

Remote execution

Table of contents

1 Remote execution via ctl-exec.....	2
2 Remote execution via nodedispatch.....	2

DRAFT: May need corrections!

Managing deployment and operations in a scalable and efficient manner must always include executing actions across a set of hosts.

CTL provides two means to execute distributed actions. Firstly, `ctl-exec` allows you to execute any ad hoc command. Secondly, defined commands can be executed across a set of hosts via the `nodedispatch` mechanism available in the CTL `controller` ant task. The following recipes show both alternatives.

Note:

These examples assume you understand have completed the CTL [project setup](#) and that you have defined your [nodes deployments](#) you wish to remotely manage.

1. Remote execution via `ctl-exec`

CTL's [ctl-exec](#) command is a convenient way to execute a command across a set of nodes.

The following example executes the `whoami` command across all the Linux hosts except for the localhost:

```
<command name="linux-whoami" description="dispatch whoami command to
linux hosts"
  command-type="AntCommand" is-static="true">
  <implementation>
    <exec executable="ctl-exec">
      <arg line="-I os-name=Linux -X localhost -- whoami" />
    </exec>
  </implementation>
</command>
```

The source code in this example show Ant's [exec](#) task calling the `ctl-exec` command passing in its arguments via the `arg` element.

The command would be run like so:

```
ctl -m mymodule -c dispatch-whomai
```

This recipe is suitable when you need to execute arbitrary system commands.

2. Remote execution via `nodedispatch`

To remotely execute defined commands use the `nodedispatch` flag in the call to the [controller](#) task.

The following example is similar to the `ctl-exec` one used above but this time instead of calling the Linux `whoami` directly, it uses `coreutil`'s command in "shellutil":

```
<command name="dispatch-whoami" description="dispatch shellutil
whomai command"
  command-type="AntCommand" is-static="true">
  <implementation>

    <controller>
      <execute nodedispatch="true">
        <nodeset>
          <include names=".*"/>
          <exclude names="localhost"/>
        </nodeset>
        <context depot="${context.depot}"/>
        <command name="whoami" module="shellutil"/>
      </execute>
    </controller>

  </implementation>
</command>
```

This pattern can be used to invoke any defined command. Here's a revised implementation that makes it very open ended.

```
<command name="dispatch-command" description="dispatch the defined
command"
  command-type="AntCommand" is-static="true">
  <implementation>

    <controller>
      <execute nodedispatch="true">
        <nodeset>
          <include names="${opts.nodes}"/>
          <exclude names="${opts.xnodes}"/>
        </nodeset>
        <context depot="${context.depot}"/>
        <command name="${opts.command}" module="${opts.module}"/>
      </execute>
    </controller>

  </implementation>
  <opts>
    <opt parameter="command" property="opts.command" type="string"
required="true"/>
    <opt parameter="module" property="opts.module" type="string"
required="true"/>
    <opt parameter="nodes" property="opts.nodes" type="string"
```

```
required="true" default=".*"/>  
  <opt parameter="xnodes" property="opts.xnodes" type="string"  
required="true" default="localhost"/>  
  </opts>  
</command>
```

With this new implementation you can run the previous whoami command example with the following invocation:

```
ctl -m mymodule -c dispatch-command -- -command whoami -module shellutil
```