### Encrypting Sensitive Data in HDP Cluster Configuration Files

In our HDP (Hortonworks Data Platform) cluster configuration files, some passwords and other sensitive information are currently stored in clear text. This poses a security risk and must be addressed by implementing encryption mechanisms. To secure this data, we will use our DevOps team's encryption script to store and retrieve sensitive information securely.

#### Steps to Implement Encryption for Sensitive Data:

1. **Identify Sensitive Data:**
   1. Review configuration files to identify clear-text passwords and other sensitive information.
   2. Maintain a list of parameters that require encryption.

As per the review in our default.yaml configuration files we have clear text password .

We already implemented in some of the module and we can use the same format.

hdp\_config\_mgmt

map.jinga

What are the files are using cleartext password and need to be updated

Cear text Password Stored here

ambari\_inf2/default.yaml

The following files are calling default.yaml detail

Infoplatform/ambari\_template/blueprint/files/blueprint\_dynamic.jinga

1. **Use DevOps Encryption Script:**
   1. Obtain the latest version of the encryption script from the DevOps team.
   2. Ensure the script is available and executable on all relevant nodes.
2. **Encrypt Passwords and Store Securely:**
   1. Use the encryption script to encrypt each sensitive parameter.
   2. Store the encrypted values in a secure vault or configuration file.
3. **Modify Configuration Files:**
   1. Replace clear-text passwords with their encrypted counterparts.
   2. Ensure the configuration files reference the encrypted values correctly.
4. **Retrieve and Decrypt at Runtime:**
   1. Modify services to use the encryption script to decrypt passwords dynamically when required.
   2. Ensure only authorized processes and users have access to decryption keys.
5. **Validate and Test:**
   1. Restart relevant services and ensure they function correctly with encrypted credentials.
   2. Monitor logs for errors related to decryption.
6. **Security Best Practices:**
   1. Limit access to the encryption keys and scripts.
   2. Rotate encryption keys periodically.
   3. Maintain logs of access to encrypted data.
   4. Ensure compliance with security policies and best practices.

### Pros and Cons

#### Pros:

1. Enhances security by preventing unauthorized access to sensitive data.
2. Helps in compliance with security regulations and best practices.
3. Reduces risk of credential exposure in configuration files.
4. Supports dynamic decryption at runtime, minimizing manual intervention.

#### Cons:

1. No High Availability (HA) available for the encryption tool.
2. The encryption tool does not support replication, posing a risk if the primary instance fails.
3. Potential performance overhead due to encryption and decryption processes.
4. Requires coordination with the DevOps team for implementation and maintenance.

By implementing these steps, we enhance the security of our HDP cluster configuration files and prevent unauthorized access to sensitive data. Please coordinate with the DevOps team to ensure proper integration and testing of this solution.