

1 Introduction

This tutorial is designed for users who are new to the PlanetLab system. It is a step-by-step guide that will take you all the way from creating a user account to running a custom software package on a world-wide collection of nodes. This guide is not intended as a replacement for the primary [PlanetLab documentation](https://www.planet-lab.org/doc)¹. Rather, its purpose is to provide a new user who is unfamiliar with the inner workings of planetlab with a functional system. You can use these instructions to get up and running quickly, and subsequently modify this procedure (with the aid of primary PlanetLab documentation, where appropriate) with little effort to suit the needs of your own custom application.

This tutorial will walk you through the steps of deploying a planet-wide *Hello World* demonstration application. In this demonstration, a script will be deployed to one node in each active site in the PlanetLab network. This script, when run on each node, will gather information about that node's site and send this information back to a central web server. The central web server will collect the data, and export a KML file containing the locations of every site for display with the Google Earth™ mapping service².

1.1 Hello, World!

The *Hello World* demonstration is composed of 2 pieces. The first piece is a small python script (`phonehome.py`) that is deployed to the nodes. This script represents your custom PlanetLab application. In modifying this procedure for your own purposes, your application should be substituted for this file. You will need python 2.4 in order to access to PlanetLab API as described in this tutorial, whether you are running the provided *Hello World* application or your own custom application.

The second piece of the *Hello World* demonstration is a collection of php scripts that run on your central web server. These scripts collect information from the various nodes that report in, and format that information into a KML file for mapping. This component is only useful for the *Hello World* demonstration itself. You do not need a web server with php support in order to deploy and run a generic application on PlanetLab. You will, however, need http access to a directory on your local filesystem in order to use the CoDeploy program described in this tutorial.

1.2 Requirements

This tutorial assumes that you are using a Linux operating system with `OpenSSH` and access to standard UNIX tools such as `tar`. This list summarizes the software requirements you will need in order to follow this tutorial. Everything in the list is required in order to follow the tutorial to the letter. Only underlined items are required to deploy and run your own application using this method.

1. Python 2.3 or higher.

¹<https://www.planet-lab.org/doc>

²*Google* and *Google Earth* are trademarks of Google Inc.

2. Write access to a directory in your local filesystem that is also readable via a web server.
3. A web server with PHP (≥ 4.1) support.
4. [Google Earth™ mapping service](#)³

1.3 Conventions Used in This Document

In order to follow this tutorial, you will need to type several commands into a UNIX/Linux shell prompt. Commands that must be entered at a shell prompt will be displayed within a box with a gray background. Some commands may span multiple lines in this document, but should be entered on a single line.

You will also need to create several scripts, using code provided in this tutorial. Code that should be copied into a script and saved will be displayed within a box with a white background.

2 Getting Started with PlanetLab

This section will walk you through the initial procedures of setting up your PlanetLab user account, configuring your public key, getting a slice, and adding nodes to it.

2.1 Creating a User Account

The first thing you need to do before you can use PlanetLab is create your PlanetLab account. This process is simple and can be accomplished on the [PlanetLab website](#)⁴. Before registering for an account, you should read the [PlanetLab Acceptable Use Policy](#)⁵ (AUP). Once you have read and understand the AUP, and agree to its terms, return to the PlanetLab homepage. Underneath the login boxes, click the “Create an account” link. Fill in the required information, making sure to select your site correctly. (N.B. Your email address will function as your PlanetLab user name.) Once you click the “Register” button, your request will be sent to your site’s Principal Investigator (PI) for approval. When your PI approves creation of your account, you will receive a confirmation email. At this time, you may continue with the tutorial.

This tutorial will assume that your username is “user” and your password is “pass”. You should replace these values with your actual username and password whenever you encounter them in this tutorial.

You will need to ssh public-key authentication to connect to nodes in the PlanetLab network. If you do not already have an ssh keypair, or would like to create a new one specifically for PlanetLab, perform the following commands. You will need to have ssh-agent configured properly to avoid having to type your passphrase many times when using automated tools such as CoDeploy. This tutorial will assume that `~/.ssh/id_rsa` is the private key that you will use for PlanetLab authentication.

