RPM Packaging for Ansible Automation Configuration Management in Linux

Minakshi Pushpendra Chavan[1], Pushpendra Madhukar Chavan[2], Dr Babasaheb S. Sonawane[3]

[1]M.Tech. CST Student – G. S. Mandal's Maharashtra Institute of Technology, Aurangabad, MH – IN
[2]M.Tech. CSE Student – CSMSS's Chh Shahu College of Engineering, Aurangabad, MH – IN
[3]Asst. Professor - CSE Dept – G.S. Mandal's Maharashtra Institute of Technology, Aurangabad, MH – IN

---***---

Abstract – Ansible automation is not a new configuration management method but is a widely used and accepted DevOps tool to manage Linux as well as non-Linux servers across the networks. It not only helps to manage systems from the control host but also enables remote administration tasks such as package management, system evaluation, security scans, user management as well as service configuration implementations. This paper utilises the same concept for systems management and enables the configuration files as well as ansible yaml files packages and shipped through fully secured and signed rpm package which guarantees that neither the rpm package is tampered nor the shipped ansible yaml playbooks are modified.

The rpm package can be built on Red Hat Enterprise Linux and can also be signed to make sure that the package doesn't get any replaced files in between the time it is shipped by the vendor till it reaches the end user/customer. This paper will also talk about the rpm building best practices including spec files, public-private key pairs, rpm macros as well as yum metadata to host the package over the internet. The deployed package will make sure that it adheres to the complete lifecycle of the rpm package management, appropriate dependencies and suitable changelog as per the Open Source and Red Hat Package Management standards.

***Key Words*:** rpm, ansible, private-public keypair, rpm-macros-spec file, security.

1. INTRODUCTION

A very well known IT Change Management practice in ITIL calls out for the need of a uniform process to follow while making changes to the system across the whole network. Any tool which ensures the points below can be accepted as a configuration management tool in any Industries.

- Faster Changes to be pushed to the end systems.
- Easy understanding of the complex processes.
- No human errors since an automation process.
- Very minimal or no downtime at all while making the change.
- Appropriate logging enabled for auditing purposes.
- Easy reporting of the systems where changes are made.
- A rollback and Disaster recovery plan.

Simple process to add or remove systems from a single point etc. Ansible automation justifies all the requirements here, the only problem here is that the ansible playbooks are very easy to be tampered with, and once tampered, the playbooks can lead to disaster if someone gets read-write access to it and those aren't pre-validated prior executed.

With the help of rpm packaging mechanism, it has become very easy to deliver data and package payload to a number of systems. RPM package not only validates the originality of the deployed files, but also helps to replace them in normal life cycle updates of the said package.

Combination of ansible custom playbooks and delivering those through rpm channels and features makes it a very approachable method. The next chapters will discuss each of these topics in detail.

2. Related Work

Automation, Configuration and Change Management is the need of today's Information Technology Infrastructure both in on-prem and Cloud setup. It becomes a challenge when these processes are followed by individuals manually, hence there is a need of tools like Ansible, Terraform & Puppet models which run these massive infrastructure changes diligently. These DevOps Tools not only make appropriate and identical changes to the system or group of systems but also keeps track of all changes done with messages/logs and even errors so an effective and related post-performance actions can be performed.

The tools like ansible, terraform and puppet are just tools, and don't just get delivered with the actions to be taken on the system, these tools need a set of instructions in the form of inputs. The related work discusses not only authoring such an extensive input file but also packaging those into a very highly secured rpm packaging format. Let's discuss these topics in brief as follows.

2.1 RPM: RPM is nothing but a packaging format backed and supported by Red Hat. A rpm packaged software guarantees easy shipping of the payload through multiple means like Internet/Network/Locally hosted Private/Public repositories. World Wide Public repositories like VLC, RPMForge, EPEL, Fedora, etc are few of many trusted ones. A rpm package is basically a compressed archive packaged in such a way that it includes package metadata, binary sources, configuration files, manual pages, scripting and sometimes ghost files.

The rpm package is built with the help of a specification file which is also known as spec file. There are many optional and mandatory sections. Some of the mandatory sections are as follows.

