MyPLC User's Guide

Mark Huang

Revision History

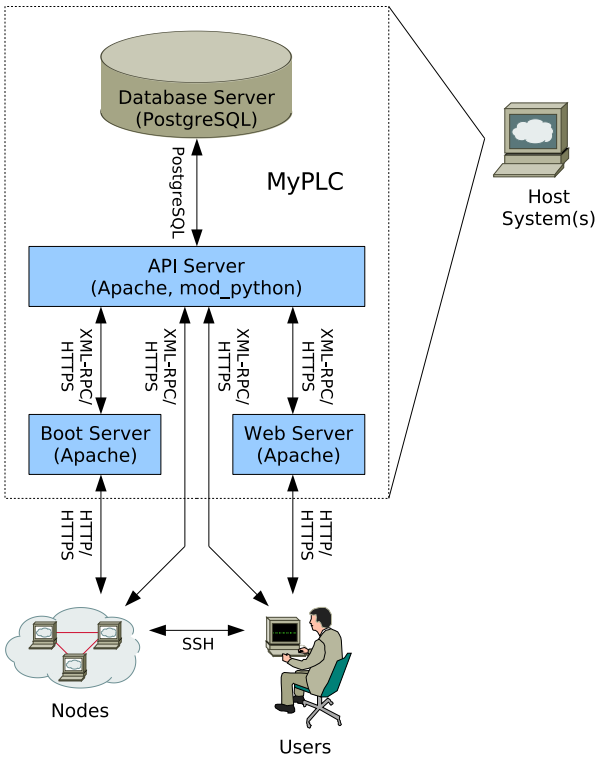
Revision 1.0 April 7, 2006 Revised by: MLH
Initial draft.
Revision 1.1 July 19, 2006 Revised by: MLH
Add development environment.
Revision 1.2 August 18, 2006 Revised by: TPT
Review section on configuration and introduce **plc-config-tty**. Present implementation details last.

This document describes the design, installation, and administration of MyPLC, a complete PlanetLab Central (PLC) portable installation contained within a **chroot** jail. This document assumes advanced knowledge of the PlanetLab architecture and Linux system administration.

Table of Contents

Overview

MyPLC is a complete PlanetLab Central (PLC) portable installation contained within a **chroot** jail. The default installation consists of a web server, an XML-RPC API server, a boot server, and a database server: the core components of PLC. The installation is customized through an easy-to-use graphical interface. All PLC services are started up and shut down through a single script installed on the host system. The usually complex process of installing and administering the PlanetLab backend is reduced by containing PLC services within a virtual filesystem. By packaging it in such a manner, MyPLC may also be run on any modern Linux distribution, and could conceivably even run in a PlanetLab slice.



MyPLC should be viewed as a single application that provides multiple functions and can run on any host system.

Figure 1. MyPLC architecture

Purpose of the *myplc-devel* package

The *myplc* package comes with all required node software, rebuilt from the public PlanetLab CVS repository. If for any reason you need to implement your own customized version of this software, you can use the *myplc-devel* package instead, for setting up your own development environment, including a local CVS repository; you can then freely manage your changes and rebuild your customized version of *myplc*. We also provide good practices, that will then allow you to resync your local CVS repository with any further evolution on the mainstream public PlanetLab software.

Requirements

myplc and *myplc-devel* were designed as **chroot** jails so as to reduce the requirements on your host operating system. So in theory, these distributions should work on virtually any Linux 2.6 based distribution, whether it supports rpm or not.

However, things are never that simple and there indeed are some known limitations to this, so here are a couple notes as a recommended reading before you proceed with the installation.

As of 17 August 2006 (i.e *myplc-0.5-2*) :

- The software is vastly based on *Fedora Core 4*. Please note that the build server at Princeton runs *Fedora Core 2*, togother with a upgraded version of yum.
- myplc and myplc-devel are known to work on both *Fedora Core 2* and *Fedora Core 4*. Please note however that, on fc4 at least, it is highly recommended to use the `Security Level Configuration` utility and to *switch off SElinux* on your box because :
  - myplc requires you to run SElinux as 'Permissive' at most
  - myplc-devel requires you to turn SElinux Off.
- In addition, as far as myplc is concerned, you need to check your firewall configuration since you need, of course, to open up the *http* and *https* ports, so as to accept connections from the managed nodes and from the users desktops.

Installating and using MyPLC

Though internally composed of commodity software subpackages, MyPLC should be treated as a monolithic software application. MyPLC is distributed as single RPM package that has no external dependencies, allowing it to be installed on practically any Linux 2.6 based distribution.

Installing MyPLC.

- If your distribution supports RPM:

```
# rpm -U http://build.planet-lab.org/build/myplc-0_4-rc1/RPMS/i386/myplc-0.4-1.planet
```

- If your distribution does not support RPM:

```
# cd /tmp
# wget http://build.planet-lab.org/build/myplc-0_4-rc1/RPMS/i386/myplc-0.4-1.planetla
# cd /
# rpm2cpio /tmp/myplc-0.4-1.planetlab.i386.rpm | cpio -diu
```

The the Section called *Files and directories involved in myplc* below explains in details the installation strategy and the miscellaneous files and directories involved.

