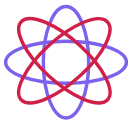


TechGirlKB

How to Use AWS SSM Parameter Store with Ansible

If you're like many frustrated Dev/Ops teams out there, you may be tired of using Ansible's vault function to encrypt secrets, and looking for other options. AWS offers a couple options for storing secrets:

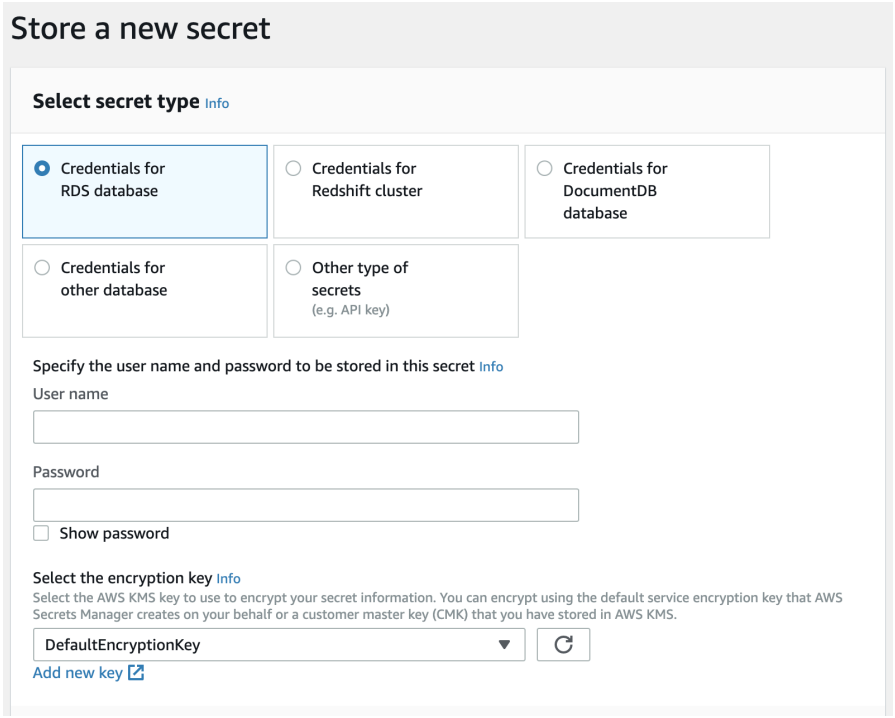
- The more obviously-named Secrets Manager tool
- The less-obvious Systems Manager Parameter Store

As it turns out, Ansible has lookup plugins for both AWS tools. In this guide we'll explore both, and why (spoiler alert!) our team decided to go with Systems Manager Parameter Store in the end.

different secure variable storage systems.

Secrets Manager

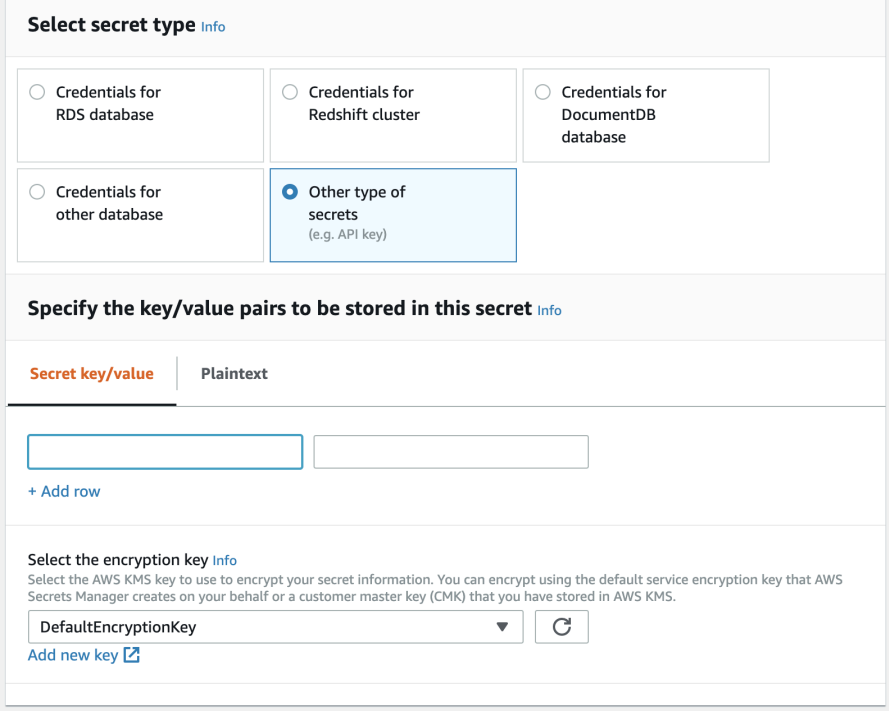
Firstly, Secrets Manager has different secret types, mostly geared toward storing, encrypting, (and regularly rotating) database credentials. Here's a quick look at the options.