- **Package Metadata including**
  1. Name of the package, packager and source.
  2. version, release, package group, architecture and build architecture information of the package.
  3. Dependencies required during package build as well as during installation of the package.
  4. Short summary of the packaged software.

5. Long description of the packaged software.
6. Licensing information.

- **Preparation, Setup & Build section**: This section prepares the pre-packaging environment, compiles the sources, fetches required files from the build host and provides it to the next rpm build process.
- **Install section**: The install section would fetch the compiled binary sources, manual pages, configuration files, directories to an appropriate buildroot directories for further packaging.
- **cleanup section**: This section cleans up all residue post packaging.
- **Files Section**: Very important section of a specification file which depicts the type of the files, permissions, ownership and group ownership of the deployed files by that specific rpm package. The files section is very helpful as the entries from this section would be populated in the rpm database located at /var/lib/rpm in the system and this database would be responding back to all rpm queries made by the system administrator.
- **Changelog**: As per ITIL Process, one needs to make sure that all the changes must be logged in order to keep track of them all and mitigate, improvise and avoid duplication of efforts. This is again a must-have section of a rpm package.
- **Scripts**: These sections are optional, but are very helpful as well. There are four types of script sections available to use in the specification file as follows.
  1. **pre** – the set of instructions to be executed before installation of the package.
  2. **post** – the set of instructions to be executed after installation of the package.
  3. **preun** – the set of instructions to be executed before uninstallation of the package.
  4. **postun** – the set of instructions to be executed after uninstallation of the package.

Upon package build, both the source package and binary package can be hosted on a public/private repository.

2.2 Ansible: Though Ansible is an open source software project, was part of Ansible Inc which got acquired by Red Hat in 2015 and since then there has been numerous positive changes to its core as well as functionalities. Ansible is a configuration management tool which works on trusted networks where ssh keys are used for remote logins and built in python binaries are utilized for client side configuration changes. Hence, unlike puppet, Ansible doesn't need to have a client deployed with a dedicated client side software. The ansible architecture consists of control host and managed nodes. As per its design, the control host would always need to be on a Linux system, whereas the managed hosts can be of any type of an Operating System including Linux, Windows and MacOS. An appropriate ansible architecture can be briefed with the following subsections.

Inventory: Inventory file is nothing but a text file written in such a way that all the systems are grouped properly in a specific format. Inventory file not only includes the system's information, but also local/global variables. An Inventory file looks like below.

```
[uat]
www.example-uat.com
www.soon-to-be-pre-prod-uat.com
[prod]
www.production-sales.com
www.production-it.com
```

Modules: The ansible modules are in fact the scripts which are pushed to the managed nodes when the playbooks operations are in place. These modules are temporarily copied to the managed nodes, and upon execution of the plays, are removed from them. There is a flexibility that one can even write custom modules as well.

Playbooks: The playbooks are actually the program or an orchestration instructions to be executed on the targeted managed nodes which can be written in YAML format. These will be the building blocks of the rpm package to be shipped as a part of payload delivery.

2.3 GPG Encryption: The GnuPG helps encrypt and sign the user data as well as communications sent over the internet. During key pair generation, GPG generates both public and private keys. Any sort of data can be encrypted with the public key and only gets decrypted with the private key which resides only at the actual owner so the data can be sent over to the only intended person. The data like rpm package can also be only signed with a private key and once altered, it needs to be signed once again and that's possible only by the private key holder, which ultimately guarantees that only the authoritative and rightful owner has permissions and rights to change the file, not someone else masquerading in between. This is only possible if the private keys are kept safely by the owner. The utmost physical safety of the private key is the sole responsibility of the rightful owner himself/herself.

2.4 Repository: A package repository is a combination of binary and source packages along with their group information and package metadata held tightly together. The repository can be hosted anywhere i.e. on the internet, or network, or even a Hard disk drive. It can be of any nature such as public or private. There are multiple software available such as Red Hat Satellite, Jfrog Artifactory, npm, spacewalk project and many more. Red Hat also provides in-built software repository tools like createrepo and yum to fetch the details from remote repositories.