³<http://earth.google.com/>

⁴<http://www.planet-lab.org/>

⁵<http://www.planet-lab.org/aup/>

```
$ ssh-keygen -t rsa -f ~/.ssh/id_rsa
Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase): <enter some passphrase>
Enter same passphrase again: <enter your passphrase again?>
```

Once you have created your ssh keypair, you will need to upload your public key to the PlanetLab database. You can do this easily through the PlanetLab website. Go to the PlanetLab home-page and log in using your newly created account. In the navigation box on the left, click “My Account.” In the “Keys” section, choose “Manage Keys.” Click “Browse” and navigate to your *public* key file (i.e. `~/.ssh/id_rsa.pub`). Click “Upload.” You should see your appear in the list.

2.2 Getting a Slice

All access to PlanetLab resources occurs through the “slice” abstraction. A slice is a collection of resources distributed across multiple PlanetLab nodes. When a node is added to a slice, a virtual server for that slice is created on that node. When a node is removed from a slice, that virtual server is destroyed.

Your PI is in charge of managing slices at your site. You will need to contact your PI directly and ask him/her to either create a new slice for you, or add your user account to an existing slice.⁶ This change may require several minutes to take effect. Once your account has been added to a slice, you will be able to view that slice by choosing “Slices” in the navigation box on the PlanetLab website. (N.B. You will need to log out and log back in after you slice has been created in order for the website to update properly.)

Slice names are typically prefixed with your site’s base name. This tutorial will assume that your slice is named `planetlab_hello`. You should replace this value with your slice’s actual name whenever you encounter it in this tutorial.

2.3 Adding Nodes to Your Slice

You must add nodes to your slice in order to gain access to them. The simplest way to add nodes is to select them one-by-one via the PlanetLab website. However, for the *Hello World* demonstration you will use a PlanetLab project called [CoMon](#)⁷ to obtain a list of nodes that are currently alive, and filter them to only select one per site. Then you will use the PlanetLab API to automatically add that list of nodes to your site.

You will need a local directory in which to store working files. This tutorial will assume that `~/planetlab/hello_world/` is that directory. Replace this value with whatever path you are using on your system.

CoMon queries are run simply by passing a variety of HTTP PUT variables to the CoMon cgi script. The main CoMon website describes a number of sample queries. For the *Hello*

⁶This tutorial will assume that you have been given a new, empty slice. If you have instead been granted access to an existing slice, you should probably refrain from actually following the procedure described in the remainder of this tutorial, as you risk interfering with an existing project.

⁷<http://comon.cs.princeton.edu/>

World demonstration, you will want to run this query, which selects for live nodes, filters 1 per site, and displays the results in a simple text format for easy parsing. Save the resulting file into your PlanetLab working directory as `nodes.txt`.

```
http://comon.cs.princeton.edu/status/tabulator.cgi?table=
table_nodeviewshort&format=nameonly&persite=1&
select='resptime>0'
```

Now that you have a list of nodes that you want to add, you can use the API to automatically add them to your slice. Two methods of accessing the API are described here. Either will work fine for the purposes of this demonstration.

2.3.1 Accessing the API with the PlanetLab Shell

The simplest way to access the PlanetLab API is through the PlanetLab Shell, `plcsh`. You will need to obtain `plcsh` from the PlanetLab CVS repository. In your PlanetLab working directory, run the following commands in a terminal window to download and compile the PlanetLab Shell.

```
$ cvs -d :pserver:anon@cvs.planet-lab.org:/cvs checkout new_plc_api
$ cd new_plc_api
$ make
```

Once you have successfully compiled the PlanetLab Shell application, you can use it to add nodes to your slice in the following manner. The PlanetLab Shell uses a python syntax, and includes a full python interpreter, so any valid python code can be run through the PlanetLab Shell.

```
$ ./plcsh -u username -r user  <-- enter your actual username here
Password: <-- enter youe password here
[ user]>>> node_list = [line.strip() for line in open("../nodes.txt")]
[ user]>>> AddSliceToNodes("your_slice_name", node_list)
[ user]>>> exit
```

2.3.2 Accessing the API with Python

If you do not wish to use the PlanetLab Shell, the API functions may also be accessed through any scripting language that supports XMLRPC. An example is presented here in Python.

In your PlanetLab working directory, run the following commands in a terminal window. The final python command should return the value “1” on success.

```
$ python
>>> import xmlrpclib
>>> api_server = \
... xmlrpclib.ServerProxy('https://www.planet-lab.org/PLCAPI/')
>>>
>>> auth = {}
>>> auth['Username'] = "user" <-- substitute your actual username here
>>> auth['AuthString'] = "pass" <-- substitute your actual password here
>>> auth['AuthMethod'] = "password"
>>>
>>> node_list = [line.strip() for line in open("nodes.txt")]
>>>
>>> api_server.AddSliceToNodes(auth, \
... "your_slice_name", node_list)
>>> ^D
```

You will now need to wait some time, at least 15 minutes but possibly longer, for virtual servers to be allocated to your slice on all the nodes. While you are waiting, you can proceed to the next step and install the CoDeploy software. You will not be able to run it successfully, however, until your virtual servers have been allocated.