QuickStart

On a Red Hat or Fedora host system, it is customary to use the **service** command to invoke System V init scripts. As the examples suggest, the service must be started as root:

Example 1. Starting MyPLC:

```
# service plc start
```

Example 2. Stopping MyPLC:

```
# service plc stop
```

In the Section called *Understanding the startup sequence*, we provide greater details that might be helpful in the case where the service does not seem to take off correctly.

Like all other registered System V init services, MyPLC is started and shut down automatically when your host system boots and powers off. You may disable automatic startup by invoking the **chkconfig** command on a Red Hat or Fedora host system:

Example 3. Disabling automatic startup of MyPLC.

```
# chkconfig plc off
```

Example 4. Re-enabling automatic startup of MyPLC.

```
# chkconfig plc on
```

Changing the configuration

After verifying that MyPLC is working correctly, shut it down and begin changing some of the default variable values. Shut down MyPLC with **service plc stop** (see the Section called *QuickStart*).

The preferred option for changing the configuration is to use the **plc-config-tty** tool. This tools comes with the root image, so you need to have it mounted first. The full set of applicable variables is described in Appendix B, but using the **u** guides you to the most useful ones. Here is sample session:

Example 5. Using plc-config-tty for configuration:

```
# service plc mount
Mounting PLC:                                              [  OK  ]
# chroot /plc/root su -
<plc> # plc-config-tty
Config file /etc/planetlab/configs/site.xml located under a non-existing directory
Want to create /etc/planetlab/configs [y]/n ? y
Created directory /etc/planetlab/configs
Enter command (u for usual changes, w to save, ? for help) u
== PLC_NAME : [PlanetLab Test] OneLab
== PLC_ROOT_USER : [root@localhost.localdomain] root@odie.inria.fr
== PLC_ROOT_PASSWORD : [root] plain-passwd
== PLC_MAIL_SUPPORT_ADDRESS : [root+support@localhost.localdomain] support@one-lab.org
== PLC_DB_HOST : [localhost.localdomain] odie.inria.fr
== PLC_API_HOST : [localhost.localdomain] odie.inria.fr
== PLC_WWW_HOST : [localhost.localdomain] odie.inria.fr
== PLC_BOOT_HOST : [localhost.localdomain] odie.inria.fr
== PLC_NET_DNS1 : [127.0.0.1] 138.96.250.248
== PLC_NET_DNS2 : [None] 138.96.250.249
Enter command (u for usual changes, w to save, ? for help) w
Wrote /etc/planetlab/configs/site.xml
Merged
        /etc/planetlab/default_config.xml
and     /etc/planetlab/configs/site.xml
```

```
into    /etc/planetlab/plc_config.xml
You might want to type 'r' (restart plc) or 'q' (quit)
Enter command (u for usual changes, w to save, ? for help) r
==================== Stopping plc
...
==================== Starting plc
...
Enter command (u for usual changes, w to save, ? for help) q
<plc> # exit
#
```

If you used this method for configuring, you can skip to the the Section called *Login as a real user*. As an alternative to using **plc-config-tty**, you may also use a text editor, but this requires some understanding on how the configuration files are used within myplc. The *default* configuration is stored in a file named `/etc/planetlab/default_config.xml`, that is designed to remain intact. You may store your local changes in any file located in the `configs/` sub-directory, that are loaded on top of the defaults. Finally the file `/etc/planetlab/plc_config.xml` is loaded, and the resulting configuration is stored in the latter file, that is used as a reference.

Using a separate file for storing local changes only, as **plc-config-tty** does, is not a workable option with a text editor because it would involve tedious xml re-assembling. So your local changes should go in `/etc/planetlab/plc_config.xml`. Be warned however that any change you might do this way could be lost if you use **plc-config-tty** later on.

This file is a self-documenting configuration file written in XML. Variables are divided into categories. Variable identifiers must be alphanumeric, plus underscore. A variable is referred to canonically as the uppercase concatenation of its category identifier, an underscore, and its variable identifier. Thus, a variable with an `id` of `slice_prefix` in the `plc` category is referred to canonically as PLC_SLICE_PREFIX.

The reason for this convention is that during MyPLC startup, `plc_config.xml` is translated into several different languages—shell, PHP, and Python—so that scripts written in each of these languages can refer to the same underlying configuration. Most MyPLC scripts are written in shell, so the convention for shell variables predominates.

The variables that you should change immediately are:

- PLC_NAME: Change this to the name of your PLC installation.
- PLC_ROOT_PASSWORD: Change this to a more secure password.
- PLC_MAIL_SUPPORT_ADDRESS: Change this to the e-mail address at which you would like to receive support requests.
- PLC_DB_HOST, PLC_DB_IP, PLC_API_HOST, PLC_API_IP, PLC_WWW_HOST, PLC_WWW_IP, PLC_BOOT_HOST, PLC_BOOT_IP: Change all of these to the preferred FQDN and external IP address of your host system.