There are many public software repositories, some of them are VLC, rpmforge, epel, fedora, google-chrome and many more. Best practices recommend hosting only signed rpm packages and these signed packages to be installed only if the public key of those specific package signers is first imported and trusted by the end system administrator installing these packages from the repository. An example repository configuration file at an end client looks like this.

```
  [repository_name]
  Name=Example Repository
  baseurl=http://somelocation/repository
  enabled=1
  gpgcheck=1
  gpgkey=http://somelocation/repo-public-
gpg-key.txt
```

3. IMPLEMENTATION

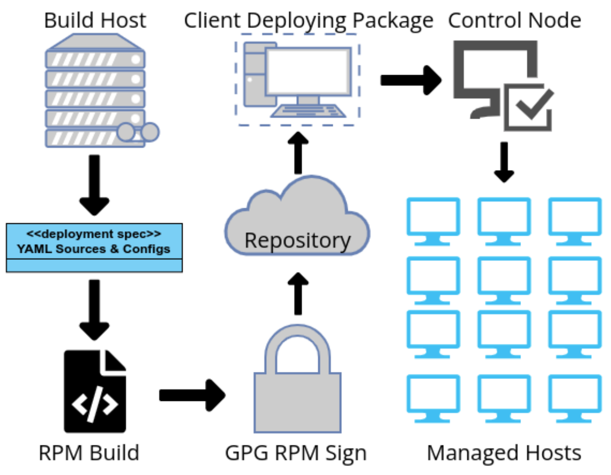
There would be two sections for the implementation model, first section would be to write an appropriate playbook and then these files along with ansible configurations and inventory would be the source code for the further stage of packaging it together.

The proof of concept model includes several steps for the implementation purpose. This research is an ongoing work hence, yaml definitions are supposed to be written whereas the packaging mechanism would include the steps below.

1. **Learn and understand the build host machine**: The system where a rpm package is supposed to be built needs a verification of supportability. The system can be of any Linux flavours such as Fedora, Red Hat Enterprise Linux, or CentOS Linux.
2. **Collect and gather the source code and yaml files**: In original implementation, the code would be the ansible configuration management playbooks written in such a way that those would perform individual and dedicated tasks over the trusted network. The code would also include an inventory file and a script which would self setup the control node.
3. **Create the SPECification file**: The code would need to be packaged and the instructions would be depicted by the spec file.
4. **Build the binary as well as source RPM package**: The verified host system from step 01 would be where the package can be built.
5. **Generate GPG key pair to sign the built rpm package**: The private key of the gpg keypair would be used to sign the package to be shipped through the repository.
6. **Verify the imported public keys and import a new public key from public-private keypair**: One time execution on a test system to verify if the keypair works as expected and is ready to use to sign the generated rpm package.
7. **Sign the newly generated binary rpm package**: Needs to be done only on the build host, the private gpg key would be used to sign the rpm package.
8. **Install the newly generated rpm package**: First test deployment of the built rpm to verify the contents and usability.
9. **Create a test repository online and publish it on an internet accessible location for test purpose**: Creation of the public repository for free use.
10. To test it further, **create a repository configuration file** on any of the test Red Hat Enterprise Linux or Fedora Linux pointing out to the newly hosted online repository: The end client systems would utilize the client repository file to fetch packages from publicly hosted repositories.
11. Finally **install the newly hosted binary package on any client** through the internet hosted repository: Finally fetch the package built and hosted on the internet.

4. RESULTS

A single package can help manage 1000s of managed nodes for the said configuration management. The rpm package can be built in such a way that it will also fetch the ansible binaries as a part of its dependencies, not only it is secure but also easy for version control, a new version can be easily pushed to the channel.



Process Flow Chart

Any changes to the existing playbooks would be easily reported to the system administration since these files would be considered libraries and rpm rules of verification would definitely catch the changes made to any of the files.

The performance model for any specific action required to complete task A manually versus using ansible playbook in minutes can be predicted approximately as follows.

