# User's Guide

This document is a guide for users developing applications on PlanetLab. This guide is not a comprehensive reference and is simply intended to help new users access PlanetLab for the first time.

## 1. Overview

This document is a guide for users developing applications on PlanetLab. This document assumes that the reader's site has been approved by the PlanetLab Consortium and that the site's nodes have been successfully installed.

## 2. Terminology

The following is a list of common terms used throughout this document.

- **Site.** A site is a physical location where PlanetLab nodes are located (e.g. Princeton University or HP Labs). Abbreviated versions of site names prefix all slice names.
- **Node.** A node is a dedicated server that runs components of PlanetLab services.
- **Slice.** A slice is a set of allocated resources distributed across PlanetLab. To most users, a slice means UNIX shell access to a number of PlanetLab nodes. PIs are responsible for creating slices and assigning them to their users. After being assigned to a slice, a user may then assign nodes to it. After nodes have been assigned to a slice, virtual servers for that slice are created on each of the assigned nodes. Slices have a finite lifetime and must be periodically renewed to remain valid.
- **Sliver.** A set of allocated resources on a single PlanetLab node.
- **Virtual Server (VServer).** Slivers are currently implemented as **Linux-Vservers**, which implements both namespace and performance isolation among slivers on a single machine. Network virtualization of slivers is implemented usig **VNET**. You may hear the terms slice, sliver, and VServer occasionally interchanged, but there are architectural distinctions between them.
- **Principal Investigator (PI).** The PIs at each site are responsible for managing slices and users at each site. PIs are legally responsible for the behavior of the slices that they create. Most sites have only one PI (typically a faculty member at an educational institution or a project manager at a commercial institution).

- **Technical Contact (Tech Contact).** Each site is required to have at least one Technical Contact who is responsible for installation, maintenance, and monitoring of the site's nodes.

- **User.** A user is anyone who develops and deploys applications on PlanetLab. PIs may also be users.

## 3. Getting Started With PlanetLab

To get started, register for a PlanetLab account by visiting the **Account Registration** page. Read the **Acceptable Use Policy (AUP)**. If you agree to abide by its terms, you will be allowed to register for an account.

If your site is not listed on the registration page, verify with your PI that your site has been approved to join PlanetLab. Otherwise, welcome to the PlanetLab Community!

If you ever forget your password, you can request that it be sent to you via e-mail by visiting the **Login** page and using the Reset Password form.

### 3.1. Create an SSH key

Before you can access any PlanetLab nodes, including those at your own site, you must create an SSH key pair for authentication purposes.

Remote access to PlanetLab nodes is restricted to SSH login using RSA authentication. RSA authentication is based on public-key cryptography. Encryption and decryption are performed with separate keys, and it is not possible to derive the decryption key from the encryption key.

To generate an SSH key pair, use the ssh-keygen program on a secure UNIX system:

```
ssh-keygen -t rsa -f ~/.ssh/id_rsa
```

This key must be in OpenSSH format. If the system is running a commercial UNIX and the first line of your .pub file does not look like

```
ssh-rsa AAAAB3Nza...
```

you need to convert your key. Consult the documentation that comes with the version of ssh-keygen that you are using.

ssh-keygen asks for a passphrase. Good passphrases are 10-30 characters long and are not simple sentences or otherwise easily guessable. ssh-keygen will generate two files: a private key file named id_rsa and a public key file named id_rsa.pub. Store the private key file id_rsa in a safe place, such as on a removable USB flash memory device. Upload the public key file id_rsa.pub to the PlanetLab website using the **Manage My Keys** page.

Protect your private key file! If your private key file is stolen and your account used for illegal activity, your site and PI may be held legally responsible. If you lose your private key file or suspect that it has been stolen or otherwise compromised, generate a new key pair with a new passphrase, upload the new public key to the website, and notify **PlanetLab Support** immediately.

### 3.2. Creating a slice

Once you have uploaded your SSH public key to the website, ask your PI to create a slice for you, or to associate your account with an existing slice.

The name of your slice will be prefixed with an abbreviated version of your site name and an underscore, e.g. princeton_test1. Once your slice has been created and you have been associated with it, you may assign nodes to the slice by using the **Manage Nodes** form.

Once you have been associated with a slice, it may take up to an hour for your slice to be created on all nodes and for your SSH public key to propagate to all of the nodes in your slice. Once your slice has been created on a node, you may login to it by following the instructions in **Getting Started With Your Slice**. Obviously, you will not be able to login to nodes that are down when your slice is created, and it may take a few minutes after a down node comes back up for your slice to be created on it.

#### 3.2.1. Troubleshooting

For a variety of reasons, including hardware failure and security compromise, nodes come and go. If a node in your slice is down permanently or temporarily, you will not be able to login to it. If the same node returns to service and you have not removed it from your slice, you will be able to access it when it becomes available again. Be aware that any node may be re-installed or even replaced at any time and that, consequently, data should not be permanently stored on any node.