After changing these variables, save the file, then restart MyPLC with **service plc start**. You should notice that the password of the default administrator account is no longer `root`, and that the default site name includes the name of your PLC installation instead of PlanetLab. As a side effect of these changes, the ISO images for the boot CDs now have new names, so that you can freely remove the ones names after 'PlanetLab Test', which is the default value of PLC_NAME

Login as a real user

Now that myplc is up and running, you can connect to the web site that by default runs on port 80. You can either directly use the default administrator user that you configured in PLC_ROOT_USER and PLC_ROOT_PASSWORD, or create a real user

through the 'Joining' tab. Do not forget to select both PI and tech roles, and to select the only site created at this stage. Login as the administrator to enable this user, then login as the real user.

Installing nodes

Install your first node by clicking `Add Node` under the `Nodes` tab. Fill in all the appropriate details, then click `Add`. Download the node's configuration file by clicking `Download configuration file` on the *Node Details* page for the node. Save it to a floppy disk or USB key as detailed in [1].

Follow the rest of the instructions in [1] for creating a Boot CD and installing the node, except download the Boot CD image from the `/download` directory of your PLC installation, not from PlanetLab Central. The images located here are customized for your installation. If you change the hostname of your boot server (PLC_BOOT_HOST), or if the SSL certificate of your boot server expires, MyPLC will regenerate it and rebuild the Boot CD with the new certificate. If this occurs, you must replace all Boot CDs created before the certificate was regenerated.

The installation process for a node has significantly improved since PlanetLab 3.3. It should now take only a few seconds for a new node to become ready to create slices.

Administering nodes

You may administer nodes as `root` by using the SSH key stored in `/etc/planetlab/root_ssh_key.rsa`.

Example 6. Accessing nodes via SSH. Replace node with the hostname of the node.

```
ssh -i /etc/planetlab/root_ssh_key.rsa root@node
```

Besides the standard Linux log files located in `/var/log`, several other files can give you clues about any problems with active processes:

- `/var/log/pl_nm`: The log file for the Node Manager.
- `/vservers/pl_conf/var/log/pl_conf`: The log file for the Slice Creation Service.
- `/var/log/propd`: The log file for Proper, the service which allows certain slices to perform certain privileged operations in the root context.
- `/vservers/pl_netflow/var/log/netflow.log`: The log file for PlanetFlow, the network traffic auditing service.

Creating a slice

Create a slice by clicking `Create Slice` under the `Slices` tab. Fill in all the appropriate details, then click `Create`. Add nodes to the slice by clicking `Manage Nodes` on the *Slice Details* page for the slice.

A **cron** job runs every five minutes and updates the file `/plc/data/var/www/html/xml/slices-0.5.xml` with information about current slice state. The Slice Creation Service running on every node polls this file every ten minutes to determine if it needs to create or delete any slices. You may accelerate this process manually if desired.

Example 7. Forcing slice creation on a node.

```
# Update slices.xml immediately
service plc start crond

# Kick the Slice Creation Service on a particular node.
ssh -i /etc/planetlab/root_ssh_key.rsa root@node \
vserver pl_conf exec service pl_conf restart
```

Understanding the startup sequence

During service startup described in the Section called *QuickStart*, observe the output of this command for any failures. If no failures occur, you should see output similar to the following:

Example 8. A successful MyPLC startup.

```
Mounting PLC:                                              [  OK  ]
PLC: Generating network files:                             [  OK  ]
PLC: Starting system logger:                               [  OK  ]
PLC: Starting database server:                             [  OK  ]
PLC: Generating SSL certificates:                          [  OK  ]
PLC: Configuring the API:                                  [  OK  ]
PLC: Updating GPG keys:                                    [  OK  ]
PLC: Generating SSH keys:                                  [  OK  ]
PLC: Starting web server:                                  [  OK  ]
PLC: Bootstrapping the database:                           [  OK  ]
PLC: Starting DNS server:                                  [  OK  ]
PLC: Starting crond:                                       [  OK  ]
PLC: Rebuilding Boot CD:                                   [  OK  ]
PLC: Rebuilding Boot Manager:                              [  OK  ]
PLC: Signing node packages:                                [  OK  ]
```

If `/plc/root` is mounted successfully, a complete log file of the startup process may be found at `/plc/root/var/log/boot.log`. Possible reasons for failure of each step include:

- `Mounting PLC`: If this step fails, first ensure that you started MyPLC as root. Check `/etc/sysconfig/plc` to ensure that PLC_ROOT and PLC_DATA refer to the right locations. You may also have too many existing loopback mounts, or your kernel may not support loopback mounting, bind mounting, or the ext3 filesystem. Try freeing at least one loopback device, or re-compiling your kernel to support loopback mounting, bind mounting, and the ext3 filesystem. If you see an error similar to `Permission denied while trying to open /plc/root.img`, then SELinux may be enabled. See the Section called *Requirements* above for details.
- `Starting database server`: If this step fails, check `/plc/root/var/log/pgsql` and `/plc/root/var/log/boot.log`. The most common reason for failure is that the default PostgreSQL port, TCP port 5432, is already in use. Check that you are not running a PostgreSQL server on the host system.
- `Starting web server`: If this step fails, check `/plc/root/var/log/httpd/error_log` and `/plc/root/var/log/boot.log` for obvious errors. The most common reason for failure is that the default web ports, TCP ports 80 and 443, are already in use. Check that you are not running a web server on the host system.
- `Bootstrapping the database`: If this step fails, it is likely that the previous step (`Starting web server`) also failed. Another reason that it could fail is if PLC_API_HOST (see the Section called *Changing the configuration*) does not resolve to the host on which the API server has been enabled. By default, all

services, including the API server, are enabled and run on the same host, so check that PLC_API_HOST is either `localhost` or resolves to a local IP address. Also check that PLC_ROOT_USER looks like an e-mail address.

