SOPS Encrypted secrets files

CONCEPTS

sops encrypts the values and leaves keys readable, so encrypted YAML/JSON files still diff and review well in git.

Plaintext file YAML, JSON, ENV, INI
↓ encrypt / edit
Data key AES256-GCM per file
↓ wrapped by
Master keys age, PGP, KMS, Vault

Each file has one random data key; master keys (age, PGP, cloud KMS) only wrap it, so you can add or remove people without re-encrypting values.

Commit .sops.yaml next to the secrets: it picks keys by file path so nobody has to remember flags.

EDIT & READ

sops edit *<file>*
Edit a file in your editor, re-encrypting on save

sops decrypt *<file>*
Print the decrypted file to stdout

sops decrypt -i *<file>*
Decrypt a file in place

sops decrypt --extract '["*<key>*"][*<n>*]' *<file>*
Print a single value

ENCRYPT

sops encrypt -i *<file>*
Encrypt in place using .sops.yaml rules

sops encrypt *<file>* > *<out>*
Encrypt to a new file

sops encrypt --age *<recipient>* *<file>*
Encrypt to age recipients

ENCRYPT (cont.)

sops encrypt --pgp *<fingerprint>* *<file>*
Encrypt to PGP keys

sops encrypt --encrypted-regex '^(data|stringData)$' *<file>*
Encrypt only keys matching a regex

CHANGE VALUES

sops set *<file>* '["a"]["b"]' '"*<value>*"'
Set a value; value is JSON

RUN WITH SECRETS

sops exec-env *<file>* '*<cmd>*'
Run a command with values as env vars

KEYS & CONFIG

sops updatekeys *<file>*
Sync a file's keys with .sops.yaml

sops rotate -i *<file>*
New data key, re-encrypt all values

ENVIRONMENT

SOPS_AGE_KEY_FILE=*<path>*
Age identity file for decryption

SOPS_AGE_KEY=*<key>*
Age secret key given directly

.SOPS.YAML

Creation rules (first match wins)

```
creation_rules:
  - path_regex: \.prod\.yaml$
    age: age1prod...,age1admin...
  - path_regex: secrets/.*
    age: age1dev...
    encrypted_regex: '^(data|
stringData)$'
  - age: age1admin...
```

A rule without path_regex matches everything, so put it last. Run sops updatekeys after changing recipients on existing files.