)

If you're using a custom Django settings file, then package that as an RPM with a dependency on `pulpdist-django` and install the custom RPM instead of `pulpdist-httpd`. More on that below.

After installation, a few configuration settings need to be adjusted.

1. Update `/etc/pulp/pulp.conf` in accordance with the Pulp Installation Guide, including:
   - setting up OAuth authentication
   - setting up LDAP user authentication
   - changing the default password
   - Ensure the file is not world-readable (as both the OAuth key and the admin password allow full access to the Pulp server)

   There are various certificate related settings in this file that can be safely ignored for current PulpDist installations. They relate to the use of access controlled repositories, which PulpDist doesn't currently support. Setting up the AMQP messaging support is also an option, but not required.
2. Update `/etc/pulpdist/site.conf` in accordance with the embedded comments. Notably:
   - Enter the initial list of system administrators
   - Set the passphrase for encrypted database fields

- Generate and enter a private Django secret key (see below)

3. Update `/etc/httpd/conf.d/pulpdist.conf` to set the Kerberos domain correctly (and, optionally, add a keytab reference for single-sign-on support). Note that `pulpdist-httpd` makes a number of assumptions that are only valid when using Kerberos for authentication - if you want to do something else (e.g. use Django's native authentication), install `pulpdist-django` instead (either directly or an RPM dependency) and include `pulpdist.django_app` as an installed application in a custom Django site definition.

4. Initialise and start the Pulp server. This will start both the MongoDB data store as well the Apache web server (note that running Pulp's `init.d` scripts directly doesn't appear to work correctly):

```
$ service pulp-server init
$ service pulp-server start
```

5. Update `/etc/pulp/admin/admin.conf` to replace `localhost.localdomain` with the fully qualified domain of the server

6. Optionally, set up a separate administrator account for the Pulp server and restrict the default admin account to read-only access (currently used via the web UI over OAuth)

```
# Use the default account to add a new administrator
pulp-admin auth login --username admin
pulp-admin user create --username ncoghlan --name "Nick Coghlan" --ldap
pulp-admin role add --role super-users --user ncoghlan

# Use the new administrator account to restrict the default account
pulp-admin auth login --username ncoghlan
pulp-admin role create --role read-only
pulp-admin permission grant --resource / --role read-only -o read
pulp-admin role add --role read-only --user admin
pulp-admin role remove --role super-users --user admin

# Check the permissions have been updated appropriately
pulp-admin permission show --resource /
```

7. Log in to the PulpDist web application as one of the system administrators configured in Step 2. Click the "Site Admin" link, then use the Django admin UI to add a reference to the colocated Pulp server. The fields are as follows:

   - Pulp site: name used in the user interface for this server
   - Hostname: fully qualified hostname for this server (will be checked by SSL)
   - Oauth key: the Pulp OAuth key configured in Step 1
   - Oauth secret: the Pulp OAuth secret configured in Step 1

The following command can be used to generate a fresh Django secret key:

```
python -c 'from random import choice; print("".join([choice("abcdefghijklmnopqrstuvwxyz0123456789!@#$
```

2.2 Django Admin CLI

The command line interface for Django administration of the site is available as:

```
$ sudo python -m pulpdist.manage_site --help
```

Refer to the Django documentation for details of what this command supports (like the default `manage.py`, `pulpdist.manage_site` is a thin convenience wrapper around `django-admin`).

2.3 REST API

The Rest API is relative to the assigned base URL for the `django_pulpdist` application (`/pulpdist/` by default):

```
api/  # API root

api/servers/                            # All configured Pulp servers
api/servers/<server_id>/                # Specific server
api/servers/<server_id>/repos           # -> api/repos/<server_id>
api/servers/<server_id>/content_types   # -> api/content_types/<server_id>
api/servers/<server_id>/distributors    # -> api/distributors/<server_id>
api/servers/<server_id>/importers       # -> api/importers/<server_id>

api/repos                                       # All repos on all servers
api/repos/<server_id>/                          # All repos on server
api/repos/<server_id>/<repo_id>/                # Specific repo
api/repos/<server_id>/<repo_id>/importer        # Importer config & status
api/repos/<server_id>/<repo_id>/distributors    # Assigned distributors
api/repos/<server_id>/<repo_id>/sync_history    # Past sync operations

api/content_types                           # All content types on all servers
api/content_types/<server_id>/              # All content types on server
api/content_types/<server_id>/<type_id>/    # Specific content type definition

api/distributors                             # All distributors on all servers
api/distributors/<server_id>/                # All distributors on server
api/distributors/<server_id>/<plugin_id>/    # Specific distributor definition

api/importers                               # All importers on all servers
api/importers/<server_id>/                  # All importers on server
api/importers/<server_id>/<plugin_id>/      # Specific importer definition
```

CHAPTER 3

PulpDist Repository Management Client

The 1.x series of the upstream Pulp project does not provide a management client for repositories that use the preliminary support for the v2 repo plugin model. Accordingly, PulpDist comes with a command line interface for working with these repositories.

3.1 Invoking the Client

At this stage, there is no separately installed executable script to manage repositories. Instead, a feature of the CPython interpreter is used to invoke the appropriate module as a command line script:

```
$ python -m pulpdist.manage_repos --help
```

Before using the command line client to manage a Pulp server, it is necessary to create the login credentials for the Pulp server. PulpDist supports three mechanisms for that:

- Kerberos tickets (default, but requires a patched version of Pulp)
- Pulp's native credential caching mechanism

The former just requires an active Kerberos ticket that will be recognised by the server. The credentials for the latter can be acquired with the upstream `pulp-admin` client (entering the appropriate password when prompted):

```
$ pulp-admin --host <HOST> auth login --username <USER>
```

To request use of these credentials (rather than a Kerberos ticket), pass the `--auth pulp` option to the client.

Like `pulp-admin` the PulpDist repo management client defaults to using the fully qualified domain name of the current host as the target server. This can be overridden by passing a different hostname via the `--host` option.

3.1.1 Synchronisation Management Commands

- `sync`: Request immediate synchronisation of repositories
- `enable`: Configure repositories to respond to sync requests
- `disable`: Configure repositories to ignore sync requests
- `cron_sync`: See *Scheduling sync operations with cron*

3.1.2 Repository Status Queries

- `list`: Display id and name for repositories
- `info`: Display details for repositories
- `status`: Display repository synchronisation status
- `history`: Display repository synchronisation history
- `log`: Display most recent synchronisation log
- `stats`: Display most recent synchronisation statistics

3.1.3 Repository Management Commands

- `init`: Create or update repositories on the server
- `delete`: Remove repositories from the server
- `validate`: Check the validity of a repository definition file
- `export`: Create a site definition file from an existing repository (Not Yet Implemented)

3.2 Limiting commands to selected repositories

The `--repo` option accepts repository identifiers and allows a command to run against the named repository. It may be supplied multiple times to run a command against multiple repositories.

The `--mirror` option accepts local mirror identifiers and allows a command to run against the named local mirror. It may be supplied multiple times to run a command against multiple repositories.

The `--tree` option accepts remote tree identifiers and allows a command to run against repositories that were configured from a site configuration file to sync with a particular remote tree. It may be supplied multiple times to run a command against mirrors of multiple trees.

The `--source` option accepts remote source identifiers and allows a command to run against repositories that were configured from a site configuration file to sync with a tree from that remote source. It may be supplied multiple times to run a command against repositories from multiple sources.

The `--server` option accepts remote server identifiers and allows a command to run against repositories that were configured from a site configuration file to sync with a tree from that remote server. It may be supplied multiple times to run a command against repositories from multiple servers.

The `--site` option accepts site identifiers and allows a command to run against repositories that were configured from a site configuration file based on the specified site settings. It may be supplied multiple times to run a command against multiple local "sites". This option is only useful if repositories are configured against more than one site on the specified Pulp server.

If no specific repositories are identified, most commands default to affecting every repository defined on the server, or, if the command accepts a configuration file, every repository named in the file.

By default, the command line client uses the metadata stored on the server to identify the available repositories. If this metadata is incomplete or invalid, the `--ignoremeta` option can be passed before the command to be executed. In this mode, the Pulp server will be treated as containing only raw repo definitions, allowing listing and manipulation of repos that would otherwise be ignored (due to the fact they aren't recorded in the stored metadata).

3.3 Scheduling sync operations

3.3.1 Scheduling sync operations with Pulp

Eventually, PulpDist will use the native Pulp task scheduler for sync operations. However, this is not yet supported by Pulp for plugin based repositories (such as those used by PulpDist).

3.3.2 Scheduling sync operations with cron

As the versions of Pulp currently supported by PulpDist do not provide native sync scheduling support for plugin based repositories), PulpDist offers a simple alternative mechanism based on cron (or any similar tool that can be used to periodically execute a Python script).

The relevant command is:

```
python -m pulpdist.manage_repos cron_sync
```

This tool is designed to be run once per hour (if a previous instance for the same Pulp host is still running, the new instance will immediately exit). For more immediate synchronisation, the `sync` command should be invoked directly.

The command first retrieves the list of repository definitions from the Pulp server and queries each one for a `["notes"]["pulpdist"]["sync_hours"]` setting in the metadata.