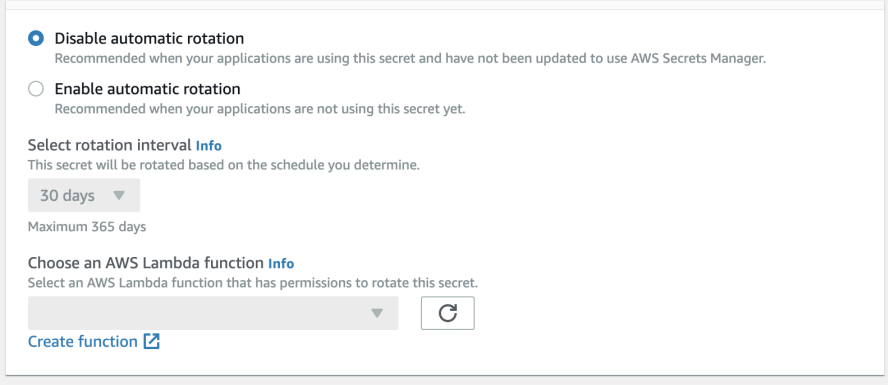
The options shown in the "Store a new secret" pane are:

- Credentials for RDS database
- Credentials for Redshift cluster
- Credentials for DocumentDB database

database credentials. There is, however, an option to store another secret in “Key” and “Value” format. This is the option our team was planning to use for most secure variables.



In this pane, you can add a simple key-value pair. On the next screen you can add an identifying name and any tags you wish to the key, followed by a pane where you can select automatic rotation for the key if you choose.



There's a lot to like about Secrets Manager, in particular the key rotation — if you haven't been obscuring your secure variables in your repos in the past, it allows for easy, hands-off rotation of these keys on a regular basis. This reduces risk in case of employee turnover or security breaches. And with encryption via KMS, you can limit access to whatever IAM users and roles actually need read/write access.

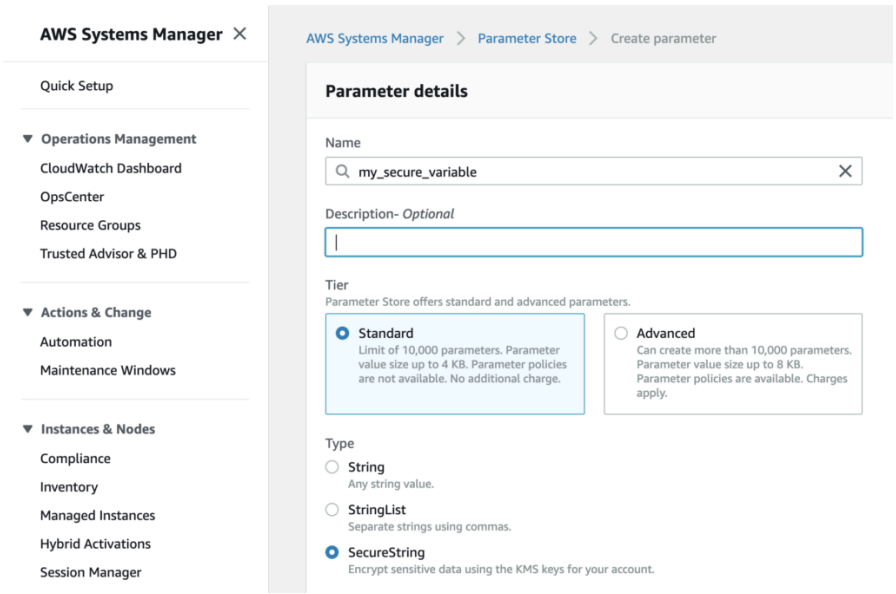
Secrets Manager stores secrets for $0.40/secret per month, and $0.05 per 10,000 API calls.

