SOPS Encrypted secrets files

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

CONCEPTS

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

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

EDIT & READ (cont.)

sops decrypt --output *<out>* *<file>*
Write plaintext to another file

sops decrypt --output-type json *<file>*
Convert format while decrypting

sops edit -s *<file>*
Show master keys while editing

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

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

sops encrypt --kms *<arn>* *<file>*
Encrypt with AWS KMS keys

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

ENCRYPT (cont.)

sops encrypt --unencrypted-regex '*<regex>*' *<file>*
Leave matching keys in plain text

sops encrypt --input-type *<type>* **--output-type** *<type>* *<file>*
Force formats (json, yaml, dotenv, ini, binary)

CHANGE VALUES

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

sops set --value-stdin *<file>* '["a"]["b"]'
Read the value from stdin

sops unset *<file>* '["a"]["b"]'
Delete a key

RUN WITH SECRETS

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

sops exec-file *<file>* '*<cmd>* {}'
Run a command with a temp file path in {}

RUN WITH SECRETS (cont.)

sops decrypt --output-type dotenv *<file>*
Print as KEY=value lines

KEYS & CONFIG

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

sops updatekeys -y *<file>*
Same, without the confirmation prompt

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

sops rotate -i --add-age *<recipient>* *<file>*
Add an age recipient while rotating

sops rotate -i --rm-age *<recipient>* *<file>*
Remove an age recipient

sops --config *<path>* *<command>*
Use a specific config file

ENVIRONMENT

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

ENVIRONMENT (cont.)

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

SOPS_AGE_RECIPIENTS=*<list>*
Age recipients to encrypt to

SOPS_EDITOR=*<editor>*
Editor for sops edit

.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.

Other rule keys: kms, gcp_kms, azure_keyvault, hc_vault_uris, pgp, unencrypted_regex, encrypted_suffix.