- `Starting crond`: If this step fails, it is likely that the previous steps (`Starting web server` and `Bootstrapping the database`) also failed. If not, check `/plc/root/var/log/boot.log` for obvious errors. This step starts the **cron** service and generates the initial set of XML files that the Slice Creation Service uses to determine slice state.

If no failures occur, then MyPLC should be active with a default configuration. Open a web browser on the host system and visit `http://localhost/`, which should bring you to the front page of your PLC installation. The password of the default administrator account `root@localhost.localdomain` (set by PLC_ROOT_USER) is `root` (set by PLC_ROOT_PASSWORD).

Files and directories involved in *myplc*

MyPLC installs the following files and directories:

1. `/plc/root.img`: The main root filesystem of the MyPLC application. This file is an uncompressed ext3 filesystem that is loopback mounted on `/plc/root` when MyPLC starts. This filesystem, even when mounted, should be treated as an opaque binary that can and will be replaced in its entirety by any upgrade of MyPLC.
2. `/plc/root`: The mount point for `/plc/root.img`. Once the root filesystem is mounted, all MyPLC services run in a **chroot** jail based in this directory.
3. `/plc/data`: The directory where user data and generated files are stored. This directory is bind mounted onto `/plc/root/data` so that it is accessible as `/data` from within the **chroot** jail. Files in this directory are marked with **%config(noreplace)** in the RPM. That is, during an upgrade of MyPLC, if a file has not changed since the last installation or upgrade of MyPLC, it is subject to upgrade and replacement. If the file has changed, the new version of the file will be created with a `.rpmnew` extension. Symlinks within the MyPLC root filesystem ensure that the following directories (relative to `/plc/root`) are stored outside the MyPLC filesystem image:
   - `/etc/planetlab`: This directory contains the configuration files, keys, and certificates that define your MyPLC installation.
   - `/var/lib/pgsql`: This directory contains PostgreSQL database files.
   - `/var/www/html/alpina-logs`: This directory contains node installation logs.
   - `/var/www/html/boot`: This directory contains the Boot Manager, customized for your MyPLC installation, and its data files.
   - `/var/www/html/download`: This directory contains Boot CD images, customized for your MyPLC installation.
   - `/var/www/html/install-rpms`: This directory is where you should install node package updates, if any. By default, nodes are installed from the tarball located at `/var/www/html/boot/PlanetLab-Bootstrap.tar.bz2`, which is pre-built from the latest PlanetLab Central sources, and installed as part of your MyPLC installation. However, nodes will attempt to install any newer RPMs located in `/var/www/html/install-rpms/planetlab`, after initial installation and periodically thereafter. You must run **yum-arch** and **createrepo** to update the **yum** caches in this directory after installing a new RPM. PlanetLab Central cannot support any changes to this directory.

- `/var/www/html/xml`: This directory contains various XML files that the Slice Creation Service uses to determine the state of slices. These XML files are refreshed periodically by **cron** jobs running in the MyPLC root.
- `/root`: this is the location of the root-user's homedir, and for your convenience is stored under `/data` so that your local customizations survive across updates - this feature is inherited from the **myplc-devel** package, where it is probably more useful.

4. `/etc/init.d/plc`: This file is a System V init script installed on your host filesystem, that allows you to start up and shut down MyPLC with a single command, as described in the Section called *QuickStart*.
5. `/etc/sysconfig/plc`: This file is a shell script fragment that defines the variables PLC_ROOT and PLC_DATA. By default, the values of these variables are `/plc/root` and `/plc/data`, respectively. If you wish, you may move your MyPLC installation to another location on your host filesystem and edit the values of these variables appropriately, but you will break the RPM upgrade process. PlanetLab Central cannot support any changes to this file.
6. `/etc/planetlab`: This symlink to `/plc/data/etc/planetlab` is installed on the host system for convenience.

Rebuilding and customizing MyPLC

The MyPLC package, though distributed as an RPM, is not a traditional package that can be easily rebuilt from SRPM. The requisite build environment is quite extensive and numerous assumptions are made throughout the PlanetLab source code base, that the build environment is based on Fedora Core 4 and that access to a complete Fedora Core 4 mirror is available.

For this reason, it is recommended that you only rebuild MyPLC (or any of its components) from within the MyPLC development environment. The MyPLC development environment is similar to MyPLC itself in that it is a portable filesystem contained within a **chroot** jail. The filesystem contains all the necessary tools required to rebuild MyPLC, as well as a snapshot of the PlanetLab source code base in the form of a local CVS repository.