If you are not able to login to a particular node that you are sure has been assigned to your slice, verify that you attempted to login with your slice name as the user name. Add -v to the ssh command line to print more information about the connection attempt.

```
ssh -v -l princeton_test1 -i ~/.ssh/id_rsa planetlab-1.cs.princeton.edu
```

Another common problem is mismatched host keys. The SSH host keys of PlanetLab nodes occasionally change. If you have a stale entry in your host key cache (by default, located at ~/.ssh/known_hosts), replace it with the current entry for the host which you may obtain from the **Known Hosts** page. You may merge this file directly into your ~/.ssh/known_hosts file if you wish.

If you are still unable to login, consult at least one of the community-supported **Status Tools** before asking **PlanetLab Support** for help. The node may simply be down for an extended period of time for known reasons. When contacting **PlanetLab Support** about a failed login attempt, include your slice name, the name of the node, and the verbose output of your ssh attempt.

#### 3.2.2. Limitations

Slices are created on demand and expire after two months. When a slice expires, it is destroyed. When a slice is destroyed, all files in the slice are removed on all nodes assigned to that slice.

The expiration date of a slice may be extended by using the **Renew Slice** form. There is no limit on the number of times a slice may be renewed, but the expiration date may never be set to more than two months into the future.

Slices also have resource limits. Disk space, memory usage, file descriptors, and bandwidth are controlled on a per-slice, per-node basis.

## 4. Getting Started With Your Slice

When your slice is created on a node, it is populated with a minimal **Fedora Core 2** Linux installation. To login to a node with SSH, provide your slice name as the login name (e.g. princeton_test1), the path to your private key file (e.g. ~/.ssh/id_rsa), and the node to login to (e.g. planetlab-1.cs.princeton.edu):

```
ssh -l princeton_test1 -i ~/.ssh/id_rsa planetlab-1.cs.princeton.edu
```

You will initially be logged in as a regular user with the same name as your slice name. Because your slice runs in a secure, isolated environment, you are granted restricted root access and may su to root, add new users, control services, install new packages, mount certain directories, and perform other actions that root is normally capable of.

However, most administrative and other privileged operations, such as hardware and network configuration, are disallowed even for root. Some interfaces, such as raw packet sockets, allow access to virtual hardware for compatibility, but these interfaces are tightly controlled. Consult the **VNET** documentation for more information about how network access is virtualized for your slice. To perform other privileged operations, such as accessing the filesystems of other slices and opening real raw sockets, use the **Proper API**.

Even though most administrative operations are disallowed to slices and slices' resource consumption is limited, it is possible for a slice to effectively disable a node, a site, or even portions of the Internet by hogging bandwidth, triggering automated Intrusion Detection Systems (IDSs), or attracting the attention of malicious hackers or

legal authorities. It is especially important that you understand your responsibility to keep your slice under control and secure from compromise. Vigilantly monitor and log your slice's activity and frequently change your SSH key.

### 4.1. Populating your slice

To save space, most of the binaries and system libraries installed in your slice by default are immutable hard links to reference copies. Consequently, you cannot modify these binaries. However, you can delete the binaries and replace them, which is the default upgrade behavior of most package managers such as RPM. Hard links do not count against your disk quota, but any files you replace, install, or otherwise create will.

To install additional standard packages in your slice, we recommend using yum. For example, to install emacs:

```
yum install emacs
```

To bring your slice up-to-date with the latest versions of all packages:

```
yum update
```

Consult the yum(8) man page or the **YUM home page** for more advanced usage. If you will be performing batch installations on all of the nodes in your slice with yum, please consider editing the default /etc/yum.conf file in your slice and replacing the FedoraCore2Base and FedoraCore2Updates entries with **public mirrors** to reduce the load on the PlanetLab boot server.

### 4.2. Building and deploying your application

Once your slice has been created and populated on a few nodes, you may begin deploying your application. Unless your application relies on PlanetLab extensions, to ensure that your application will run on PlanetLab, we recommend compiling and testing it on a local set of Fedora Core 2 development machines before deploying it.

The most straightforward way of deploying your application to a single node is with scp:

```
scp -i ~/.ssh/id_rsa -r test1
princeton_test1@planetlab-1.cs.princeton.edu:
```

If your slice were named princeton_test1, the above command would recursively copy the local directory test1 to your home directory on planetlab-1.cs.princeton.edu.

rsync may be used to copy just those files that do not exist or have changed since the last time they were copied:

```
rsync -a -e "ssh -l princeton_test1 -i ~/.ssh/id_rsa"
test1 planetlab-1.cs.princeton.edu:
```

PlanetLab users have contributed a variety of other tools and applications for managing application deployment and execution. Notable among these are **Parallel SSH**, the **PlanetLab Slice Deploy Toolkit**, and **vxargs**. A complete list of contributed software may be found on the **ContributedSoftware** page on the **PlanetLab Wiki**.