3 Deploying Software to Your Nodes

There are many ways of deploying software to all of your nodes. There are several tools available for PlanetLab that allow you to run parallel ssh commands on a collection of nodes. However, you will likely find that (if your slice contains more than a handful of nodes) instructing all of your nodes to simultaneously download a file will create sudden, excessive load for the central web server providing that file. If this is a third party web server, such excessive load may violate the PlanetLab AUP.

Alternatively, you can use a content distribution network to deploy software to your nodes. This technique will avoid creating large, sudden load spikes on an individual content server. The [CoDeeN](http://codeen.cs.princeton.edu/)⁸ project, a content distribution network built on top of PlanetLab, provides a tool named [CoDeploy](http://codeen.cs.princeton.edu/codeploy)⁹ specifically for this purpose.

3.1 Installing CoDeploy

The [CoDeploy website](http://codeen.cs.princeton.edu/) provides detailed instructions for downloading and installing the CoDeploy software. A brief description of the actions necessary to use CoDeploy for the

⁸<http://codeen.cs.princeton.edu/>

⁹<http://codeen.cs.princeton.edu/codeploy>

purposes of this demonstration are provided below. Download the CoDeploy tarball into your working directory, and then execute the following commands in that directory. (N.B. your nodes.txt file should already be in your working directory.)

```
$ tar xzf codeploy.tar.gz
$ cd codeploy
$ make
$ export MQ_NODES='../nodes.txt'
$ export MQ_SLICE='planetlab_hello' <-- substitute your slice name here
$ echo "StrictHostKeyChecking no" >> ~/.ssh/config
```

The last line above will stop ssh from attempting to verify the host key of every node to which it connects. Without this option, you would be prompted to access the node's host key every time you connect to a new PlanetLab node. This configuration will leave you vulnerable to a potential 'man-in-the-middle' attack, however. If you are more concerned about security, you can download a complete list of host keys for all PlanetLab nodes from https://www.planet-lab.org/db/nodes/known_hosts.php and manually merge this list with your existing `~/.ssh/known_hosts` file.

3.2 Hello World Central Web Application

If you wish to run the entire *Hello World* demonstration, you will need to set up a central web application as described in this section. Nothing described in this section is necessary for general use of the PlanetLab system. If you are only using this tutorial to deploy your own custom application, you can skip ahead to Section 3.3.

To set up the central *Hello World* web application, you will need a web server with php (version ≥ 4.1) installed. This tutorial will assume that the url of such a web site is `http://www.your.url/~username/hw_demo/` and that you can access this directory via the local path `~/public_html/hw_demo/`.

You will need to save the following two php scripts (`phonehome.php` and `phonelog.php`) into this directory. You will also need to create the file `phonehome.txt` and set it to be writable by the user account under which the web server process is running. This can be accomplished simply by executing the following commands in the `hw_demo` directory.

```
$ touch phonelog.txt
$ chmod 666 phonelog.txt
```

This procedure will make the `phonelog.txt` file world-writable. Although this is generally not a good thing, setting permissions so that it is writable only by the web server user can be a fairly complicated procedure without root access, depending on your web server's configuration. If you understand your web server's security implementation well enough to do this a better way, then do so. The only requirement for the *Hello World* application is