Installation

Install the MyPLC development environment similarly to how you would install MyPLC. You may install both packages on the same host system if you wish. As with MyPLC, the MyPLC development environment should be treated as a monolithic software application, and any files present in the **chroot** jail should not be modified directly, as they are subject to upgrade.

- If your distribution supports RPM:

```
# rpm -U http://build.planet-lab.org/build/myplc-0_4-rc2/RPMS/i386/myplc-devel-0.4-2.
```

- If your distribution does not support RPM:

```
# cd /tmp
# wget http://build.planet-lab.org/build/myplc-0_4-rc2/RPMS/i386/myplc-devel-0.4-2.pl
# cd /
# rpm2cpio /tmp/myplc-devel-0.4-2.planetlab.i386.rpm | cpio -diu
```

Configuration

The default configuration should work as-is on most sites. Configuring the development package can be achieved in a similar way as for *myplc*, as described in the Section called *Changing the configuration*. **plc-config-tty** supports a *-d* option for supporting the *myplc-devel* case, that can be useful in a context where it would not guess it by itself. Refer to Appendix B for a list of variables.

Files and directories involved in *myplc-devl*

The MyPLC development environment installs the following files and directories:

- `/plc/devel/root.img`: The main root filesystem of the MyPLC development environment. This file is an uncompressed ext3 filesystem that is loopback mounted on `/plc/devel/root` when the MyPLC development environment is initialized. This filesystem, even when mounted, should be treated as an opaque binary that can and will be replaced in its entirety by any upgrade of the MyPLC development environment.
- `/plc/devel/root`: The mount point for `/plc/devel/root.img`.
- `/plc/devel/data`: The directory where user data and generated files are stored. This directory is bind mounted onto `/plc/devel/root/data` so that it is accessible as `/data` from within the **chroot** jail. Files in this directory are marked with **%config(noreplace)** in the RPM. Symlinks ensure that the following directories (relative to `/plc/devel/root`) are stored outside the root filesystem image:
  - `/etc/planetlab`: This directory contains the configuration files that define your MyPLC development environment.
  - `/cvs`: A snapshot of the PlanetLab source code is stored as a CVS repository in this directory. Files in this directory will *not* be updated by an upgrade of `myplc-devel`. See the Section called *Updating CVS* for more information about updating PlanetLab source code.
  - `/build`: Builds are stored in this directory. This directory is bind mounted onto `/plc/devel/root/build` so that it is accessible as `/build` from within the **chroot** jail. The build scripts in this directory are themselves source controlled; see the Section called *Building MyPLC* for more information about executing builds.
  - `/root`: this is the location of the root-user's homedir, and for your convenience is stored under `/data` so that your local customizations survive across updates.
- `/etc/init.d/plc-devel`: This file is a System V init script installed on your host filesystem, that allows you to start up and shut down the MyPLC development environment with a single command.

Fedora Core 4 mirror requirement

The MyPLC development environment requires access to a complete Fedora Core 4 i386 RPM repository, because several different filesystems based upon Fedora Core 4 are constructed during the process of building MyPLC. You may configure the location of this repository via the PLC_DEVEL_FEDORA_URL variable in `/plc/devel/data/etc/planetlab/plc_config.xml`. The value of the variable should be a URL that points to the top level of a Fedora mirror that provides the `base`, `updates`, and `extras` repositories, e.g.,

- `file:///data/fedora`
- `http://coblitz.planet-lab.org/pub/fedora`

- `ftp://mirror.cs.princeton.edu/pub/mirrors/fedora`
- `ftp://mirror.stanford.edu/pub/mirrors/fedora`
- `http://rpmfind.net/linux/fedora`

As implied by the list, the repository may be located on the local filesystem, or it may be located on a remote FTP or HTTP server. URLs beginning with `file://` should exist at the specified location relative to the root of the **chroot** jail. For optimum performance and reproducibility, specify PLC_DEVEL_FEDORA_URL=file:///data/fedora and download all Fedora Core 4 RPMS into `/plc/devel/data/fedora` on the host system after installing `myplc-devel`. Use a tool such as **wget** or **rsync** to download the RPMS from a public mirror:

Example 9. Setting up a local Fedora Core 4 repository.

```
# mkdir -p /plc/devel/data/fedora
# cd /plc/devel/data/fedora

# for repo in core/4/i386/os core/updates/4/i386 extras/4/i386 ; do
>     wget -m -nH --cut-dirs=3 http://coblitz.planet-lab.org/pub/fedora/linux/$repo
> done
```

Change the repository URI and **--cut-dirs** level as needed to produce a hierarchy that resembles:

```
/plc/devel/data/fedora/core/4/i386/os
/plc/devel/data/fedora/core/updates/4/i386
/plc/devel/data/fedora/extras/4/i386
```

A list of additional Fedora Core 4 mirrors is available at http://fedora.redhat.com/Download/mirrors.html.

Building MyPLC

All PlanetLab source code modules are built and installed as RPMS. A set of build scripts, checked into the `build/` directory of the PlanetLab CVS repository, eases the task of rebuilding PlanetLab source code.