Systems Manager Parameter Store

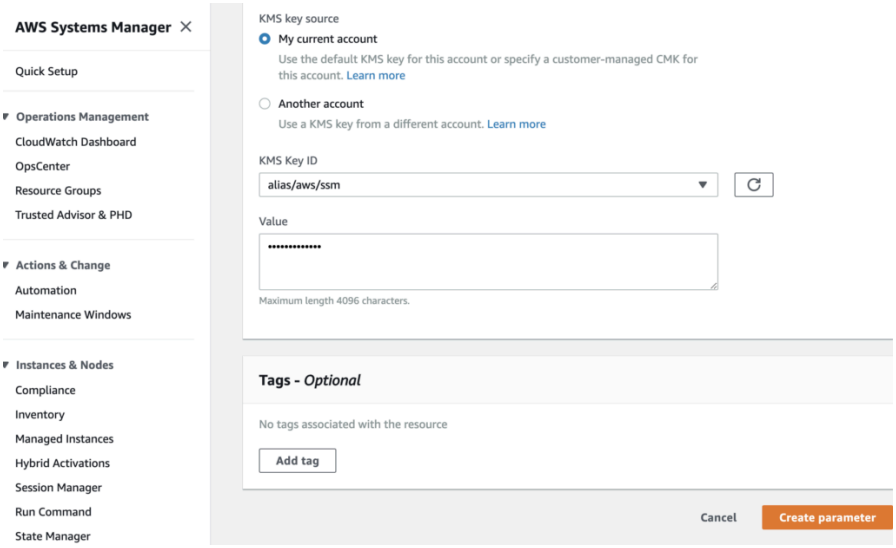
By comparison, AWS Systems Manager offers a Parameter Store which is a simple key-value pair storage option. It also offers encryption via AWS KMS, which allows the same security and simplicity

- Patch Management
- Role/Identity Association
- Scheduled commands
- Run commands on a subset of servers at once
- Organize resources into Resource Groups based on Tags
- Show compliance with Patching and Access/Permissions policies
- Store secure, encrypted variables in Parameter Store.

When it comes to storing parameters, the setup pane asks for a key name (which must be unique), and a value. You can store parameters as a basic String, a StringList, or a SecureString.

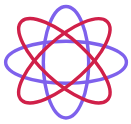


"Advanced" tier secrets are priced at $0.05/advanced parameter per month.



If you choose the Advanced tier, expiration policies can be set on the parameters stored as well. Just like with Secrets Manager, additional tags can be added, and the values can be encrypted with the KMS key of your choice, making access control for your secrets more simple.

To recap, Parameter Store may offer more simplistic key-value pair storage, but is much less expensive (even at the Advanced tier). Secrets Manager offers several different storage types, most of which center around database credentials, but does offer a more



TechGirlKB

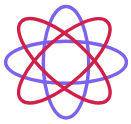
Ansible and Secret Management

With two options for secret management within AWS, it was difficult to know which to choose. We started with Secrets Manager, as Ansible offers both an aws_secret module, and an aws_secret lookup plugin.

aws_secret lookups

In our case, we were less interested in storing new secrets, and more interested in looking up the key and retrieving the value, for use in templates. That being the case, we chose to use the aws_secret lookup plugin. The example given in the documentation is:

```
- name: Create RDS instance with aws_secr
  rds:
    command: create
    instance_name: app-db
    db_engine: MySQL
    size: 10
    instance_type: db.m1.small
    username: dbadmin
    password: "{{ lookup('aws_secret', 'D
```



'aws_secret' reference, and the name of the secret. Unfortunately it was not as simple for us.

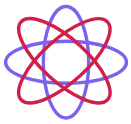
Firstly, we found that adding the region to the command was necessary, like so:

```
"{{ lookup('aws_secret', 'my_api_key',
region='us-west-1') }}"
```

That worked well enough in our vars_files to get through the deploy, provided the server running the ansible command had the proper IAM permissions. But, to my dismay, I found that this lookup didn't return the "value" of the key-value pair, but rather a json string with BOTH the key and the value (shown below).

```
[{ 'my_api_key', 'my_api_key_value' }]
```

Unfortunately the only way I could get it to return just the "value" of the simple key-value style Secret was to add additional parsing in a script. So, as of now anyways, it looks like the Ansible aws_secret lookup plugin is limited to database secrets usage.



