

# Project Setup

## Table of contents

1 Server setup.....	2
1.1 Repository.....	2
1.2 Create the project depot.....	3
2 Client setup.....	3
3 CTL nodes.....	3
4 CTL deployments.....	4

CTL allows you to organize modules and objects into projects. Projects allow you to partition management into separate areas. You might choose to create a project for managing procedures for each of your environments or might choose to designate a project for managing a related set of services. Each project gets its own *depot*, a local repository containing its modules and object data. The `ctl-setup` command you ran at installation time created a default project called "default".

The `ctl-depot` command provides a set of administrative actions for creating, updating and removing project depots. Typing `ctl-depot -h` will show the command usage but this document discusses it further.

CTL is designed to work in a distributed environment wherein a user may designate a central host to control actions across target hosts. Additionally, there may be files that need to be distributed during various operations. Finally, you may wish to distribute modules from a central repository. The following two sections describe how to setup a new project for both a centralized "server" host and targeted "client" hosts.

#### Note:

CTL really has no distinction between server and client software installations but rather sees these as conventions.

## 1. Server setup

This section describes how to setup a host to act as a central point of control, as well as, describe how to setup a repository to support file distribution.

### 1.1. Repository

The repository need be nothing more than a simple HTTP server handling GET calls but could also be a WebDAV server if it is desirable to publish files to the repository via PUT.

To setup a server repository, make the following changes to your `$CTL_BASE/etc/framework.properties` file

```
# path to your web server's content root:
framework.webdav.rootdir=/path/to/your/web/rootdir
# Well known name of your web server. This must equal
${framework.node.hostname}
framework.server.hostname = ${framework.node.hostname}
```

Ensure you can access that content via your chosen webserver. Now when you run `ctl-depot` to create new project depots, `ctl-depot` will bootstrap it with a directory structure.

The `ctl-depot` command will then create a structure like this:

```
web-rootdir
+--- project          // project base directory
    +--- publish
        +--- modules  // contains packaged modules
    +--- etc          // contains project configuration
        |
        --- deployments.properties
```

## 1.2. Create the project depot

On the server host run create the depot for your project name:

```
ctl-depot -p project -a create
```

The project depot setup procedure will create a directory workspace in the `CTL_BASE/depots`, initialize it with any standard modules and create a directory hierarchy in the repository.

## 2. Client setup

Client setup makes the following assumptions:

1. The client machine has CTL software installed and you ran `ctl-setup`.
2. You can SSH from the server to the client without any prompting.

The procedure to create the depot for a new project on a client is pretty much the same as a server (minus the repository).

On each client machine, run (substitute "project" with your project's name):

```
ctl-depot -p project -a create
```

## 3. CTL nodes

CTL's dispatch relies on information about the hosts to which it executes commands. This is defined in the [nodes.properties](#) file. It's best to manage this file centrally so keep the authoritative copy on the repository.

Edit the `nodes.properties` file on the repository: `webroot/project/etc/nodes.properties` and stipulate what nodes you want your modules (or objects) to be deployed. The example file listing below gives you a sample entry:

```
### file format:
#
# node.name.attribute = value
#
```

```
# example: define node1
#
# node1
node.node1.description=The node1 host
node.node1.type=Node
node.node1.name=node1
node.node1.hostname=node1
node.node1.os-arch=i386
node.node1.os-family=unix
node.node1.os-name=Linux
node.node1.os-version=9.2.0
node.node1.tags=sandbox
```

## 4. CTL deployments

The next part to setting things up is deciding where things should go, or what CTL calls "deployments". This is defined in the [deployments.properties](#) file. It's best to manage this file centrally so keep the authoritative copy on the repository.

Edit the `deployments.properties` file on the repository:  
`webroot/project/etc/deployments.properties` and stipulate what nodes you want your modules (or objects) to be deployed. The example file listing below gives you a sample entry:

```
### module deployments format:
#
# module-deployment.project.module = node1,node2,...,node3
#
# example: stipulate that for "yourproject" the
#           "shellutil" module should be located on "node1,2,3"
#
module-deployment.yourproject.shellutil = node1,node2,node3
```

After you make any change to the `deployments.properties` file, you can run `ctl-depot` to roll out the change. Use the `install` argument to the `-a` option.

```
ctl-depot -p project -a install
```

The "install" action first downloads the `deployments.properties` file from the repository and then for each module or object deployment defined, installs it.

### Note:

You might be interested to know that the `install` action is really a wrapper around two commands found in the CTL base module, [Managed-Entity](#). For module deployments, a call to [Install-Module](#) is run. While for the object deployments, a call to [Install](#) is invoked. You can always run either command directly, if you wish.

## Synchronizing

If you registered your client machines to use shellutil module in

## *Project Setup*

deployments.properties, you can use ctl's network execution capability to call `ctl-depot` on the client machines.

For example, if `coreutils` has been installed then on your hosts run the following command anytime you want to synchronize your deployments:

```
ctl -I ".*" -m shellutil -c exec -- -executable ctl-depot -argline "-D"
```