**Configuration Management tool: Provisioner**

**Preface:**

This document details the architecture of the configuration management tool and how to install and use it. It also explains the key design limitations.

**Architecture:**

This tool will use a client-server model, where one node in the network will serve as a master and contain the core programming modules and all dependencies. All instructions will be executed on the remote nodes using ssh protocol. The following assumptions must be true for the tool to work:

A user with elevated privileges of root in the environment will setup initial ssh connections between the master and all client nodes using the root or another super-user account with sudo permissions for installing and removing packages and overwriting configurations.

The tool will take the following input:

1. Json file: This tool accepts a single Json file as input for performing all of its supported capabilities. The Json format will be provided further below in this document. The supported capabilities are categorized under 4 Json objects: “install”, “remove”, “configure” & “service” (case-sensitive). The file must contain one more object “servers” which specify the remote servers where the instructions will be executed.
2. Templates: This directory will contain any file or folders that the user wants to push to the remote nodes. The file/folder permissions, ownership and information in this template directory will override the ones on the destination servers.

**JSON Format:**

The tool will perform operations in the following rigid order, regardless how the input file is structured:

1. remove - Uninstall any packages specified under this object. For removing debian packages the input must adhere to this format:

{

“remove” {

“Package\_Name”: “Version”, (must mention the exact package version)

“Package\_Name”: “auto” (to let the system automatically choose the package version)

}

}

1. install - Install Any packages specified under this object. Installation instructions must stick to this format:

{

“install” {

“Package\_Name”: “Version”, (must mention the exact package version)

“Package\_Name”: “auto” (to let the system automatically choose the package version)

}

}

1. configure - Copy any files or folders mentioned under this, from the templates directory to the mentioned destination. The format for transmitting files or folders must adhere to:

{

“configure” {

“File/Folder Name”: “Destination\_Location” (Absolute path on the destination server)

}

}

Note: These files/folders must be available under the templates directory in application folder.

1. service – Perform the actions mentioned under this object for the services mentioned. Service name and corresponding action should be specified like:

{

“remove” {

“Service\_Name”: “Action” (Action to be performed on the service)

}

}

The list of servers where these actions will be executed have to be mentioned under the “servers” object in the same input file.

{

“servers” {

“Server\_Name/IP”: “Super-User-ID” (SSH access must exist between the master and the server mentioned here using this ID)

}

}

This is a sample Json file that will install php5 and apache2 on a Ubuntu server and then copy a modified apache2.conf and push some webpages to the doc root : /var/www/ and restart the apache2 service at the end of these operations:

{

"servers": {

"34.204.11.16":"root"

},

"service": {

"apache2": "restart"

},

"install": {

"php5": "auto",

"apache2": "auto"

},

"configure": {

"apache2.conf":"/etc/apache2/",

"html":"/var/www/"

}

}

The tool will not function if the servers list and at least one supported action is not provided in the input Json file.

**Install Steps:**

Extract the tar file Provisioner.tar.gz under any directory and execute the **install.sh** script using either root or a super user account with sudo permissions. This will download and install all dependencies and setup the log and templates directory for the application. The account that is used to execute the installation and run the tool must have ssh access to all the destination servers to which configurations will be pushed.

**How to Run the Application:**

The user must execute Provision.pl perl script which resides under bin in the application directory on master. The input file must be provided as an argument for this execution:

./Provision.pl /Absolute\_Path/input.json

Application logs can be read from the log folder in the install location.

Templates for pushing files and folders to remote servers must be placed in the templates directory in the install folder

**Limitations:**

Due to the way Perl handles Json files, there cannot be more than one entry for each Object in the Json file. And there cannot be more than one entry for each action under these objects in the Json file. The input Json file is parsed and stored in an Associative array, hence duplicates overwrite previous inputs and the most recent input is the only one that is considered.

A super user account must be used to execute all of the actions supported by this tool, for installation and removal the tool uses apt-get which is part of all Ubuntu systems. For syncing files/folders the tool uses Rsync, therefore permissions must exist on the destination for this tool to overwrite files and folders.

After removal of certain packages, lingering dependencies may exist which can be removed using apt-get purge –auto-remove, therefore manual user intervention may be required to execute this in certain situations. This is a conscious choice due to the risks involved in blindly purging libraries which still be used by other packages.