If sync operations on the repository are currently enabled, the repository does not already have a sync operation in progress, the `sync_hours` setting is found and is non-zero,and the current time (in hours) relative to midnight is a multiple of the `sync_hours` setting, then a new thread is spawned to request immediate synchronisation of the repository through the Pulp REST API.

Otherwise, the repository is ignored until the next check for new sync operations.

As long as any sync operations are still in progress, the client will periodically query the server for updated information, scheduling sync operations as appropriate.

As soon as all sync operations are complete (regardless of success or failure), the client will terminate.

The following options can be set to control the sync operation:

- `--threads`: maximum number of concurrent sync operations (default: 4)
- `--day`: rsync bandwidth limit to apply during the day (6 am - 6 pm)
- `--night`: rsync bandwidth limit to apply at night (6 pm - 6 am)

By default, no bandwidth limits are applied.

Note: Support for bandwidth limiting is not yet implemented

3.4 The repository definition file format

The `init` and `validate` commands provided by `manage_repos` both require a repository definition file. The `export` command generates a respository definition file describing the server contents.

These are JSON files that specify the information needed to create the repositories on the Pulp server, and appropriately configure the associated importer plugins. See *PulpDist Site Configuration* for more details.

3.5 PulpDist metadata in Pulp

When PulpDist repositories are initialised from a site configuration file, a `pulpdist-meta` repo is automatically created to record the full contents of the original site configuration. This information is stored in the "notes" field for that repository.

Additional information is also recorded in the `notes` field of each created Pulp repo to support some features of the PulpDist command line client. This additional metadata is stored in the format:

- `pulpdist`: Top-level mapping entry to identify pulpdist related metadata
  - `sync_hours`: The remote tree `sync_hours` setting (if any)
  - `site_id`: The site settings used to configure this repo
  - `mirror_id`: The local mirror name for this repo
  - `tree_id`: The remote tree mirrored by this repo
  - `source_id`: The remote source for this tree
  - `server_id`: The remote server for this tree

The `repo_id` of the associated Pulp repository is built from the `mirror_id` and `site_id` of the local mirror definition.

CHAPTER 4

PulpDist Custom Plugins

The custom Pulp plugins for PulpDist use `rsync` under the hood to perform efficient updates over the network. They currently use the `rsync` CLI directly, but may eventually move to a more programmatic API based on `librsync`.

4.1 Sync Operation Results

Each of the PulpDist sync plugins may report the following results:

- `SYNC_UP_TO_DATE`: the local copy was up to date, no changes were made.
- `SYNC_COMPLETED`: upstream changes were found and applied locally
- `SYNC_PARTIAL`: upstream changes were found, but the attempt to apply them locally failed to incorporate some changes (see log output for details)
- `SYNC_FAILED`: sync completely failed (e.g. upstream could not be reached)
- `SYNC_DISABLED`: the plugin has been set to ignore sync requests

These statuses may also be reported with `_DRY_RUN` appended to indicate that a sync operation was executed with rsync configured to avoid actually transferring any files (some temporary local copies of small metadata files may still be made in order to determine the details of the dry run operation).

4.2 Simple Tree Sync

A simple tree sync is a convenient way to define and schedule an rsync task. Configuration options for this plugin are:

- `tree_name`: A short text name for the tree
- `remote_server`: The host name or IPv4 address of the source rsync server
- `remote_path`: The path to read from on the remote server
- `local_path`: The local destination path for the received tree
- `exclude_from_sync`: A list of rsync `--exclude` patterns applied to the tree synchronisation operation. Defaults to no exclusions.
- `sync_filters`: A list of rsync `--filter` patterns applied to the tree synchronisation operation. Defaults to no filtering.
- `bandwidth_limit`: If provided and not zero, passed to rsync as `--bwlimit` to limit the amount of bandwidth used by the operation.

- `old_remote_daemon`: If provided and true, passes `--no-implied-dirs` to rsync to run it in a mode compatible with older versions of the rsync daemon.
- `rsync_port`: If provided and not zero, passed to rsync as `--port` to allow connections to a remote daemon that isn't running on the default port.
- `enabled`: If provided and true, actually performs a sync operation when invoked by Pulp. Defaults to ignoring sync requests.
- `dry_run_only`: If provided and true, passes `-n` to rsync to run it in "dry run" mode (i.e. no actual file transfers will take place).

Adding files named `PROTECTED` to directories at downstream sites will keep the plugin from overwriting (or otherwise altering) them.

4.3 Versioned Tree Sync

A versioned tree sync works like a series of simple tree syncs. It is intended for directories containing multiple versions of a single tree, where each tree may change over time. The trees are synchronised in separate operations, but the sync process attempts to create hard links between the trees whenever possible.

In addition to all of the simple tree sync configuration options, the versioned tree sync has the following additional options that are used to build the list of individual subtrees to be synchronised:

- `listing_pattern`: An rsync `--include` pattern identifying the subtrees to synchronise. Defaults to all subdirectories of `remote_path`.
- `exclude_from_listing`: A list of rsync `--exclude` patterns applied to the subtree listing operation. Defaults to no filtering.
- `listing_filters`: A list of rsync `--filter` patterns applied to the subtree listing operation. Defaults to no filtering.
- `delete_old_dirs`: If provided and true, removes local subdirectories that are no longer present on the source server. By default, local subdirectories are retained until explicitly deleted by a system administrator. Adding a `PROTECTED` file will also ensure a directory is not deleted automatically.

  To avoid data loss due to network and remote storage glitches, the plugin treats the case where absolutely no relevant remote directories are found as an error and never deletes local directories in that situation. Similarly, if the overall job will be reported as `SYNC_FAILED` or `SYNC_PARTIAL`, then no local directories will be removed.

The versioned tree sync also reproduces locally any upstream symlinks that match the listing pattern and point to destinations that exist on the local server after the sync operation is otherwise complete.

4.4 Snapshot Tree Sync

A snapshot tree sync works like a versioned tree sync, but versions are never updated after their initial release. "STATUS" marker files in the root directory of each tree are used to indicate when a tree is completed. Each tree is synchronised only if the remote tree includes a STATUS file containing the text `FINISHED`, and there is no existing local tree that contains such a file.

The big advantage of snapshot tree syncs is that once a tree has been marked as complete locally, it never needs to be checked against the upstream site again.

In addition to all of the versioned tree sync configuration options, the snapshot tree sync has the following additional options that allow special treatment for the most recent snapshot (as determined by the timestamps in the remote directory listing):

- `latest_link_name`: If provided and not `None`, a local symbolic link is created with this name that points to the most recent snapshot after each sync operation. By default, no symbolic link is created.
- `sync_latest_only`: If provided and true, only the most recent remote snapshot will be mirrored locally. By default, all remote snapshots are mirrored.

The snapshot tree sync also modifies the behaviour of the `delete_old_dirs` setting: the most recently synchronised snapshot will *never* be deleted automatically, even after it has been deleted remotely. This is useful when mirroring snapshots generated by an automatic build process that only retains a limited number of build attempts, regardless of whether or not the build succeeded. Retaining the most recent snapshot ensures that there will always be a version of the tree available for local use, the "latest snapshot" symlink (if defined) will remain valid, and future sync operations will have a base to use for hardlinking previously synchronised files.

4.5 Snapshot Delta Sync

Delta syncs actually require an upstream Pulp server (rather than just an rsync daemon) and use a chain of 3 custom Pulp plugins.

At the upstream site, rsync is run in batch mode to generate delta files to update from the previous version of the tree to the latest snapshot.

These delta files are then published for retrieval by the downstream servers.

The downstream servers first check if a delta file is available that is applicable to the most recent version of the tree they have completed locally. If it exists, they download and apply it. Otherwise, they fall back on doing a full synchronisation via rsync (i.e. the same process as an ordinary snapshot tree sync)

CHAPTER 5

PulpDist Site Configuration

The PulpDist site configuration file is used to describe the full set of mirroring tasks to be carried out at a site. It is designed to allow data source definitions to be shared amongst multiple sites, and even to define the jobs for multiple sites within a single file.

5.1 Site Configuration Components

A site config file consists of a top-level JSON mapping, defining the following attributes:

- `LOCAL_MIRRORS`: A sequence of *local mirror definitions*.
- `REMOTE_TREES`: A sequence of *remote tree definitions*.
- `REMOTE_SOURCES`: A sequence of *remote source definitions*.
- `REMOTE_SERVERS`: A sequence of *remote server definitions*.
- `SITE_SETTINGS`: A sequence of *site definitions*.
- `RAW_REPOS`: A sequence of *raw repo definitions*.

The general concept is that:

- each local tree mirrors a particular remote tree
- each remote tree is provided by a particular remote source
- each remote source is provided by a particular remote server
- these settings are combined with the appropriate site settings to create raw repo definitions that are uploaded to the server
- details of the original settings are stored in the raw repo metadata, allowing them to be exported again if necessary
- additional raw repos can be defined and are passed directly to the Pulp server

The current format doesn't allow for the definition of alternative sources for a given tree, but this capability may be added in the future.

5.1.1 Local Mirror Definitions

A local mirror is a PulpDist managed mirror (possibly filtered) of a remote tree.

