Ansible for AIX: Developing for an Open-source Collection to Modernize an Operating System's Offerings

A Technical Report submitted to the Department of Computer Science

Presented to the Faculty of the School of Engineering and Applied Science
University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree
Bachelor of Science, School of Engineering

Richard William Taylor
Spring, 2022

On my honor as a University Student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments

Rosanne Vrugtman, Department of Computer Science
Briana Morrison, Department of Computer Science

Ansible for AIX: Developing for an Open-source Collection to Modernize an Operating System's Offerings

CS4991 Capstone Report, 2023

Richard Taylor
Computer Science
The University of Virginia
School of Engineering and Applied Science
Charlottesville, Virginia USA
rwt7uxg@virginia.edu

ABSTRACT
IBM found that customers utilizing their operating system AIX desired more modern tools for automation of routine tasks. A team of developers was tasked with creating and maintaining an Open-Source collection of Ansible modules to support automation of AIX processes, including my task of creating modules for managing encryption settings of storage device partitions. To develop these encryption modules, I gathered information about the features of the AIX commands that needed to be automated and followed a template of the structure of other modules in the collection to construct–two modules that allow users to automate various encryption processes and settings. In addition, I created an extensive test suite to ensure the modules' functionalities prior to every new deployment of the collection. These modules afforded AIX users increased efficiency in handling encryption tasks and modernized the way AIX could be controlled. Future work involves changes to the way passwords are accepted by AIX to ensure that these functionalities are secure.

1. INTRODUCTION

Data breaches pose a significant threat to enterprises, as the loss of sensitive information can result in financial and reputational losses. Globally, it is estimated that the annual cost of data breaches in 2019 was $2.1 trillion, with 43% of these leaks coming from internal employees (Cheng, et al., 2017). The process of encrypting data across all of a business or organization's servers can be both complex and time-consuming, leading to an increased likelihood of errors and unaddressed gaps. A focus on simplifying the process of protecting data against these kinds of leaks is often taken as a preventative measure.

AIX's built in hdcryptmgr commands simplify the process of managing encryption settings, but must be run manually on each drive partition on every machine that is to be encrypted. System administrators at organizations using AIX must devote significant time whenever any change to the system's encryption settings must be made, and can easily miss or overlook parts of the system. Modernizing AIX by adding automation capabilities means administrators no longer have to manually run the commands for every part of the system, saving time and resources, and increasing the value of AIX to the organizations who use it.

2. RELATED WORKS

To create the hdcrypt Ansible modules, I referenced IBM's AIX documentation on the hdcyrptmgr command suite to learn about their functions and how to use them (IBM, 2023). IBM and the existing Ansible for AIX team provides thorough documentation for the most recent version of the Ansible collection (IBM, 2020). The modules I created follow the same

format as those set up by the existing modules in the collection, and I used the documentation to learn about how the existing modules and the Ansible tool itself worked.

3. PROCESS DESIGN

In order to develop for the Ansible AIX collection, I first needed to understand what requirements were necessary for the 2 hdcrypt modules, and requirements that all ansible modules have.

3.1 Ansible Collection Structure

The Ansible AIX collection is made up mainly of modules, playbooks, and roles. Modules encompass different actions or commands that users invoke to perform some function on AIX machines. Playbooks are special scripts that specify target end machines to control and what tasks to run on them. Each task is run on sequentially on each target machine, and specifies which module and the necessary options for that module to execute. Figure 1 shows a sample Ansible Playbook.

```
- name: Update db servers
  hosts: databases
  remote_user: root

  tasks:
  - name: Ensure postgresql is at the latest versio
    ansible.builtin.yum:
      name: postgresql
      state: latest
  - name: Ensure that postgresql is started
    ansible.builtin.service:
      name: postgresql
      state: started
```

Figure 1: Example Playbook

Roles are essentially a mechanism for executing complex tasks that would require multiple playbooks and modules using the same variables to be run, simplifying creation and reuse of playbooks. My contribution to the collection consisted of two modules, hdcrypt_encrypt and hdcrypt_auth, as well as sample playbooks that showed the capabilities of each module and how they were used, and a suite of test cases that covered every use case of the modules.

3.2 Ansible Module Requirements

Ansible modules are Python files which import the AnsibleModule class from the base ansible package, and use an instance of this class to perform most of the functionalities in the module. This object makes secure SSH connections to the target machines and executes commands on them. Because of this, all information required to complete any task must be provided in the playbook. Users are therefore able to execute often complex sequences of commands repeatedly with the execution of a playbook. Ansible modules by design must be idempotent, meaning re-running a playbook with the same inputs does not change anything after the first run. Modules are also Atomic, meaning that if any part of a task fails, no change will be done for that task at all. These are properties that I had to include in the development of the hdcrypt modules.