Systems Manager Parameter Store option instead. As with Secrets Manager, Ansible also has Parameter Store functionality in the form of the aws_ssm_parameter_store_module and the aws_ssm lookup plugin. And again, since we're wanting to just read the value of secrets, we don't need to mess with the module — just the lookup plugin. Ansible provides the following examples (although there are more use cases shown in the documentation):

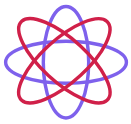
```
- name: lookup ssm parameter store in the
  debug: msg="{{ lookup('aws_ssm', 'Hello

- name: lookup ssm parameter store in nom
  debug: msg="{{ lookup('aws_ssm', 'Hello

- name: lookup ssm parameter store withou
  debug: msg="{{ lookup('aws_ssm', 'Hello

- name: lookup ssm parameter store in nom
  debug: msg="{{ lookup('aws_ssm', 'Hello
```

The examples given show easily enough how to use aws_ssm lookups within a playbook, but it can also be used in your vars_files like so:



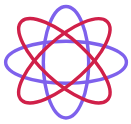
Providing your instance is setup with the proper IAM permissions to read SSM parameters and read access to the KMS key used to encrypt them (if SecureString was selected), your variables should populate into templates without having to store them in an vaulted file or vaulting/encrypting individual strings.

Automating Parameter Addition

If your project (like ours) has a lot of vars to store, you may find it very tiresome to add all the keys one by one into the Systems Manager panel in AWS. As a DevOps engineer, it made me cringe thinking of having to add the variables by hand. So, I made a script that uses the AWS CLI to upload parameters.

A couple notes:

- **This script assumes you have an AWS CLI config file setup at ~/.aws/config, with multiple AWS account profiles.** The one referenced is called “aws-main” — replace this with your own profile, or remove the line if you only have one profile.



replace as needed.

```
#!/usr/local/bin/bash -xe
declare -A vars

vars[env]=develop
vars[debug]=true
vars[key]="key_value"
#(more vars listed here...)

for i in "${!vars[@]}"
do
  aws ssm put-parameter \
  --profile "aws-main" \
  --name "dev_${i}" \
  --type "SecureString" \
  --value "${vars[$i]}" \
  --key-id "alias/dev-kms-key" \
  --tags Key=Environment,Value=Develop Ke
  --region "us-west-2"
done
```

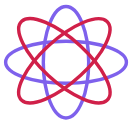
The bash script above declares an array “vars,” of which there are keys (env, debug, key) and values (develop, true, key_value). The loop uses the key as

values as needed per environment, and change tags/prefixes to reflect new environments. But this script helped cut the time to add parameters in 1/4 or more! Definitely a win in my book.

Conclusions

After some trial and error, here's a recap of what we learned:

- Secrets Manager is a more robust solution that offers rotation of secrets/keys. However, it is more expensive and charges for API calls.
- If you're looking to just populate the values of secrets for your variables in Ansible, SSM Parameter Store will work better for your needs.
- Ansible's aws_secret lookup works best for database Secrets.



TechGirlKB

Have any success or failure stories to share with either Secrets Manager or Parameter Store? Share in the comments, or contact me.

Share this:

Related

Adding version control to an existing application
March 31, 2019
In "Ansible"

Automate Patching Using AWS Systems Manager (SSM)
May 29, 2020
In "AWS"

Adding Nginx HSTS Headers on AWS Load Balancer
January 17, 2020
In "AWS"

Comments

Eric says
February 3, 2020 at 2:23 pm

Still a huge gap in the AWS secret management modules as of 2.9. I still have to use the AWS cli to put secrets and parameters.

I wish they had support like Terraform

Martin Stevens says
March 20, 2020 at 5:43 am

With the secret json returned.

{“secret1”: “we_are_hunting_rabbits”, “secret2”: “i_shall_call_him_george”}

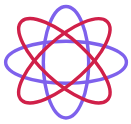
We should be able to:

– set_fact:
secret: “{{ lookup(‘aws_secret’, ‘poc-secrets’, region=’eu-west-1′)}}”
no_log: true

– debug:
msg: “{{ secret.secret1}}”

– debug:
msg: “{{ secret.secret2}}”

Reply



TechGirlKB

Your email address will not be published. Required fields are marked *

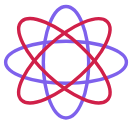
Comment

Name *

Email *

Website

☐ Notify me of follow-up comments by email.



TechGirlKB

Categories

Ansible
AWS
Git
Linux
Optimization
Performance
PHP
Scalability
Security
Uncategorized
WordPress

Search this website