A local mirror definition is a mapping with the following attributes:

- `mirror_id`: locally unique ID (alphanumeric characters and hyphens only)
- `tree_id`: the ID of the remote tree that this local tree mirrors
- `site_id`: the ID of the site settings used for this tree (default: `"default"`)
- `name`: human readable name of local tree (default: same as remote tree)
- `description`: description of local tree (default: same as remote tree)
- `mirror_path`: final path segment for this tree (default: same as tree_path)
- `enabled`: whether the tree starts with sync enabled (default: false)
- `dry_run_only`: whether the tree starts in dry run mode (default: false)
- `exclude_from_sync`: rsync wildcard patterns to ignore when retrieving files (optional)
- `sync_filters`: additional rsync filters applied when retrieving files (optional)
- `notes`: additional notes to store in the Pulp repo metadata (optional)

The `exclude_from_sync` and `sync_filters` settings are appended to the default filtering options including in the remote tree definition.

The following additional settings are only valid if the remote tree specifies the use of either `versioned` or `snapshot` as the sync algorithm:

- `delete_old_dirs`: whether local dirs no longer in the remote tree are deleted (default: false)
- `exclude_from_listing`: additional rsync wildcard patterns to ignore when determining which version directories to synchronise (optional)
- `listing_filters`: additional rsync filters applied when determining which version directories to synchronise (optional)

The `exclude_from_listing` and `listing_filters` settings are appended to the default filtering options including in the remote tree definition.

The following additional settings is only valid if the sync algorithm is set to `snapshot`:

- `sync_latest_only`: If provided and true, only the most recent remote snapshot will be mirrored locally. By default, all remote snapshots are mirrored.

5.1.2 Remote Tree Definitions

A remote tree is a file tree available for synchronisation via rsync.

A remote tree definition is a mapping with the following attributes:

- `tree_id`: locally unique ID (alphanumeric characters and hyphens only)
- `source_id`: the ID of the remote source that publishes this tree
- `name`: human readable name of tree
- `description`: description of tree
- `tree_path`: final path segment for this tree (before the tree contents)
- `sync_hours`: used for *Scheduling sync operations with cron*.
- `sync_type`: the tree sync algorithm to use. See below for details.
- `exclude_from_sync`: rsync wildcard patterns to ignore when retrieving files (optional)
- `sync_filters`: additional rsync filters applied when retrieving files (optional)

The currently supported sync algorithms are:

- `simple`: Settings are derived for a *Simple Tree Sync*
- `versioned`: Settings are derived for a *Versioned Tree Sync*
- `snapshot`: Settings are derived for a *Snapshot Tree Sync*

The following additional settings are only valid if the sync algorithm is either `versioned` or `snapshot`:

- `listing_pattern`: rsync wildcard pattern used to determine which directories to synchronise (default: '*')
- `listing_prefix`: alternative mechanism to specify the listing pattern as `listing_prefix + listing_suffix` (where the latter comes from the remote source settings).
- `exclude_from_listing`: rsync wildcard patterns to ignore when determining which directories to synchronise (optional)
- `listing_filters`: rsync filters applied when determining which directories to synchronise (optional)

The following additional setting is only valid if the sync algorithm is set to `snapshot`:

- `latest_link`: the filename used for a symlink that refers to the most recently synchronised snapshot directory. If omitted, indicates that no such symlink should be created.

5.1.3 Remote Source Definitions

A remote source describes common settings for a group of remote trees.

A remote source definition is a mapping with the following attributes:

- `source_id`: locally unique ID (alphanumeric characters and hyphens only)
- `server_id`: the ID of the remote server that publishes these trees
- `name`: human readable name for this group of remote trees
- `remote_path`: shared path prefix for these trees on the remote server
- `listing_suffix`: rsync wildcard pattern to append when a remote tree definition uses the `listing_prefix` setting

5.1.4 Remote Server Definitions

A remote server describes the location of an actual rsync server.

A remote server definition is a mapping with the following attributes:

- `server_id`: locally unique ID (alphanumeric characters and hyphens only)
- `name`: human readable name for this server
- `dns`: DNS name used to access this server
- `old_daemon`: Server runs an old version of rsync (default: False)
- `rsync_port`: Port rsync daemon is listening on (default: rsync default)

5.1.5 Site Definitions

A site definition is a mapping with the following attributes:

- `site_id`: locally unique ID (alphanumeric characters and hyphens only)
- `name`: human readable name for this site
- `storage_prefix`: The shared path prefix for the local data storage area
- `server_prefixes`: mapping from `server_id` values to path segments
- `source_prefixes`: mapping from `source_id` values to path segments
- `exclude_from_listing`: rsync wildcard patterns to ignore by default when determining which version directories to synchronise (if one of these filters matches the wildcard pattern identifying *desired* versions, then that exclusion filter will be omitted from the raw repo definition).
- `exclude_from_sync`: rsync wildcard patterns that are always ignored when creating a raw repo definition (e.g. to exclude standard locations for temporary working files)

5.1.6 Raw Repo Definitions

Raw repo definitions are a low-level interface that corresponds directly with the settings accepted by the underyling calls to the Pulp REST API. They allow direct specification of sync operations at the rsync level without needing to create single use remote tree, source and server definitions.

A raw repo definition is a mapping with the following attributes:

- `repo_id`: Locally unique repo ID (alphanumeric characters and hyphens only)
- `display_name`: Human readable short name for the repository
- `description`: Longer description of the repository contents
- `notes`: Arbitrary notes about the repository as a JSON mapping
- `importer_type_id`: Importer plugin type identifier. See below.
- `importer_config`: JSON mapping with plugin configuration data. See below.

The plugin names in the list below are the exact names that should be used in the `importer_type_id` field for the PulpDist plugins, while the links go to the descriptions of the individual plugins. The options described in those sections are the values that need to be provided in the `importer_config` mapping.

- `simple_tree`: *Simple Tree Sync*
- `versioned_tree`: *Versioned Tree Sync*
- `snapshot_tree`: *Snapshot Tree Sync*

For further information, refer to the documentation for the Pulp Create Repository and Add Importer REST API calls.

5.2 Deriving Raw Repo Definitions from Local Mirror Definitions

Deriving raw repo definitions from local mirror definitions requires that a specific site be nominated. If no site is nominated, or the site settings have no entry for a particular value, then the corresponding settings for the `default` site are used instead.

The local path used in the import configuration is calculated as:

```
storage_prefix/server_prefix/source_prefix/local_tree_path
```

Where:

- `storage_prefix` is taken directly from the site settings
- `server_prefix` is looked up in the server prefixes map. If it is not defined for either the specified site or the default site, then the empty string is used (and the now redundant extra path separator is omitted).
- `source_prefix` is looked up in the source prefixes map. If it is not defined for either the specified site or the default site, then the empty string is used (and the now redundant extra path separator is omitted).
- `local_tree_path` is the `tree_path` setting for the local tree, if it is defined, otherwise it uses the setting for the remote tree.

The remote path used to retrieve a tree is calculated as:

```
rsync://server_dns/source_remote_path/remote_tree_path
```

These values are all taken directly from the appropriate remote server, remote source and remote tree settings, respectively.

The filtering options for the sync process (and, if applicable, the listing process) are determined by inspecting the settings for the local mirror, the remote tree, the local site and the default site. All filtering options given in any of those applications are applied to the underlying rsync command. (The one exception is that any listing exclusion settings that would exclude directories matching the listing pattern for a particular tree are omitted from the remote listing command for that tree).

For the `sync_filters` and `listing_filters` properties, order is preserved and the filters for the local mirror are added to the command line before those for the remote tree.

For the `exclude_from_sync` and `exclude_from_listing` options, order is not preserved. The settings for the local mirror, remote tree, specific site (if any) and default site are merged into a single list in sorted order with any duplicates remove.

Other settings are derived as detailed in the descriptions of the individual setting.