Before you try building MyPLC, you might check the configuration, in a file named *plc_config.xml* that relies on a very similar model as MyPLC, located in */etc/planetlab* within the chroot jail, or in */plc/devel/data/etc/planetlab* from the root context. The set of applicable variables is described in Appendix B.

To build MyPLC, or any PlanetLab source code module, from within the MyPLC development environment, execute the following commands as root:

Example 10. Building MyPLC.

```
# Initialize MyPLC development environment
service plc-devel start

# Enter development environment
chroot /plc/devel/root su -

# Check out build scripts into a directory named after the current
# date. This is simply a convention, it need not be followed
# exactly. See build/build.sh for an example of a build script that
# names build directories after CVS tags.
DATE=$(date +%Y.%m.%d)
cd /build
cvs -d /cvs checkout -d $DATE build
```

```
# Build everything
make -C $DATE
```

If the build succeeds, a set of binary RPMS will be installed under `/plc/devel/data/build/$DATE/RPMS/` that you may copy to the `/var/www/html/install-rpms/planetlab` directory of your MyPLC installation (see the Section called *Installating and using MyPLC*).

Updating CVS

A complete snapshot of the PlanetLab source code is included with the MyPLC development environment as a CVS repository in `/plc/devel/data/cvs`. This CVS repository may be accessed like any other CVS repository. It may be accessed using an interface such as CVSweb[2], and file permissions may be altered to allow for fine-grained access control. Although the files are included with the `myplc-devel` RPM, they are *not* subject to upgrade once installed. New versions of the `myplc-devel` RPM will install updated snapshot repositories in `/plc/devel/data/cvs-%{version}-%{release}`, where `%{version}-%{release}` is replaced with the version number of the RPM.

Because the CVS repository is not automatically upgraded, if you wish to keep your local repository synchronized with the public PlanetLab repository, it is highly recommended that you use CVS's support for vendor branches to track changes, as described here[3] and here[4]. Vendor branches ease the task of merging upstream changes with your local modifications. To import a new snapshot into your local repository (for example, if you have just upgraded from `myplc-devel-0.4-2` to `myplc-devel-0.4-3` and you notice the new repository in `/plc/devel/data/cvs-0.4-3`), execute the following commands as root from within the MyPLC development environment:

Example 11. Updating /data/cvs from /data/cvs-0.4-3.

Warning: This may cause severe, irreversible changes to be made to your local repository. Always tag your local repository before importing.

```
# Initialize MyPLC development environment
service plc-devel start

# Enter development environment
chroot /plc/devel/root su -

# Tag current state
cvs -d /cvs rtag before-myplc-0_4-3-merge

# Export snapshot
TMP=$(mktemp -d /data/export.XXXXXX)
pushd $TMP
cvs -d /data/cvs-0.4-3 export -r HEAD .
cvs -d /cvs import -m "Merging myplc-0.4-3" -ko -I ! . planetlab myplc-0_4-3
popd
rm -rf $TMP
```

If there are any merge conflicts, use the command suggested by CVS to help the merge. Explaining how to fix merge conflicts is beyond the scope of this document; consult the CVS documentation for more information on how to use CVS.

More information : the FAQ wiki page

Please refer to, and feel free to contribute, the FAQ page on the Princeton's wiki [5].

A. Configuration variables (for *myplc*)

Listed below is the set of standard configuration variables and their default values, defined in the template `/etc/planetlab/default_config.xml`. Additional variables and their defaults may be defined in site-specific XML templates that should be placed in `/etc/planetlab/configs/`.

This information is available online within **plc-config-tty**, e.g.:

Example A-1. Advanced usage of plc-config-tty

```
<plc> # plc-config-tty
Enter command (u for usual changes, w to save, ? for help) V plc_dns
========== Category = PLC_DNS
### Enable DNS
# Enable the internal DNS server. The server does not provide reverse
# resolution and is not a production quality or scalable DNS solution.
# Use the internal DNS server only for small deployments or for testing.
PLC_DNS_ENABLED
```

List of the **myplc** configuration variables:

PLC_NAME

Type: string

Default: PlanetLab Test

The name of this PLC installation. It is used in the name of the default system site (e.g., PlanetLab Central) and in the names of various administrative entities (e.g., PlanetLab Support).

PLC_SLICE_PREFIX

Type: string

Default: pl

The abbreviated name of this PLC installation. It is used as the prefix for system slices (e.g., pl_conf). Warning: Currently, this variable should not be changed.

PLC_ROOT_USER

Type: email

Default: root@localhost.localdomain

The name of the initial administrative account. We recommend that this account be used only to create additional accounts associated with real administrators, then disabled.

PLC_ROOT_PASSWORD

Type: password

Default: root

The password of the initial administrative account. Also the password of the root account on the Boot CD.

PLC_ROOT_SSH_KEY_PUB

Type: file

Default: /etc/planetlab/root_ssh_key.pub

The SSH public key used to access the root account on your nodes.

PLC_ROOT_SSH_KEY

Type: file

Default: /etc/planetlab/root_ssh_key.rsa

The SSH private key used to access the root account on your nodes.

PLC_DEBUG_SSH_KEY_PUB

Type: file

Default: /etc/planetlab/debug_ssh_key.pub

The SSH public key used to access the root account on your nodes when they are in Debug mode.

