There was an investigation (DF-81) regarding the solutions offered by AWS to store credentials and secrets, namely AWS Secrets Manager and AWS Systems Manager - Parameter Store .

Both solutions seem fairly similar, just to point out the main advantages for both of them:

- Very good flexibility on defining access permissions via IAM and encryption via KMS.
- Possibility to rotate secrets using AWS Lambda.
- Fair structured way of keeping the secrets grouped for each environment, project.
- Programmatic retrieval of secrets using the same PythonSDK - Boto 3.
- AWS CloudFormation support for both solutions.

AWS Secrets Manager

- Even though advertised to automatically rotate secrets, it mainly does automatic rotation just for AWS RDS, other secrets will have to use a custom AWS Lambda.
- History for changes made to each key/value pair not really straight forward.
- **COST**: $0.40 per secret per month - Per 10,000 API calls. (More details on costs here) With this investigation we had to start the default 30-day Trial Period also.
- More structured way of keeping the credentials. A "secret" is an entity which contains multiple key-value pairs.
- Cross Account Access if there is a need to share secrets with other AWS Accounts.
- Security compliance (ISO, PCI, etc.)

AWS Systems Manager - Parameter Store

- No native automatic secret rotation, only using custom AWS Lambda/Cloudwatch Event.
- Secrets manager was released in Apr. 2018 (and it *seems* to be an attempt to monetize parameter store by offering somehow the same features, wrapped a bit differently and **may probably** make Parameter Store **obsolete someday -** even though features are still being added)
- Should be **free of charge** for *standard* secrets which suit our use cases, *advanced* secrets have a cost though (More details on costs here).
- Secrets are stored in a path-like structure:`/PROJECT/ENVIRONMENT/VARIABLE1`

To fetch the credentials stored in Secrets Manager / SSM Parameter Store we attempted to use a different approach, right from Kubernetes instead of the standard python sdk, but setting it up

didn’t seem to be as straight forward as we expected.
More on this matter:

- https://uk.godaddy.com/engineering/2019/04/16/kubernetes-external-secrets/
- https://github.com/godaddy/kubernetes-external-secrets
- https://github.com/aws-samples/aws-workshop-for-kubernetes/tree/master/04-path-security-and-networking/401-configmaps-and-secrets#secrets-using-aws-secrets-manager

---

NOTE! While this investigation was ongoing, **GitLab** had a new feature released: a scope was added to the variables defined under Settings → CI/CD → Variables.

The main advantage of this approach is that one can simply add a wildcard prefix (like the `production` in the screenshot above) for a variable, making it only available for the environment matching that specific prefix.
We tested this approach, it works and adds no additional costs.

The main **downsides** at a first glance are:

- Difficult (not sure if it’s even possible for our current plan) to enforce permissions similar to the AWS IAM/KMS, to make sure variables don’t end up on a different environment than the one intended. It’s fairly easy to have a typo in one of this wildcards and the variable will go a different environment.
- No easy way to have a history for changes made to each variable (Makes the previous bullet point even worse).
- All the variables are stored in the same place, in the repository settings (Settings → CI/CD → Variables) and are sorted by key as default i.e. if there are 3 environment for a repository (`staging`, `production`, `feature environment`) and there is a `DATABASE_URL` variable defined for each of them, this variable will show up in the list three times, one next to another, the difference being the *scope prefix* and the *value* (which is by default hidden, so only a `*****` string are visible) , making it even more prone to typing errors.
- No easy way to rotate secrets, change encryption keys.
- Not sure if it’s a reliable, tested, ready for production feature. It just appeared in the GitLab interface, in a place where it used to be an error (the `-` minus icon you can see now on the right side of the `Scope` column in the screenshot above, used to be misaligned and was under the column header `Scope`, clicking it simply deletes the variable, no other confirmation required).