CHAPTER 6

Site Configuration Tutorial

The following file is an example site definition provided in the PulpDist source tree (as `misc/example_site.json`) for demonstration purposes:

```
{
  "SITE_SETTINGS": [
    {
      "site_id": "default",
      "name": "Default Site",
      "storage_prefix": "/var/www/pub",
      "server_prefixes": {
        "demo_server": "sync_demo",
        "other_demo_server": "sync_demo_trees"
      },
      "source_prefixes": {
        "sync_demo": "sync_demo_trees"
      },
      "exclude_from_sync": ["*dull*"],
      "exclude_from_listing": ["*justfortesting*"]
    },
    {
      "site_id": "other",
      "name": "Other Site",
      "storage_prefix": "/var/www/pub/sync_demo"
    }
  ],
  "LOCAL_MIRRORS": [
    {
      "mirror_id": "simple_sync",
      "tree_id": "simple_sync",
      "exclude_from_sync": ["*skip*"],
      "sync_filters": ["exclude_irrelevant/"],
      "notes": {
        "basic": "note",
        "site_custom": {
          "origin": "PulpDist example repository"
        }
      }
    },
    {
      "mirror_id": "versioned_sync",
      "tree_id": "versioned_sync",
      "site_id": "other",
      "exclude_from_sync": ["*skip*"],
```

```
      "sync_filters": ["exclude_dull/"],
      "exclude_from_listing": ["relevant-but*"],
      "notes": {
        "site_custom": {
          "origin": "PulpDist example repository"
        }
      }
    },
    {
      "mirror_id": "snapshot_sync",
      "tree_id": "snapshot_sync",
      "notes": {
        "site_custom": {
          "origin": "PulpDist example repository"
        }
      }
    }
  ],
  "REMOTE_TREES": [
    {
      "tree_id": "simple_sync",
      "name": "Simple Sync Demo",
      "description": "Demonstration of the simple tree sync plugin",
      "tree_path": "simple",
      "sync_type": "simple",
      "sync_hours": 0,
      "source_id": "sync_demo"
    },
    {
      "tree_id": "versioned_sync",
      "name": "Versioned Sync Demo",
      "description": "Demonstration of the versioned tree sync plugin",
      "tree_path": "versioned",
      "sync_type": "versioned",
      "sync_hours": 12,
      "source_id": "sync_demo_other",
      "listing_pattern": "relevant*",
      "exclude_from_sync": ["*skip*"],
      "sync_filters": ["exclude_irrelevant/"]
    },
    {
      "tree_id": "snapshot_sync",
      "name": "Snapshot Sync Demo",
      "description": "Demonstration of the snapshot tree sync plugin",
      "tree_path": "snapshot",
      "sync_type": "snapshot",
      "sync_hours": 1,
      "source_id": "sync_demo",
      "listing_prefix": "re*ev",
      "latest_link": "latest-relevant",
      "exclude_from_listing": ["relevant-but*"],
      "exclude_from_sync": ["*skip*"],
      "sync_filters": ["exclude_irrelevant/", "exclude_dull/"]
    }
  ],
  "REMOTE_SOURCES": [
    {
      "source_id": "sync_demo",
```

```
      "server_id": "demo_server",
      "name": "Sync Demo Trees",
      "remote_path": "demo",
      "listing_suffix": "*"
    },
    {
      "source_id": "sync_demo_other",
      "server_id": "other_demo_server",
      "name": "Other Sync Demo Trees",
      "remote_path": "demo",
      "listing_suffix": "*"
    }
  ],
  "REMOTE_SERVERS": [
    {
      "server_id": "demo_server",
      "name": "Sync Demo Server",
      "dns": "localhost"
    },
    {
      "server_id": "other_demo_server",
      "name": "Other Sync Demo Server",
      "dns": "localhost"
    }
  ],
  "RAW_REPOS": [
    {
      "repo_id": "raw_sync",
      "display_name": "Raw Sync Demo",
      "description": "Demonstration of raw sync configuration in site config",
      "notes": {
        "pulpdist": {
          "sync_hours": 24
        },
        "site_custom": {
          "origin": "PulpDist example repository"
        }
      },
      "importer_type_id": "simple_tree",
      "importer_config": {
        "tree_name": "Raw Simple Tree",
        "remote_server": "localhost",
        "remote_path": "/demo/simple/",
        "local_path": "/var/www/pub/sync_demo_raw/",
        "exclude_from_sync": ["*skip*"],
        "sync_filters": ["exclude_irrelevant/", "exclude_dull/"]
      }
    }
  ]
}
```

The example configuration is actually based on the PulpDist test suite - it is designed to exercise most of the major features of the PulpDist plugins in a single comprehensive scenario (some other key features, such as the use of PROTECTED files to prevent the deletion of directories, or the creation of symlinks to the most recent snapshot directory, are testing by setting up the standard scenario and adjusting some of the settings or the filesystem layout appropriately). This section aims to break the example down into components and explain how each of them works.

6.1 Working with the Example Configuration

The example configuration is designed to be used with a local rsync daemon and the `misc/create_demo_tree.py` script in the source repo.

Using `/var/pulpdist_example_data` as the location for our demonstration tree, then `/etc/rsyncd.conf` should look something like this:

```
log file = /var/log/rsyncd.log

[demo]
comment="PulpDist Example Data Source"
path=/var/pulpdist_example_data
```

With `pulpdist` installed (or else with the `src` directory in a source checkout as the current directory), the following command will create a demonstration tree:

```
python create_demo_tree.py /var/pulpdist_example_data
```

The file tree created is laid out as follows (see below for details of the subtree layout represented by `...`):

```
simple/
  ...
versioned/
  ignored/
    ...
  relevant-1/
    ...
  relevant-2/
    ...
  relevant-3/
    ...
  relevant-4/
    ...
  relevant-but-not-really/
    ...
snapshot/
  ignored/
    ...
  relevant-1/
    STATUS
    ...
  relevant-2/
    STATUS
    ...
  relevant-3/
    ...
  relevant-4/
    STATUS
    ...
  relevant-but-not-really/
    ...
```

The common subtrees all look like the following:

```
data.txt
data2.txt
skip.txt
subdir/
```

```
  data.txt
  data2.txt
  skip.txt
  subdir/
    data.txt
    data2.txt
    skip.txt
subdir2/
  data.txt
  data2.txt
  dull/
    data.txt
    data2.txt
    skip.txt
  skip.txt
```

All `STATUS` files contain the text `FINISHED` (and nothing else), while the example text files contain the text `PulpDist test data!`.

6.2 The Raw Repo Definition

The example configuration includes a single *Raw Repo Definition*. For ease of reference, it is reproduced here:

```
"RAW_REPOS": [
  {
    "repo_id": "raw_sync",
    "display_name": "Raw Sync Demo",
    "description": "Demonstration of raw sync configuration in site config",
    "notes": {
      "pulpdist": {
        "sync_hours": 24
      },
      "site_custom": {
        "origin": "PulpDist example repository"
      }
    },
    "importer_type_id": "simple_tree",
    "importer_config": {
      "tree_name": "Raw Simple Tree",
      "remote_server": "localhost",
      "remote_path": "/demo/simple/",
      "local_path": "/var/www/pub/sync_demo_raw/",
      "exclude_from_sync": ["*skip*"],
      "sync_filters": ["exclude_irrelevant/", "exclude_dull/"]
    }
  }
]
```

Raw repos map almost directly to the Pulp settings for the corresponding plugin. This has the advantage of making them entirely self contained and very flexible, but also makes their configuration very repetitive if multiple trees are being mirrored from the same source location.

The first three fields, `repo_id`, `display_name` and `description` are mainly of significance for humans. The repo ID is the unique string identifier used to refer to this repository in the command line interface, while the display name and description are shown in the web interface.

The `notes` field uses a feature of Pulp that allows arbitrary additional information to be associated with each

repository. The site_custom data is just there as an example, but the pulpdist metadata section is used to control interaction with command line client. In this case, the value 24 means that the python -m pulpdist.manage_repos cron_sync command will synchronise this repo at midnight each day if synchronisation is enabled on the repo (like all trees in the example configuration, this one has synchronisation disabled by default).

The importer_type_id field indicates which kind of synchronisation operation is being defined. The value of simple_tree indicates that this configuration entry will set up a *Simple Tree Sync* on the server.

Finally, the importer_config field actually sets up the synchronisation operation. In this case, a simple tree sync maps directly to a single call to rsync, so there isn't a great deal to be configured.

The tree_name value (along with repo_id) will appear in the sync operation logs created by the server.

The remote_server and remote_path operations are used to identify the location of the source rsync daemon (rsync over ssh is not currently supported). The local_path entry states exactly where to save the mirrored files. For the example configuration, this means files will be retrieved from rsync://localhost/demo/simple/ and saved to /var/www/pub/sync_demo_raw (the Pulp plugins run as the Apache user, and saving the files to pub makes it easy to share them again).

The last two entries are a little more interesting, as they map to rsync's filtering options. Any files or directories mentioned in exclude_from_sync are passed via rsync's --exclude option, while those mentioned in sync_filters are passed with the --filter option. This offers a great deal of flexibility in determining exactly what gets copied from the data source into the local mirror.

6.2.1 Synchronization Behaviour

The effect of this configuration is that, after running the following two commands:

```
python -m pulpdist.manage_repos enable --repo raw_sync --force
python -m pulpdist.manage_repos sync --repo raw_sync --force
```

The following filtered tree layout should be seen in /var/www/pub/sync_demo_raw:

```
data.txt
data2.txt
subdir/
  data.txt
  data2.txt
  subdir/
    data.txt
    data2.txt
subdir2/
  data.txt
  data2.txt
```

The skip.txt files because they match the pattern in the exclude_from_sync filter.

The dull directory and its contents get excluded by the exclude_dull/ entry in the sync_filters setting.

6.3 Local Mirror Definition: Simple Tree

Where a raw repo definition aims to include all the information needed to configure the rsync task directly, local mirror definitions are designed to work as part of a wider mirroring network, where various upstream servers publish trees for consumption by downstream clients. A local mirror definition is converted to a raw repo definition by the command line client before being uploaded to the Pulp server at a site.

The example configuration includes a number of *Local Mirror Definitions*.To introduce the concepts involved, we'll first review the simplest of the definitions, which describes a *Simple Tree Sync* task, just like the example raw repo definition.

6.3.1 Defining the Local Mirror

The basic mirror definition appears in the `LOCAL_MIRRORS` section of the configuration file:

```
"LOCAL_MIRRORS": [
  {
    "mirror_id": "simple_sync",
    "tree_id": "simple_sync",
    "exclude_from_sync": ["*skip*"],
    "sync_filters": ["exclude_irrelevant/"],
    "notes": {
      "basic": "note",
      "site_custom": {
        "origin": "PulpDist example repository"
      }
    }
  }
]
```

This example creates a local mirror named `simple_sync` at the default site (see below for more on sites), which will be a copy of the remote tree `simple_sync`. While the mirror and the remote tree have the same name in the example, that isn't a requirement in general.