3.3 hdcrypt Requirements

The Ansible for AIX team determined that the hdcrypt modules should cover the actions of encrypting and decrypting logical volumes, which are virtual partitions of storage devices on AIX, and volume groups, as well as adding and removing authentication methods, and unlocking logical volumes and volume groups for decryption with specific authentication methods. Users were to have the option of performing these actions on a single or multiple volumes or volume groups, or a mix of any number of volumes and volume groups. These actions were split into two modules, hdcrypt_encrypt to handle encrypting and decrypting, and hdcrypt_auth to handle managing authentication methods. The modules also needed to be able to support use of any authentication method available in AIX.

3.4 Development Process

To develop these modules, I broke them down into two “epics,” or major features, separating the encryption module from the authentication module. I then further broke down the epics

into more deliverable issues that could be completed each development sprint cycle. Issues included smaller features, such adding the encryption action for a single volume or volume group, which would eventually build up to the complete module. Along with each individual issue, I developed test cases that covered each functionality added in that issue, and ran all the tests created with each issue to ensure that the new features were functional and compatible with the already added ones.

3.5 Issues and Solutions

The biggest challenge that occurred during the development of the modules related to the way the hcryptmgr commands work on AIX. All logical volumes that have encryption enabled in AIX are required to have a password authentication method at all times, and must be in an unlocked state to perform any encryption or authentication methods on them. Currently, in order to add a password authentication method and unlock, encrypt or decrypt a volume with a password, a user must provide the password in response to command prompts when the commands are run. This violated the requirement that the modules be able to run without any user input beyond what is provided in the playbook.

Initially, I sought to understand the way passwords were accepted by the hdcryptmgr command and stored in AIX, and attempted to see if it was possible to change the command to accept passwords in another way. However, due to the way encrypted passwords are stored in AIX, this was not immediately possible. Instead, I opted to handle the user passwords prompts by running the command in Linux expect scripts. These scripts were formed by the module on the ansible client using the options provided in the playbook and then sent to be ran on the target end machine. Expect scripts work by running a provided command, and "expecting" some output from that command, which when received sends the next part of the script. This solution circumvented the password prompts in the hdcryptmgr command, but introduced new security flaws. Because the expect script is sent from the host to each target over encrypted SSH connections, it was not an issue that the password was available in clear text within it; however, if a user is looking at all running processes on the AIX end machine when the expect command is run, the password would appear in cleartext, which is not secure. The ansible team accepted this solution temporarily, as most functionality of the modules did not rely on it or have any other user prompts.

4. RESULTS

I was able to successfully complete two hdcrypt modules, affording users greater efficiency in completing various encryption-related tasks, which was one of the features current users most requested be added to the AIX collection. The modules changed the process of encrypting logical volumes from a multi-step, time-consuming process that could only be done manually on each machine to a fully automatic one that could be completed across many AIX machines with a single command and little user input. The most recent version of the collection, which includes some of my work on this project, has garnered over 160,000 downloads, and is one of the most widely used IBM Ansible collections.

5. CONCLUSION

During my experience as an intern, I was able to solve part of the important problem of modernizing one of IBM's oldest operating system offerings. Through the creation of the hdcrypt modules, the process of encryption management on AIX can be greatly simplified for users, allowing for greater protection against data breaches and data loss, which pose significant threats to enterprises.

6. FUTURE WORK

In order for the hdcrypt modules to be fully introduced into the public facing collection, the vulnerability regarding using expect scripts to send passwords to hdcryptmgr commands

must be remedied. One potential solution proposed to and being examined by the Ansible for AIX team is to rework the hdcryptmgr commands to not require user input at all, but rather be able to accept an already encrypted password as a command line parameter. However, such a large change would require input and acceptance from other development teams in charge of the hdcryptmgr commands, which would require a lot of time. Alternatively, the ansible team is also researching how other collections implement these modules, as the hdcryptmgr commands are closely related to commands on other UNIX based operating systems. In addition to this, future work for the collection and hdcrypt suite of modules includes creating modules to deal with managing Platform Key Store (PKS) and KeyServer settings and authentication options.

REFERENCES

[1] Cheng, L., Liu, F., and Yao, D. 2017. Enterprise data breach: Causes, challenges, prevention, and Future Directions. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 7, 5 (2017). DOI: http://dx.doi.org/10.1002/widm.1211

[2] IBM. 2023. hdcryptmgr Command. (January 2023). Retrieved from https://www.ibm.com/docs/en/aix/7.2?topic=h-hdcryptmgr-command

[3] IBM. 2020. IBM Power Systems Aix Collection for ansible. Retrieved from https://ibm.github.io/ansible-power-aix/index.html