### 4.3. Configuring a service for automatic startup

Nodes reboot periodically for scheduled maintenance and upgrades. If your application is a long-running service, you may want to configure it to start automatically after the node boots and to shut down cleanly before the node halts.

To do so, after deploying your application to a node, set up a System V Init script for it in /etc/rc.d/init.d/. We recommend using one of the existing scripts as a template so that the script works with chkconfig. You may configure your service to start and stop in a particular order relative to other services by modifying the line beginning with "# chkconfig:" near the top of the script. The first number is the list of runlevels the script should be run in; the second number is the relative order in which it should start; and the third number is the relative order in which it should stop. Your slice will always start in runlevel 3 and stop in runlevel 0; list the contents of /etc/rc3.d/ and /etc/rc0.d/ to determine the current service order.

After creating the script, enable it with chkconfig:

```
chkconfig --add test1
chkconfig test1 on
```

After installing the script, you may manually start and stop your service with service:

```
service test1 start
service test1 stop
```

Which is how the system will call your script after it boots and before it halts.

If you do not care to write a graceful shutdown procedure, you may simply edit /etc/rc.d/rc.local and insert appropriate code to start your service. rc.local is called last during startup.

## 5. Slice Maintenance

### 5.1. Slice API

Nodes come and go. To ease the chore of adding and deleting nodes from your slice, you may choose to use a **programmatic interface to the slice database** instead of the website. The website itself uses the programmatic interface to perform all operations related to slice maintenance. Any operation related to your slice that you

are allowed to perform on the website, can also be performed using the programmatic interface.

### 5.2. Admin API

Similarly, any function regarding your account, as well as a host of other functions for accessing PlanetLab Central data including node and site lists, may be called through a **programmatic interface to the PLC administrative database**.

### 5.3. Slice expiration

Please remember that your slice has a finite lifetime and must be renewed periodically. If you no longer use a slice, let it expire, or preferably delete it, to free its resources for other users to use.

Your slice's expiration date may be reset by an administrator for a number of reasons. If the description of your slice is insufficient, or the link to your project website is broken, an administrator may set your slice to expire within a few days, which will force you to **update the description**. Do not provide a bogus description again; if you continue to provide insufficient descriptions, your slice will be deleted.

Slice creation and renewal may also be temporarily disabled for your site if your site's nodes have been down or unreachable for several weeks, and multiple attempts to contact your site's PI(s) and Technical Contact(s) have all failed. If this happens, contact your site's PI(s) and Technical Contact(s) and ask them to bring up your site's nodes. If you believe that your site's nodes are up and reachable, contact **PlanetLab Support**. Otherwise, visit your site's **Site Details** page to find out more about your site's nodes, and how to contact your site's PI(s) and Technical Contact(s).

### 5.4. Etiquette and netiquette

The isolated execution environment in which your slice runs does not allow you to monitor other slice's processes with standard tools such as top or ps. Be assured that nodes are almost never idle, however. Although the system limits your slice's disk, memory, CPU, I/O, and bandwidth usage, it is still possible to interfere with other slices' activities by "hogging" a particular resource.

A few obvious recommendations are listed below.

- Maintain a web page that describes your slice's purpose, the researchers in charge of it, and their contact information. Link to this web page from your **Slice Description**.
- Plan for success. If your service is successful, consider how your code will scale, how you will support it, and how you will maintain it as it grows.

- Thoroughly debug your code, especially its failure modes, before deploying it. Continually monitor and log its execution. Use a distributed status tool like **CoMon** to monitor the health of nodes on which your slice is deployed and investigate any anomalies.

- Do not perform systematic port scans or other experiments that may be interpreted as an attack or other type of illegal activity. Such activities are violations of the **AUP**, and you may be held legally responsible for them.

### 5.5. Other resources

The PlanetLab Community consists of over 1,000 users at over 200 sites. More often than not, someone on the **PlanetLab Users mailing list** will be able to answer any question you might have regarding application compilation, deployment, and execution.

If you are having technical difficulty with a node, with your slice, with your account, or with the website, contact **PlanetLab Support**. If you maintain a long-running service, we urge and will eventually require you to participate in the support system to answer questions and complaints regarding your service.

The current execution environment for the PlanetLab platform is Linux, so a working knowledge of Linux and UNIX APIs, interfaces, and conventions is essential. However, assisting users with general Linux or UNIX questions is beyond the scope and capabilities of PlanetLab Support. If you have such a question, please try to resolve it through **Google** or through the archives of the **PlanetLab Users mailing list** before contacting PlanetLab Support.

## 6. Document Revision History

| | | | |
|---|---|---|---|
| Revision 1.0 | 02.18.2005 | Initial draft. | Mark L. Huang |
| Revision 2.0 | 02.01.2006 | Update for PlanetLab 4.0 release. | Marc E. Fiuczynski |