The `notes` entry just defines a few arbitrary notes that will be added to the tree definition. This can be used to record additional information about the mirror, such as the initial rationale for creating it.

The `exclude_from_sync` and `sync_filters` entries contribute to the filter settings in the derived raw repo definition.

A local mirror definition can actually override most of the settings defined for the remote tree being mirrored. However, this particular example doesn't do that. See the *config reference* for details.

6.3.2 Defining the Remote Tree

The `tree_id` entry names a particular *Remote Tree Definition* in the `REMOTE_TREES` section:

```
"REMOTE_TREES": [
  {
    "tree_id": "simple_sync",
    "name": "Simple Sync Demo",
    "description": "Demonstration of the simple tree sync plugin",
    "tree_path": "simple",
    "sync_type": "simple",
    "sync_hours": 0,
    "source_id": "sync_demo"
  }
]
```

The `tree_id` is just a unique identifier for the tree, while the `name` and `description` fields are used for display to users.

The `tree_path` defines the name of the directory to be synchronised, relative to the base location defined by the `source_id`.

It is expected that this configuration format will eventually be expanded to include a list of alternate sources for the tree, but that feature is not yet supported.

The `sync_type` setting selects the specific importer plugin to be used. Currently only PulpDist provided plugins are supported, but this may change in future versions.

As in the raw repo example, the `sync_hours` ties into the `cron_sync` scheduling command. In this case, a setting of `0` servers to disable automatic synchronisation, even if synchronisation is enabled for the repo.

Most of the settings in the tree definition are inherited by local mirrors that don't override them. See the *config reference* for details.

6.3.3 Defining the Remote Source

The `source_id` entry names a particular *Remote Source Definition* in the `REMOTE_SOURCES` section:

```
"REMOTE_SOURCES": [
  {
    "source_id": "sync_demo",
    "server_id": "demo_server",
    "name": "Sync Demo Trees",
    "remote_path": "demo",
    "listing_suffix": "*"
  }
]
```

The `source_id` is just a unique identifier for the source, while the `name` field is intended for display to users.

The `remote_path` setting defines an the leading path component to use for the remote path when deriving the raw repo definition.

The `server-id` defines the rsync server that hosts the content provided by this source.

The `listing_suffix` isn't relevant for a simple tree definition, but can be of significance for `versioned` and `snapshot` trees. It will be discussed in more detail later in the tutorial.

See the *config reference* for additional options and details.

6.3.4 Defining the Remote Server

The `server_id` entry names a particular *Remote Server Definition* in the `REMOTE_SERVERS` section:

```
"REMOTE_SERVERS": [
  {
    "server_id": "demo_server",
    "name": "Sync Demo Server",
    "dns": "localhost"
  }
]
```

The `server_id` is just a unique identifier for the source, while the `name` field is intended for display to users.

The `dns` field is either a hostname or IP address for the source rsync server.

See the *config reference* for additional options and details.

6.3.5 Defining the Local Site

A local mirror definition may include a `site_id` setting that names a particular local site configuration to be used when deriving the raw repo definition. If no specific site is named, then the default site definition is used. The default site definition is also used to provide default values that are used when a specific site definition doesn't replace them with more specific values.

This particular mirror definition is for the default *Site Definition* in the `SITE_SETTINGS` section:

```
"SITE_SETTINGS": [
  {
    "site_id": "default",
    "name": "Default Site",
    "storage_prefix": "/var/www/pub",
    "server_prefixes": {
      "demo_server": "sync_demo",
      "other_demo_server": "sync_demo/sync_demo_trees"
    },
    "source_prefixes": {
      "sync_demo": "sync_demo_trees"
    },
    "exclude_from_sync": ["*dull*"],
    "exclude_from_listing": ["*justfortesting*"]
  }
]
```

The `site_id` is just a unique identifier for the site, while the `name` field is intended for display to users.

The `storage_prefix` is included in all local paths.

The `server_prefixes` and `source_prefixes` mappings are used to map `server_id` and `source_id` values to local path components. For this local mirror, the relevant entries are `demo_server` and `sync_demo` respectively.

The `exclude_from_sync` and `exclude_from_listing` settings affect the filtering used for various rsync operations. For a simple sync operation, only the `exclude_from_sync` operation is relevant.

See the *config reference* for additional options and details.

6.3.6 Equivalent Raw Repo Definition

A local mirror definition isn't used to configure a repo directly. Instead, an equivalent raw repo definition is derived from the local mirror definition and all of the related settings. The *config reference* gives an overview of this process.

For the simple tree mirror, the equivalent definition would look like this:

```
{
  "repo_id": "simple__default",
  "display_name": "Simple Sync Demo",
  "description": "Demonstration of the simple tree sync plugin",
  "notes": {
     "basic": "note",
     "pulpdist": {
       "mirror_id": "simple_sync",
       "server_id": "demo_server",
       "site_id": "default",
       "source_id": "sync_demo",
       "sync_hours": 0,
       "tree_id": "simple_sync"
```

```
      },
      "site_custom": {
        "origin": "PulpDist example repository"
      }
  },
  "importer_type_id": "simple_tree",
  "importer_config": {
    "tree_name": "simple_sync__default",
    "remote_server": "localhost",
    "remote_path": "/demo/simple/",
    "local_path": "/var/www/pub/sync_demo/sync_demo_trees/simple/",
    "exclude_from_sync": ["*dull*", "*skip*"],
    "sync_filters": ["exclude_irrelevant/"]
  }
}
```

The `repo_id` is a combination of the `mirror_id` and the `site_id`. This allows multiple nominal sites to be configured on the same Pulp server without identifier conflicts. Note that the command line client displays these merged IDs a little differently (`<mirror_id>(<site_id>)`). To select a mirror by its repo id, use the back end form with the double underscore separator (`<mirror_id>__<site_id>`).

The `display_name` and `description` in this case come directly from the remote tree definition.

The `notes` are a combination of those specified in the local mirror definition, along with those automatically created by the derivation process. The derived notes include the identifiers for each of the components used to derive the repo definition, along with the `sync_hours` setting for use by the `cron_sync` scheduling operation.

The `importer_type_id` is derived from the `sync_type` setting in the remote tree definition.

The import configuration details used for a simple sync operation are common to all supported importer plugins.

`tree_name` is always just the derived `repo_id` for the local mirror.

`remote_server` is the `dns` property of the remote server definition.

`remote_path` in this case is a combination of the `remote_path` entry in the remote source definition and the `tree_path` entry in the remote tree definition.

`local_path` is a combination of the `storage_prefix` from the site settings, the prefixes for the remote server and source respectively (both retrieved from the site settings) and finishing with the `tree_path` entry from the remote tree definition (this is one of those settings where the value from the remote tree definition is used if the local mirror definition doesn't override it).

The `exclude_from_sync` setting includes the value from the local mirror definition along with the value from the default site settings.

The `sync_filters` setting is taken directly from the local mirror definition, as this particular remote tree definition omits all of the filtering options.

Unlisted configuration options are left at their default values.

6.3.7 Synchronization Behaviour

The effect of this configuration is that, after running the following two commands:

```
python -m pulpdist.manage_repos enable --mirror simple_sync --force
python -m pulpdist.manage_repos sync --mirror simple_sync --force
```

The following filtered tree layout should be seen in `/var/www/pub/sync_demo/sync_demo_trees/simple`:

```
data.txt
data2.txt
subdir/
  data.txt
  data2.txt
  subdir/
    data.txt
    data2.txt
subdir2/
  data.txt
  data2.txt
```

This is the same as the tree layout produced by the example raw repo definition.

6.3.8 Why Use Mirror Definitions?

From the worked example, it may seem that mirror definitions are actually harder to use than the equivalent raw repo definitions. If you only want to mirror a single tree, this is true (that's why the option to provide a raw repo definition exists).

The primary use case for PulpDist, however, is for an internal mirroring network, where any given rsync server will be publishing multiple trees, and any given site will be downloading multiple trees (potentially from different sources).

The advantage of the mirror definition format is that it allows this arrangement to be modelled directly - when setting up a new local mirror for an existing remote tree, all you need to know is the id of the remote tree and the id of the site where the mirror is being created, rather than all of the details necessary to create the raw repo definition by hand. Avoiding the data duplication also helps ensure consistency between mirrors, and also makes various data changes substantially easier (for example, changing the hostname of a particular upstream rsync server).

6.4 Local Mirror Definition: Versioned Tree

Where a simple sync definition maps directly to a single invocation of rsync, a versioned sync performs an initial listing step to identify a set of remote directories. A separate rsync task is then invoked for each directory. This is useful when a subset of directories from a particular remote directory are being split out to separate locations in the local mirror.

The versioned tree definition in the example site configuration is set up to show the mechanism for limiting a mirror definition to a specific site. It also shows the additional filtering options that become available once the mirroring plugin switches to the two-step process of first doing a remote listing to identify the trees to be synchronised and then issuing a separate rsync command to mirror each tree.

The "versioned tree" name comes from the original use case for this plugin, which is to mirror a subset of versions from a product directory where each version is split out into a separate directory, but new maintenance releases may be added to old version directories. In practice, the plugin works for any tree where it is desirable to mirror a subset of the available top-level directories using a set of selection filters that differ from those used for the actual mirror operations.