PLC_DEBUG_SSH_KEY

Type: file

Default: /etc/planetlab/debug_ssh_key.rsa

The SSH private key used to access the root account on your nodes when they are in Debug mode.

PLC_ROOT_GPG_KEY_PUB

Type: file

Default: /etc/planetlab/pubring.gpg

The GPG public keyring used to sign the Boot Manager and all node packages.

PLC_ROOT_GPG_KEY

Type: file

Default: /etc/planetlab/secring.gpg

The SSH private key used to access the root account on your nodes.

PLC_MA_SA_NAMESPACE

Type: ip

Default: test

The namespace of your MA/SA. This should be a globally unique value assigned by PlanetLab Central.

PLC_MA_SA_SSL_KEY

Type: file

Default: /etc/planetlab/ma_sa_ssl.key

The SSL private key used for signing documents with the signature of your MA/SA. If non-existent, one will be generated.

PLC_MA_SA_SSL_CRT

Type: file

Default: /etc/planetlab/ma_sa_ssl.crt

The corresponding SSL public certificate. By default, this certificate is self-signed. You may replace the certificate later with one signed by the PLC root CA.

PLC_MA_SA_CA_SSL_CRT

Type: file

Default: /etc/planetlab/ma_sa_ca_ssl.crt

If applicable, the certificate of the PLC root CA. If your MA/SA certificate is self-signed, then this file is the same as your MA/SA certificate.

PLC_MA_SA_CA_SSL_KEY_PUB

Type: file

Default: /etc/planetlab/ma_sa_ca_ssl.pub

If applicable, the public key of the PLC root CA. If your MA/SA certificate is self-signed, then this file is the same as your MA/SA public key.

PLC_MA_SA_API_CRT

Type: file

Default: /etc/planetlab/ma_sa_api.xml

The API Certificate is your MA/SA public key embedded in a digitally signed XML document. By default, this document is self-signed. You may replace this certificate later with one signed by the PLC root CA.

PLC_NET_DNS1

Type: ip

Default: 127.0.0.1

Primary DNS server address.

PLC_NET_DNS2

Type: ip

Default:

Secondary DNS server address.

PLC_DNS_ENABLED

Type: boolean

Default: true

Enable the internal DNS server. The server does not provide reverse resolution and is not a production quality or scalable DNS solution. Use the internal DNS server only for small deployments or for testing.

PLC_MAIL_ENABLED

Type: boolean

Default: false

Set to false to suppress all e-mail notifications and warnings.

PLC_MAIL_SUPPORT_ADDRESS

Type: email

Default: root+support@localhost.localdomain

This address is used for support requests. Support requests may include traffic complaints, security incident reporting, web site malfunctions, and general requests for information. We recommend that the address be aliased to a ticketing system such as Request Tracker.

PLC_MAIL_BOOT_ADDRESS

Type: email

Default: root+install-msgs@localhost.localdomain

The API will notify this address when a problem occurs during node installation or boot.

PLC_MAIL_SLICE_ADDRESS

Type: email

Default: root+SLICE@localhost.localdomain

This address template is used for sending e-mail notifications to slices. SLICE will be replaced with the name of the slice.

PLC_DB_ENABLED

Type: boolean

Default: true

Enable the database server on this machine.

PLC_DB_TYPE

Type: string

Default: postgresql

The type of database server. Currently, only postgresql is supported.

PLC_DB_HOST

Type: hostname

Default: localhost.localdomain

The fully qualified hostname of the database server.

PLC_DB_IP

Type: ip

Default: 127.0.0.1

The IP address of the database server, if not resolvable by the configured DNS servers.

PLC_DB_PORT

Type: int

Default: 5432

The TCP port number through which the database server should be accessed.

PLC_DB_NAME

Type: string

Default: planetlab3

The name of the database to access.

PLC_DB_USER

Type: string

Default: pgsqluser

The username to use when accessing the database.

PLC_DB_PASSWORD

Type: password

Default:

The password to use when accessing the database. If left blank, one will be generated.

PLC_API_ENABLED

Type: boolean

Default: true

Enable the API server on this machine.

PLC_API_DEBUG

Type: boolean

Default: false

Enable verbose API debugging. Do not enable on a production system!

PLC_API_HOST

Type: hostname

Default: localhost.localdomain

The fully qualified hostname of the API server.

PLC_API_IP

Type: ip

Default: 127.0.0.1

The IP address of the API server, if not resolvable by the configured DNS servers.

PLC_API_PORT

Type: int

Default: 80

The TCP port number through which the API should be accessed. Warning: SSL (port 443) access is not fully supported by the website code yet. We recommend that port 80 be used for now and that the API server either run on the same machine as the web server, or that they both be on a secure wired network.

PLC_API_PATH

Type: string

Default: /PLCAPI/

The base path of the API URL.

PLC_API_MAINTENANCE_USER

Type: string

Default: maint@localhost.localdomain

The username of the maintenance account. This account is used by local scripts that perform automated tasks, and cannot be used for normal logins.

