

File distribution

Table of contents

1 Push versus pull.....	2
2 Push via Scp.....	2
3 Pull via fileutil: copy.....	3

DRAFT: May need corrections!

Ant users have several choices for distributing files. Ant's [optional tasks](#) include [Scp](#) and [Ftp](#) to transfer files. There is even the core task, [Get](#), which one can use to pull files from a web repository. If you are a WebDAV user, there are [WebDAV ant tasks](#). Beyond these tasks one can use the [Exec](#) task as a general way to invoke an external tool like rsync.

Before choosing what task or tool is the best to move a file from a central host to a target host, it's best to first establish what distribution model is ideal: push or pull.

1. Push versus pull

The table below shows two models for distribution, push and pull. In either case, a central admin host is assumed to be the store house of files that will be distributed to target hosts.

Push	With the push model, the process is completely managed at the central admin host, and the target hosts passively receive files. Typically, Scp or rsync are used with the push method.	
Pull	With the pull model, the process is coordinated at the central admin host, where each target is told to pull files from the admin host. Ftp, Scp, HTTP Get, and rsync are commonly used to pull files from a repository during distribution to target hosts.	

CTL will support either distribution model but often times we have found the pull model is more scalable.

The following section show recipes using the push and pull models described above.

2. Push via Scp

The first example, describes how to use the Scp task to push a list of files to a set of nodes. The ant-contrib [For](#) task is used to loop over the set of hosts.

Additionally, the command uses several parameters to make the command as generic as possible.

```
<command name="sync-files-push" description="synchronize files via
push."
    command-type="AntCommand" is-static="true">
  <implementation>
    <property name="atChar" value="@"/>
    <for list="{opts.hosts}" param="host">
      <sequential>
        <scp
          todir="{opts.user}{atChar}@{host}::{opts.destdir}"
          keyfile="{user.home}/.ssh/id_dsa"
          passphrase=""
          <fileset includes="{opts.files}"/>
        </scp>
      </sequential>
    </implementation>
    <opts>
      <opt parameter="files" property="opts.files" type="string"
required="true"/>
      <opt parameter="hosts" property="opts.hosts" type="string"
required="true"/>
      <opt parameter="destdir" property="opts.destdir" type="string"
required="true"/>
      <opt parameter="user" property="opts.user" type="string"
required="false"
        defaultproperty=user.name/>
    </opts>
  </command>
```

To run "sync-files-push" via the `ctl` command line type:

```
ctl -m mymodule -c sync-files-push -- \
  -hosts target1,target2 -files file1,file2 -destdir /appdir
```

Note, one can use `rsync` or other remote copy commands instead of `Scp` but use the same implementation pattern, by substituting the alternative for the `Scp` task used in this example.

3. Pull via `fileutil`: copy

There are roughly two approaches to implementing a pull-based distribution model.

1. Coordinating modules. This approach assumes there are two modules that play a master slave relationship. On the admin host (the master) a module has a command that initiates the pull action on the target nodes (the slaves).
2. Module and external remote tool. This approach is similar to the one above except the invoked action is a call to an external tool that resides on the remote nodes.

Either approach can be used, but we have found the first approach is more flexible and better supports the evolution of the procedure that is executed on the target hosts.

This example will assume two modules. Firstly, the "fileutil" module, a module CTL offers an alternative to directly using the Ant tasks mentioned above and in a manner that fits well with the CTL module framework plays the role of "slave". Secondly, a custom module and command described here will play the role of "master".

The fileutil module provides a high level function for copying files both locally and also from a repository to a local directory. The fileutil copy command provides a number of options useful when doing deployment:

```
copy [-backup] -file <> [-overwrite] [-suffix <.backup>] -target <>
```

Besides creating a backup if a file by that name already exists, it will also download a file via HTTP GET, if the file argument contains the http:// prefix. The pull model described here, will take advantage of the HTTP GET feature.

The job of the coordinating command that executes on the central admin host is to call the copy command in the "fileutil" module across the desired hosts. The distributed command dispatching can be handled by CTL's nodedispatch strategy.

```
<command name="sync-files-pull" description="synchronize files via
pull."
  command-type="AntCommand" is-static="true">
  <implementation>
  <property name="atChar" value="@"/>
  <for list="{opts.files}" param="file">
    <sequential>
      <controller>
        <execute strategy="nodedispatch">
          <context depot="{context.depot}"/>
          <command name="copy" module="fileutil"/>
          <arg line="-file @{file} -dest ${opts.destdir}"/>
          <nodeset>
            <include hostname="{opts.hosts}"/>
          </nodeset>
        </execute>
      </controller>
    </sequential>
  </implementation>
  <opts>
    <opt parameter="files" property="opts.files" type="string"
required="true"/>
    <opt parameter="hosts" property="opts.hosts" type="string"
required="true"/>
    <opt parameter="destdir" property="opts.destdir" type="string"
required="true"/>
    <opt parameter="user" property="opts.user" type="string"
required="false"
defaultproperty=user.name/>
```

```
</opts>  
</command>
```

Usage would look as follows:

```
ctl -m mymodule -c sync-files-pull --\  
-files file1,file2,file3 -hosts target1,target2,target3 -destdir  
/appdir
```

The code sample above shows "sync-files-pull" command uses the controller CTL Ant task to call fileutil->copy command with a nodeset include filter set to the list of hosts defined in the -hosts parameter.

The fileutil module offers some extra features beyond the simple HTTP GET but you can choose alternatives to the fileutil module, using another task or tool that downloads files from a central repository.