6.4.1 Defining the Local Mirror

The basic mirror definition appears in the `LOCAL_MIRRORS` section of the configuration file:

```
"LOCAL_MIRRORS": [
  {
    "mirror_id": "versioned_sync",
```

```
    "tree_id": "versioned_sync",
    "site_id": "other",
    "exclude_from_sync": ["*skip*"],
    "sync_filters": ["exclude_dull/"],
    "exclude_from_listing": ["relevant-but*"],
    "notes": {
      "site_custom": {
        "origin": "PulpDist example repository"
      }
    }
  }
]
```

This example creates a local mirror named `versioned_sync` at the site `other`, which will be a copy of the remote tree `versioned_sync`. As with the `simple_sync` example, using the same name for the local mirror and the remote tree is entirely optional.

The `notes` entry is again used to record additional information about the mirror, such as the initial rationale for creating it.

The `exclude_from_sync` and `sync_filters` entries contribute to the filter settings in the derived raw repo definition. These filters apply to the step of synchronising the individual trees

The `exclude_from_listing` setting controls which remote directories will be synchronised at all.

See the *config reference* for additional options and details.

6.4.2 Defining the Remote Tree

The `tree_id` entry names a particular *Remote Tree Definition* in the `REMOTE_TREES` section:

```
"REMOTE_TREES": [
  {
    "tree_id": "versioned_sync",
    "name": "Versioned Sync Demo",
    "description": "Demonstration of the versioned tree sync plugin",
    "tree_path": "versioned",
    "sync_type": "versioned",
    "sync_hours": 12,
    "source_id": "sync_demo_other",
    "listing_pattern": "relevant*",
    "exclude_from_sync": ["*skip*"],
    "sync_filters": ["exclude_irrelevant/"]
  }
]
```

The settings here are largely the same as those for the simple local mirror.

The setting of `12``for ``sync_hours` indicates that `cron_sync` should sync this repo at 12 AM and 12 PM each day.

The `listing_pattern` setting restricts the trees which will be considered for synchronisation, while `exclude_from_sync` and `sync_filters` contribute to the rsync settings for the actual tree synchronisation tasks.

See the *config reference* for additional options and details.

6.4.3 Defining the Remote Source

The `source_id` entry names a particular *Remote Source Definition* in the `REMOTE_SOURCES` section:

```
"REMOTE_SOURCES": [
  {
    "source_id": "sync_demo_other",
    "server_id": "other_demo_server",
    "name": "Other Sync Demo Trees",
    "remote_path": "demo",
    "listing_suffix": "*"
  }
]
```

Aside from referring to a different remote server, the settings here are essentially the same as those for the simple local mirror.

While the `listing_suffix` can be relevant for versioned tree definitions, in this case it is superseded by the `listing_pattern` setting in the remote tree definition.

See the *config reference* for additional options and details.

6.4.4 Defining the Remote Server

The `server_id` entry names a particular *Remote Server Definition* in the `REMOTE_SERVERS` section:

```
"REMOTE_SERVERS": [
  {
    "server_id": "other_demo_server",
    "name": "Other Sync Demo Server",
    "dns": "localhost"
  }
]
```

As this is just an example site configuration, the "other" remote server also resolves to the local machine. This can also occur in real mirroring networks if multiple logical servers end up being combined on a single physical server.

See the *config reference* for additional options and details.

6.4.5 Defining the Local Site

The scope of this mirror is limited to a specific site. This means the settings for the named site become relevant, while those for the default site also still apply.

Both of these *Site Definitions* are given in the `SITE_SETTINGS` section:

```
"SITE_SETTINGS": [
  {
    "site_id": "default",
    "name": "Default Site",
    "storage_prefix": "/var/www/pub",
    "server_prefixes": {
      "demo_server": "sync_demo",
      "other_demo_server": "sync_demo_trees"
    },
    "source_prefixes": {
      "sync_demo": "sync_demo_trees"
    },
```

```
    "exclude_from_sync": ["*dull*"],
    "exclude_from_listing": ["*justfortesting*"]
  },
  {
    "site_id": "other",
    "name": "Other Site",
    "storage_prefix": "/var/www/pub/sync_demo"
  }
]
```

The interesting point to note is that this site definition overrides the `storage_prefix` setting. This will be used in preference to the default setting when deriving the raw repo configuration.

See the *config reference* for additional options and details.

6.4.6 Equivalent Raw Repo Definition

For the versioned tree mirror, the equivalent raw repo definition looks like this:

```
{
  "repo_id": "versioned__other",
  "display_name": "Versioned Sync Demo",
  "description": "Demonstration of the versioned tree sync plugin",
  "notes": {
    "pulpdist": {
      "mirror_id": "versioned_sync",
      "source_id": "sync_demo_other",
      "server_id": "other_demo_server",
      "site_id": "other",
      "sync_hours": 12,
      "tree_id": "versioned_sync"
    },
    "site_custom": {
      "origin": "PulpDist example repository"
    }
  },
  "importer_type_id": "versioned_tree",
  "importer_config": {
    "tree_name": "versioned_sync__other",
    "remote_server": "localhost",
    "remote_path": "/demo/versioned/",
    "local_path": "/var/www/pub/sync_demo/sync_demo_trees/versioned/",
    "exclude_from_sync": ["*dull*", "*skip*"],
    "sync_filters": ["exclude_dull/", "exclude_irrelevant/"],
    "listing_pattern": "relevant*",
    "exclude_from_listing": ["*justfortesting*", "relevant-but*"]
  }
}
```

The derivation of most of these settings is essentially the same as that for the simple mirror.

`local_path` is slightly different, in that the `storage_prefix` comes from the settings for the `other` site, while the prefix for the remote server still comes from the default site, and there is no prefix at all for the nominated remote source.

The `exclude_from_sync` setting includes the value from the local mirror definition along with the value from the default site settings. Note that the duplicate value from the remote tree settings has been omitted.

The `sync_filters` setting includes the filter options from both the local mirror and remote tree defitions.

The completely new configuration settings all relate to the remote listing step.

The `listing_pattern` is taken directly from the remote tree configuration and is passed to rsync to indicate which directories to include in the listing.

The `exclude_from_listing` setting includes the value from the local mirror definition along with the value from the default site settings.

Unlisted configuration options are left at their default values.

6.4.7 Synchronization Behaviour

The effect of this configuration is that, after running the following two commands:

```
python -m pulpdist.manage_repos enable --mirror versioned_sync --force
python -m pulpdist.manage_repos sync --mirror versioned_sync --force
```

The following filtered tree layout should be seen in `/var/www/pub/sync_demo/sync_demo_trees/versioned`:

```
relevant-1/
  ...
relevant-2/
  ...
relevant-3/
  ...
relevant-4/
  ...
```

Where the individual tree layouts represented by `...` are the same as those produced by both the local mirror and raw repo simple sync definitions.

The `ignored` directory is omitted because it does not match the derived `listing_pattern` setting.

The `relevant-but-not-really` directory is omitted because it matches one of the patterns in the `exclude_from_listing` setting.

6.5 Local Mirror Definition: Snapshot Tree

Snapshot tree definitions are very similar to versioned tree definitions, as they also perform an initial directory listing step before proceeding to separate sync operations for each identified directory.

The difference is that snapshot sync operations are designed for systems where individual trees are never modified after their initial creation (for example, a system which creates automatic nightly builds with a date-based naming scheme for the build directories).

The state of individual trees is recorded in a STATUS at the root of each directory. If this file exists and contains the text `FINISHED` then it indicates that the tree is available for synchronisation (if present at the remote site) or has already been synchronised (if present at the local site). For large trees, this allows a lot of wasted data transfers to be skipped: already synchronised trees don't need to be checked for changes, and unusable trees from the remote site don't need to be copied in the first place.

6.5.1 Defining the Local Mirror

The basic mirror definition appears in the `LOCAL_MIRRORS` section of the configuration file:

```
"LOCAL_MIRRORS": [
  {
    "mirror_id": "snapshot_sync",
    "tree_id": "snapshot_sync",
    "notes": {
      "site_custom": {
        "origin": "PulpDist example repository"
      }
    }
  }
]
```

This example aims to show an almost minimal local mirror definition. The only optional information here is the note indicating why this mirror exists.

See the *config reference* for additional options and details.

6.5.2 Defining the Remote Tree

The `tree_id` entry names a particular *Remote Tree Definition* in the `REMOTE_TREES` section:

```
"REMOTE_TREES": [
  {
    "tree_id": "snapshot_sync",
    "name": "Snapshot Sync Demo",
    "description": "Demonstration of the snapshot tree sync plugin",
    "tree_path": "snapshot",
    "sync_type": "snapshot",
    "sync_hours": 1,
    "source_id": "sync_demo",
    "listing_prefix": "re*ev",
    "latest_link": "latest-relevant",
    "exclude_from_listing": ["relevant-but*"],
    "exclude_from_sync": ["*skip*"],
    "sync_filters": ["exclude_irrelevant/", "exclude_dull/"]
  }
]
```

The settings here are largely the same as those for the simple local mirror.

The setting of `1``for ``sync_hours` indicates that `cron_sync` should sync this repo every hour.

The `listing_prefix` setting is another way to restrict the trees which will be considered for synchronisation. Unlike `listing_pattern`, which completely defines the inclusion filter, `listing_prefix` is combined with the `listing_suffix` setting from the relevant remote source definition.