PLC_API_MAINTENANCE_PASSWORD

Type: password

Default:

The password of the maintenance account. If left blank, one will be generated. We recommend that the password be changed periodically.

PLC_API_MAINTENANCE_SOURCES

Type: hostname

Default:

A space-separated list of IP addresses allowed to access the API through the maintenance account. The value of this variable is set automatically to allow only the API, web, and boot servers, and should not be changed.

PLC_API_SSL_KEY

Type: file

Default: /etc/planetlab/api_ssl.key

The SSL private key to use for encrypting HTTPS traffic. If non-existent, one will be generated.

PLC_API_SSL_CRT

Type: file

Default: /etc/planetlab/api_ssl.crt

The corresponding SSL public certificate. By default, this certificate is self-signed. You may replace the certificate later with one signed by a root CA.

PLC_API_CA_SSL_CRT

Type: file

Default: /etc/planetlab/api_ca_ssl.crt

The certificate of the root CA, if any, that signed your server certificate. If your server certificate is self-signed, then this file is the same as your server certificate.

PLC_WWW_ENABLED

Type: boolean

Default: true

Enable the web server on this machine.

PLC_WWW_DEBUG

Type: boolean

Default: false

Enable debugging output on web pages. Do not enable on a production system!

PLC_WWW_HOST

Type: hostname

Default: localhost.localdomain

The fully qualified hostname of the web server.

PLC_WWW_IP

Type: ip

Default: 127.0.0.1

The IP address of the web server, if not resolvable by the configured DNS servers.

PLC_WWW_PORT

Type: int

Default: 80

The TCP port number through which the unprotected portions of the web site should be accessed.

PLC_WWW_SSL_PORT

Type: int

Default: 443

The TCP port number through which the protected portions of the web site should be accessed.

PLC_WWW_SSL_KEY

Type: file

Default: /etc/planetlab/www_ssl.key

The SSL private key to use for encrypting HTTPS traffic. If non-existent, one will be generated.

PLC_WWW_SSL_CRT

Type: file

Default: /etc/planetlab/www_ssl.crt

The corresponding SSL public certificate for the HTTP server. By default, this certificate is self-signed. You may replace the certificate later with one signed by a root CA.

PLC_WWW_CA_SSL_CRT

Type: file

Default: /etc/planetlab/www_ca_ssl.crt

The certificate of the root CA, if any, that signed your server certificate. If your server certificate is self-signed, then this file is the same as your server certificate.

PLC_BOOT_ENABLED

Type: boolean

Default: true

Enable the boot server on this machine.

PLC_BOOT_HOST

Type: hostname

Default: localhost.localdomain

The fully qualified hostname of the boot server.

PLC_BOOT_IP

Type: ip

Default: 127.0.0.1

The IP address of the boot server, if not resolvable by the configured DNS servers.

PLC_BOOT_PORT

Type: int

Default: 80

The TCP port number through which the unprotected portions of the boot server should be accessed.

PLC_BOOT_SSL_PORT

Type: int

Default: 443

The TCP port number through which the protected portions of the boot server should be accessed.

PLC_BOOT_SSL_KEY

Type: file

Default: /etc/planetlab/boot_ssl.key

The SSL private key to use for encrypting HTTPS traffic.

PLC_BOOT_SSL_CRT

Type: file

Default: /etc/planetlab/boot_ssl.crt

The corresponding SSL public certificate for the HTTP server. By default, this certificate is self-signed. You may replace the certificate later with one signed by a root CA.

PLC_BOOT_CA_SSL_CRT

Type: file

Default: /etc/planetlab/boot_ca_ssl.crt

The certificate of the root CA, if any, that signed your server certificate. If your server certificate is self-signed, then this file is the same as your server certificate.

B. Development configuration variables (for *myplc-devel*)

PLC_DEVEL_FEDORA_RELEASE

Type: string

Default: 4

Version number of Fedora Core upon which to base the build environment. Warning: Currently, only Fedora Core 4 is supported.

PLC_DEVEL_FEDORA_ARCH

Type: string

Default: i386

Base architecture of the build environment. Warning: Currently, only i386 is supported.

PLC_DEVEL_FEDORA_URL

Type: string

Default: file:///data/fedora

Fedora Core mirror from which to install filesystems.

PLC_DEVEL_CVSROOT

Type: string

Default: /cvs

CVSROOT to use when checking out code.

PLC_DEVEL_BOOTSTRAP

Type: boolean

Default: false

Controls whether MyPLC should be built inside of its own development environment.

Bibliography

[1] Mark Huang, *PlanetLab Technical Contact's Guide*[1].

Notes

1. http://fedora.redhat.com/Download/mirrors.html
2. http://www.freebsd.org/projects/cvsweb.html
3. http://ximbiot.com/cvs/wiki/index.php?title=CVS--Concurrent_Versions_System_v1.12.12.1:_Tracking_third-party_sources
4. http://cvsbook.red-bean.com/cvsbook.html#Tracking%20Third-Party%20Sources%20(Vendor%20Branches)
5. https://wiki.planet-lab.org/twiki/bin/view/Planetlab/MyplcFAQ

1. http://www.planet-lab.org/doc/TechsGuide.php