Number of Managed Nodes	Time Required to perform action A manually (in minutes)	Time Required to perform action A using process defined (in minutes)
1	5	2
2	10	3
5	25	5
10	50	7
20	100	10
100	500	20

* the time is in minutes and isn't accurate but approximate

5. CONCLUSIONS

Use of ansible playbooks for configuration change management helps extremely faster and reliable methods as compared to manual methods. Use of rpm adds up additional security and guarantee to the predefined and preprogrammed YAML files of not to be tampered with. The minimal pros and cons can be briefed as follows.

Pros:

1. Secured, Guaranteed.
2. Ease of version control.
3. Easy access via Public repositories.
4. Faster than manual method

5. Extensive logging mechanism

Cons:

1. Works only on trusted network
2. Single point of failure if control node goes down.
3. Can't be used for infrastructure deployment unlike Terraform.

ACKNOWLEDGEMENT

Gratitude to the original creators of RPM, Ansible and Red Hat Enterprise Linux Projects. A number of contributors all across the world have freely improvised these tools so today the world can rely on them.

Special Thanks to Red Hat Organization for making all these tools open source and free to use without seeking copyright or licensing control on use of ansible, rpm package management, and repository tools for educational as well as production purpose.

REFERENCES

1. Documentation – Adam Miller, Maxim Svistunov, Marie Doleželová, et al, RPM Packaging Guide, https://rpm-packaging-guide.github.io/

2. Documentation – Fedora Documentation, https://rpm-packaging-guide.github.io/

3. Siphon Mongkolluksame; Chavee Issariyapat; Panita Pongpaibool; Koonlachat Meesublak; Nontaluck Nulong; Sirikarn Pukkawanna (2012), "A management system for software package distribution" - https://ieeexplore.ieee.org/abstract/document/6304372, Date of Conference: 29 July 2012 – 02 August 2012, Date Added to IEEE Xplore: 17 September 2012, Print ISSN: 2159-5100, INSPEC Accession Number: 12999654, Publisher: IEEE, Conference Location: Vancouver, BC, Canada

4. Jing Wang; Qingbo Wu; Yusong Tan; Jing Xu; Xiaoli Sun (2015), "A graph method of package dependency analysis on Linux Operating system" - https://ieeexplore.ieee.org/abstract/document/7490780, 2015 4th International Conference on Computer Science and Network Technology (ICCSNT), Date of Conference: 19-20 December 2015, Date Added to IEEE Xplore: 16 June 2016, Electronic ISBN:978-1-4673-8173-4, CD:978-1-4673-8172-7, INSPEC Accession Number: 16090264, DOI: 10.1109/ICCSNT.2015.7490780, Publisher: IEEE, Conference Location: Harbin

5. Shrinidhi G Hegde; G Ranjani (2021), "Package Management System in Linux" - https://ieeexplore.ieee.org/abstract/document/954480541, 2021 Asian Conference on Innovation in Technology (ASIANCON), Date of Conference: 27-29 August 2021, Date Added to IEEE Xplore: 04 October 2021, Electronic ISBN:978-1-7281-8402-9, CD:978-1-7281-8400-5, USB ISBN:978-1-7281-8401-2, Print on Demand(PoD) ISBN:978-1-7281-8403-6, INSPEC Accession Number: 21202856, DOI: 10.1109/ASIANCON51346.2021.9544805, Publisher: IEEE

BIOGRAPHIES

Minakshi Chavan is currently pursing Masters of Technology in G S Mandal's Maharashtra Institute of Technology, Aurangabad. She holds a Bachelor of Engineering Degree in Computer Science and Engineering. Her areas of interests are Linux and DevOps. Minakshi is a Red Hat Certified Engineer.

Pushpendra Chavan is a student of M.Tech. In CSE department of CSMSS's Chh Shahu College of Engineering. His areas of interest are Linux Security and Cloud Computing. He is a Red Hat Certified Architect.

Dr B. S. Sonawane is an Assistant Professor at G. S. Mandal's Maharashtra Institute of Technology, Aurangabad and has hold positions like Vice Principal at MIT Rotegaon. Prof. Sonawane is a PhD holder in the area of Diskless Clients and has expertise in Linux Networking and AWS.