The `exclude_from_listing` filter provides a pattern for directories that would otherwise match the inclusion filter, but should still not be synchronised.

The `latest_link` setting indicates that a symlink should be created that always points to the most recently synchronised tree, and that it should be called `latest-relevant`

As with the versioned tree, `exclude_from_sync` and `sync_filters` contribute to the rsync settings for the actual tree synchronisation tasks.

See the *config reference* for additional options and details.

6.5.3 Defining the Remote Source

The `source_id` entry names a particular *Remote Source Definition* in the `REMOTE_SOURCES` section:

```
"REMOTE_SOURCES": [
  {
    "source_id": "sync_demo",
    "server_id": "demo_server",
    "name": "Sync Demo Trees",
    "remote_path": "demo",
    "listing_suffix": "*"
  }
]
```

This is the exact same source as is used for the simple local mirror definition.

The `listing_suffix` becomes relevant in this case, as this source is now being used for a sync operation with a listing step based on `listing_prefix`. While the example configuration allows any suffix, real deployments may use this setting to enforce a standard version numbering or date formatting scheme for a particular remote source.

See the *config reference* for additional options and details.

6.5.4 Defining the Remote Server

The `server_id` entry names a particular *Remote Server Definition* in the `REMOTE_SERVERS` section:

```
"REMOTE_SERVERS": [
  {
    "server_id": "demo_server",
    "name": "Sync Demo Server",
    "dns": "localhost"
  }
]
```

As the remote server is specified by the remote source, this is the exact same server as is used for the simple local mirror definition.

See the *config reference* for additional options and details.

6.5.5 Defining the Local Site

Like the simple local mirror, the snapshot mirror example uses the default site settings directly.

This *Site Definitions* is given in the `SITE_SETTINGS` section:

```
"SITE_SETTINGS": [
  {
    "site_id": "default",
    "name": "Default Site",
    "storage_prefix": "/var/www/pub",
    "server_prefixes": {
      "demo_server": "sync_demo",
      "other_demo_server": "sync_demo_trees"
    },
    "source_prefixes": {
      "sync_demo": "sync_demo_trees"
    },
    "exclude_from_sync": ["*dull*"],
```

```
    "exclude_from_listing": ["*justfortesting*"]
  }
]
```

The only difference with the simple local mirror is that the `exclude_from_listing` setting becomes relevant, as the snapshot sync plugin includes the listing step.

See the *config reference* for additional options and details.

6.5.6 Equivalent Raw Repo Definition

For the versioned tree mirror, the equivalent raw repo definition looks like this:

```
{
  "repo_id": "snapshot_sync__default",
  "display_name": "Snapshot Sync Demo",
  "description": "Demonstration of the snapshot tree sync plugin",
  "notes": {
    "pulpdist": {
      "mirror_id": "snapshot_sync",
      "source_id": "sync_demo",
      "server_id": "demo_server",
      "sync_hours": 1,
      "site_id": "default",
      "tree_id": "snapshot_sync"
    },
    "site_custom": {
      "origin": "PulpDist example repository"
    }
  },
  "importer_type_id": "snapshot_tree",
  "importer_config": {
    "sync_filters": ["exclude_irrelevant/", "exclude_dull/"],
    "remote_path": "/test_data/snapshot/",
    "latest_link_name": "latest-relevant",
    "tree_name": "snapshot_sync__default",
    "exclude_from_sync": ["*dull*", "*skip*"],
    "exclude_from_listing": ["*justfortesting*", "relevant-but*"],
    "remote_server": "localhost",
    "listing_pattern": "re*ev*",
    "local_path": "/var/www/pub/sync_demo/sync_demo_trees/snapshot/"
  }
}
```

The derivation of most of these settings is essentially the same as in the previous examples.

The `exclude_from_sync` setting includes the value from the remote tree definition along with the value from the default site settings.

The `latest_link_name` and `sync_filters` settings are taken directly from the remote tree settings.

The `listing_pattern` is derived by concatenating the `listing_prefix` from the remote tree settings with the `listing_suffix` from the remote source settings.

The `exclude_from_listing` setting includes the value from the remote tree definition along with the value from the default site settings.

Unlisted configuration options are left at their default values.

6.5.7 Synchronization Behaviour

The effect of this configuration is that, after running the following two commands:

```
python -m pulpdist.manage_repos enable --mirror snapshot_sync --force
python -m pulpdist.manage_repos sync --mirror snapshot_sync --force
```

The following filtered tree layout should be seen in `/var/www/pub/sync_demo/sync_demo_trees/snapshot`:

```
relevant-1/
  ...
relevant-2/
  ...
relevant-4/
  ...
latest-relevant -> ./relevant-4
```

Where the individual tree layouts represented by `...` are the same as those produced by both the local mirror and raw repo simple sync definitions.

The `ignored` directory is omitted because it does not match the derived `listing_pattern` setting.

The `relevant-but-not-really` directory is omitted because it matches one of the patterns in the `exclude_from_listing` setting.

The `relevant-3` directory is omitted because it does not contain the `STATUS` file to indicate that the tree is valid.

The `latest-relevant` symlink refers to `relevant-4` as that is the most recent tree to be synchronised.

CHAPTER 7

PulpDist Python API

This is the current Python API exposed by the `pulpdist` package. It is in a *very* preliminary state and comes with no backwards compatibility guarantees at all.

7.1 pulpdist Package

7.1.1 `manage_repos` Module

7.1.2 `manage_site` Module

7.1.3 Subpackages

cli Package

`commands` Module

`display` Module

`repo_cli` Module

`thread_pool` Module

Simple size constrained thread pool that blocks when all threads are busy

exception `pulpdist.cli.thread_pool.`**`PendingTasks`**
Bases: `exceptions.Exception`

Exception thrown if ThreadPool.wait_for_tasks() times out

class `pulpdist.cli.thread_pool.`**`Task`**(*priority*, *func*, *args*, *kwds*)
Bases: `tuple`

`args`
Alias for field number 2

`func`
Alias for field number 1

`kwds`
Alias for field number 3

priority
Alias for field number 0

class pulpdist.cli.thread_pool.**ThreadPool**(*num_threads*, *name='ThreadPool'*)
Pool of threads consuming tasks from a queue

add_task(*priority*, *func*, **args*, ***kwds*)
Add a task to the queue. Blocks if all threads are busy.

wait_for_tasks(*timeout=None*)
Wait for completion of all the tasks in the queue

class pulpdist.cli.thread_pool.**Worker**(*tasks*, *name=None*)
Bases: threading.Thread

Thread executing tasks from a given tasks queue

run()

core Package

mirror_config Module

Convert from a site mirror config to a PulpDist repo config

class pulpdist.core.mirror_config.**MirrorConverter**(*mirror*)
Bases: object

build_importer_config()

build_notes()

pulpdist.core.mirror_config.**make_repo**(*mirror*)

pulpdist.core.mirror_config.**make_repo_id**(*mirror_id*, *site_id*)
Derive a Pulp repo ID from a mirror ID and a site ID

pulpapi Module

repo_config Module

Config definitions and helpers for pulpdist importer plugins

class pulpdist.core.repo_config.**RepoConfig**(*config*)
Bases: *pulpdist.core.validation.ValidatedConfig*

validate()

shellutil Module

shellutil - additional shell utilities (beyond the standard library's shutil)

class pulpdist.core.shellutil.**WalkedDir**(*path*, *subdirs*, *files*, *depth*)
Bases: tuple

depth
Alias for field number 3

files
Alias for field number 2

path
Alias for field number 0

subdirs
Alias for field number 1

pulpdist.core.shellutil.**filtered_walk**(*top*, *file_pattern=None*, *dir_pattern=None*, *excluded_files=None*, *excluded_dirs=None*, *depth=None*, *followlinks=False*, *onerror=None*, *onloop=None*)

filtered_walk is similar to os.walk, but offers the following additional features:

- yields a named tuple of (path, subdirs, files, depth)
- allows independent glob-style filters for filenames and subdirectories
- allows independent exclusion filters for filenames and subdirectories
- emits a message to stderr and skips the directory if a symlink loop is encountered when following links
- allows a recursion depth limit to be specified

Selective walks are always top down, as the directory listings must be altered to provide the above features. If not None, depth must be at least 0. A depth of zero can be useful to get separate filtered subdirectory and file listings for a given directory.

onerror is passed to os.walk to handle os.listdir errors onloop (if provided) can be used to override the default symbolic loop handling. It is called with the directory path as an argument when a loop is detected. Any false return value will skip the directory, any true value means the directory will be processed as normal.

pulpdist.core.shellutil.**temp_dir**(**args*, ***kwds*)

site_config Module

site_sql Module

sync_config Module

Config definitions and helpers for pulpdist importer plugins

class pulpdist.core.sync_config.**SnapshotSyncConfig**(*config=None*)
Bases: *pulpdist.core.sync_config.VersionedSyncConfig*

class pulpdist.core.sync_config.**TreeSyncConfig**(*config=None*)
Bases: *pulpdist.core.validation.ValidatedConfig*

class pulpdist.core.sync_config.**VersionedSyncConfig**(*config=None*)
Bases: *pulpdist.core.sync_config.TreeSyncConfig*

pulpdist.core.sync_config.**retrieves_listing**(*sync_type*)

sync_trees Module

sync_trees - utilities for synchronising trees with rsync

class pulpdist.core.sync_trees.**BaseSyncCommand**(*config*, *log_dest=None*)
Bases: object