that `phonehome.php`, when run through the web server, can write to `phonelog.txt`.
Save the following two scripts as `phonelog.php` and `phonehome.php`, respectively.

```
<?
### phonelog.php

header('Content-type: application/vnd.google-earth.kml+xml');
header('Content-disposition: attachment;
    filename="hello_world.kml"');
echo '<?xml version="1.0" encoding="UTF-8"?>' . "\n";
echo '<kml xmlns="http://earth.google.com/kml/2.0">' . "\n";
echo '<Document>' . "\n";

$phonelog = file("phonelog.txt");

foreach ($phonelog as $entry) {
    $fields = explode("\t", trim($entry));
    if ($fields[4] == "None" || $fields[5] == "None") continue;
    ?>
    <Placemark>
        <Snippet><?= htmlspecialchars($fields[3]) ?></Snippet>
        <name><?= htmlspecialchars($fields[1]) ?></name>
        <LookAt>
            <longitude><?= $fields[5] ?></longitude>
            <latitude><?= $fields[4] ?></latitude>
            <range>1000000</range>
        </LookAt>
        <visibility>1</visibility>
        <Point>
            <extrude>1</extrude>
            <altitudeMode>relativeToGround</altitudeMode>
            <coordinates>
                <?= $fields[5] . "," . $fields[4] ?>,0
            </coordinates>
        </Point>
    </Placemark>
    <? }
?>
</Document>
</kml>
```

```

<?
### phonehome.php

    if (isset($_GET['reset'])) {
        $phonelog = fopen("phonelog.txt", "w");
        fclose($phonelog);
        echo "Cleared Phone Log.";
        die();
    }
    if (!isset($_POST['site_id'])) die;
    $existinglog = file("phonelog.txt");
    foreach ($existinglog as $entry) {
        $f = explode("\t", trim($entry));
        if ($f[0] == $_POST['site_id']) die;
    }

    $phonelog = fopen("phonelog.txt", "a");
    fwrite($phonelog, $_POST['site_id'] . "\t" . $_POST['name'] .
        "\t" . $_POST['login_base'] . "\t" . $_POST['url'] .
        "\t" . $_POST['latitude'] . "\t" . $_POST['longitude'] .
        "\n");
    fclose($phonelog);
?>

```

3.3 Deploying the *Hello World* Script

In order to use CoDeploy, you will need access to a local directory that is also readable via HTTP. The software you wish to deploy should be stored in this directory. CoDeploy may also create temporary files in this directory during the deployment. This tutorial will assume that `~/public_html/helloworld` is a local directory that is also accessible via `http://www.your.url/~username/helloworld`.

The *Hello World* demonstration consists of a single python script that should be deployed to all your nodes. Simply save the code below into a file called `phonehome.py` in the `~/public_html/helloworld/` directory. You should edit the value of `phonehome_url` to reflect the location of the *Hello World* central web application that you configured in section 3.2.

```
#!/usr/bin/python
### phonehome.py
### Hello World demonstration script

phonehome_url = "http://www.your.url/~username/hw_demo/phonehome.php"

import sys, urllib, xmlrpclib, socket

api_server = xmlrpclib.ServerProxy('https://www.planet-lab.org/PLCAPI/')

auth = {}
auth['AuthMethod'] = "anonymous"
auth['Role'] = "user"

hostname = socket.gethostname()
query = api_server.GetNodes(auth,
                             {'hostname': hostname}, ['site_id'])
site_id = query[0]['site_id']

site_info = api_server.GetSites(auth,
                                 {'site_id': site_id}, ['site_id', 'name', 'url',
                                                       'latitude', 'longitude', 'login_base'])

site_info = urllib.urlencode(site_info[0])
urllib.urlopen(phonehome_url, site_info)
```

To deploy the software, simply run CoDeploy with the appropriate directories as arguments. The third argument to CoDeploy is the directory in which to store the software (in your home directory) on each node.

```
$ ./codedeploy ~/public_html/helloworld \
> http://www.your.url/~username/helloworld/ hello
```

3.4 Running Your Software

Now that your software is deployed, you probably want to run it on all of your nodes. The MultiQuery tool provided with CoDeploy provides a simple way to run a command simultaneously on a large number of nodes via ssh. For the *Hello World* demonstration you will use MultiQuery to register a cronjob on each node that will run the `phonehome.py` script every 2 minutes. This will generate a large number of requests for your central *Hello World* web application, so you probably do not want to leave the cronjob active while you are not

viewing the demo.

To register the *Hello World* on each node, run the following command in the directory in which you installed CoDeploy.

```
$ ./multiquery 'echo "*/* * * * python hello/phonehome.py" |  
  crontab -; sudo /sbin/service crond start'
```

To stop the nodes from continuing to phone home when you are finished with the demo you can simply stop the cron daemon on each node, again using MultiQuery.

```
$ ./multiquery 'sudo /sbin/service crond stop'
```

3.5 Viewing the Demo

After a few minutes, most of your nodes should have phoned home. A list of the sites that have responded can be found at http://www.you.url/~username/hw_demo/phonelog.txt. The `phonelog.php` script will convert that information into a KML file, containing the latitude and longitude coordinates of each site. This file can be used to display the location of each of your nodes in a mapping program, such as the Google Earth™ mapping service.