CONFIG_TYPE = None

DRY_RUN_SUFFIX = '_DRY_RUN'

SYNC_COMPLETED = 'SYNC_COMPLETED'

SYNC_DISABLED = 'SYNC_DISABLED'

SYNC_FAILED = 'SYNC_FAILED'

SYNC_PARTIAL = 'SYNC_PARTIAL'

SYNC_UP_TO_DATE = 'SYNC_UP_TO_DATE'

fetch_dir(*remote_source_path*, *local_dest_path*, *local_seed_paths=()*)
Fetch a single directory from the remote server

run_sync()
Execute the full synchronisation task

Ensures the sync log is flushed before returing

class pulpdist.core.sync_trees.**SyncFromDelta**
Bases: *pulpdist.core.sync_trees.BaseSyncCommand*

Create a new local snapshots from an upstream delta

class pulpdist.core.sync_trees.**SyncSnapshotDelta**
Bases: *pulpdist.core.sync_trees.BaseSyncCommand*

Create an rsync delta from a snapshot directory

class pulpdist.core.sync_trees.**SyncSnapshotTree**(*config*, *log_dest=None*)
Bases: *pulpdist.core.sync_trees.SyncVersionedTree*

Sync the contents of a directory containing multiple snapshots of a tree

CONFIG_TYPE
alias of SnapshotSyncConfig

class pulpdist.core.sync_trees.**SyncStats**
Bases: *pulpdist.core.sync_trees.SyncStats*

classmethod from_rsync_output(*raw_data*, *old_daemon=False*)

class pulpdist.core.sync_trees.**SyncTree**(*config*, *log_dest=None*)
Bases: *pulpdist.core.sync_trees.BaseSyncCommand*

Sync the contents of a directory

CONFIG_TYPE
alias of TreeSyncConfig

class pulpdist.core.sync_trees.**SyncVersionedTree**(*config*, *log_dest=None*)
Bases: *pulpdist.core.sync_trees.BaseSyncCommand*

Sync the contents of a directory containing multiple versions of a tree

CONFIG_TYPE
alias of VersionedSyncConfig

remote_ls(*remote_ls_path*)

util Module

util - miscellaneous utility functions

pulpdist.core.util.**call_repr**(*name*, *args*)

pulpdist.core.util.**format_iter**(*iterable*, *fmt='{0!r}'*, *sep=', '*)

pulpdist.core.util.**obj_repr**(*obj*, *fields*)

validation Module

Simple validation for JSON compatible data structures

class pulpdist.core.validation.**ValidatedConfig**(*config=None*)
Bases: object

classmethod check()

classmethod ensure_validated(*config*)
Returns a mapping that has been validated against the spec

classmethod from_json(*json_config*)
Read the config from a JSON file and ensure it is valid

classmethod post_validate(*value*)

validate()

exception pulpdist.core.validation.**ValidationError**
Bases: exceptions.Exception

pulpdist.core.validation.**check_host**(*allow_none=False*)

pulpdist.core.validation.**check_mapping**(*spec*, *allow_none=False*, *allow_extra=False*)

pulpdist.core.validation.**check_mapping_items**(*key_validator*, *value_validator*, *allow_none=False*)

pulpdist.core.validation.**check_path**(*allow_none=False*)

pulpdist.core.validation.**check_pulp_id**(*expected='valid Pulp ID'*, *allow_none=False*)

pulpdist.core.validation.**check_regex**(*pattern*, *expected=None*, *allow_none=False*)

pulpdist.core.validation.**check_remote_path**(*allow_none=False*)

pulpdist.core.validation.**check_rsync_filter**(*allow_none=False*)

pulpdist.core.validation.**check_rsync_filter_sequence**()

pulpdist.core.validation.**check_sequence**(*item_validator*, *allow_none=False*)

pulpdist.core.validation.**check_simple_id**(*expected='simple ID (alphanumeric, underscores, hyphens)'*, *allow_none=False*)

pulpdist.core.validation.**check_text**(*allow_none=False*)

pulpdist.core.validation.**check_type**(*expected_type*, *allow_none=False*)

pulpdist.core.validation.**check_value**(*allowed_values*, *allow_none=False*)

pulpdist.core.validation.**fail_validation**(*fmt*, **args*, ***kwds*)

pulpdist.core.validation.**validate_config**(*config*, *spec*, **args*, ***kwds*)

django_app Package

admin Module

auth Module

fields Module

forms Module

models Module

restapi Module

urls Module

util Module

views Module

Subpackages

templatetags Package

pulpdist_tags Module

django_site Package

dummy_auth Module

management_settings Module

settings Module

urls Module

7.2 pulpdist_importers Package

Note that this package cannot be imported directly from most Python code. Instead, it is installed into the Pulp importer plugins directory to be loaded by the Pulp server.

7.2.1 importer Module

CHAPTER 8

PulpDist Development

PulpDist is written primarily in Python and developed in git on Fedora Hosted. Issue tracking is handled in Bugzilla.

8.1 Target Platforms

The code is currently tested and known to work under Python 2.7 on Fedora and under Python 2.6 on RHEL6. It should also run under either version of Python on other *nix systems (so long as the relevant dependencies are available).

The client and plugins are written to work with the 1.x series of Pulp. Any errors encountered while using Pulp 1.x should be reported on the bug tracker.

The Pulp 2.x series (due for initial release in July 2012) is not currently supported.

8.2 Build/Test Dependencies

- setuptools/distribute (packaging)
- setuptools-git (tito RPM build tool support)
- tito (RPM build tool)
- sphinx (the reStructuredText documentation tool)
- sphinxcontrib-blockdiag (not used yet, but will be eventually)
- nose (test runner)
- unittest2 (backport of Python 2.7 unittest module to earlier versions)
- mock (the Python test library, not the Fedora packaging utility)
- mock/mockbuild (the Fedora packaging utility)
- djangosanetesting (web app test runner)
- parse (date/time checking)

8.3 Plugin Dependencies

(not necessarily complete)

- rsync (currently used via CLI, may some day switch to librsync)

- pulp (of course!)

8.4 Web Application Dependencies

(not necessarily complete)

- Django 1.3+ (built on Class Based Views)
- Django-south (database migrations)
- python-m2crypto (OAuth support, including protected config storage)
- python-oauth2 (OAuth based access to Pulp)
- django-tables2 (simple HTML display of tabular data)
- djangorestframework (simple development of rich REST APIs)
- pulp-admin (used to simplify access to server REST API)

Standard deployment configuration assumes Apache + mod_wsgi + mod_auth_kerb deployment, but alternatives are likely possible.

8.5 Setting up a basic devel environment

First, install the pulp-admin client as described in the *Pulp Installation Guide*.

The following set of instructions should then provide a working development instance of the `pulpdist` web application on a Fedora 16 system:

```
$ sudo yum install Django Django-south python-nose python-m2crypto python-oauth2 tito
$ sudo wget -O /etc/yum.repos.d/fedora-pulpdist.repo http://repos.fedorapeople.org/repos/pulpdist/pul
$ sudo yum install python-django-tables2 python-djangorestframework python-mock python-djangosanetest

$ git clone git://fedorahosted.org/pulpdist.git pulpdist
$ cd pulpdist/src
$ python -m pulpdist.manage_site syncdb
$ python -m pulpdist.manage_site migrate
$ python -m pulpdist.manage_site runserver
```

Pointing your preferred browser at `http://localhost:8000` should then display the web UI with the dummy authentication scheme enabled. Pulp server definitions can be entered either through the REST API or else via the Django admin interface (use `pulpdist-test-su` as the login name to get access to the latter).

Pulp Installation Guide: http://pulpproject.org/ug/UGInstallation.html

8.6 Running the unit tests

Running the test suite (from the base directory of the source checkout):

```
$ make test
```

Some of these test may require a Pulp server running on the local machine with OAuth enabled. Refer to the *Pulp Installation Guide* and OAuth authentication for details.

8.7 Building the PulpDist RPMs

Currently, there are no prebuilt RPMs for PulpDist available. However,creating them locally is intended to be straightforward:

```
$ make rpm
```

This will create a `pulpdist` SRPM, along with the following `noarch` RPMs:

- `pulpdist` - the core Python package for PulpDist
- `pulpdist-plugins` - the custom Pulp plugins for tree synchronisation
- `pulpdist-django` - a meta-package that brings in the additional dependencies needed to actually run `pulpdist.django_app`
- `pulpdist-httpd` - installs the PulpDist web application, largely preconfigured to run under Apache using Kerberos-over-Basic-Auth for authentication.
- `pulpdist-devel` - a meta-package that isn't currently very useful, but will eventually be available in the public repo to make it easy to bring in all the dependencies needed to work on PulpDist.

`pulpdist-plugins` should be installed on all Pulp servers in a PulpDist network.

`pulpdist-httpd` can be installed directly to use the standard PulpDist Django site settings. Alternatively, any RPM-based Django site definitions that use the PulpDist Django application should depend on `pulpdist` and `pulpdist-django`.

CHAPTER 9

Indices and tables

- genindex
- modindex
- search

p

P

R

S

T

V

W