

Sylva WG6

Git strategy (repos, branches, tags)

15/5/2025

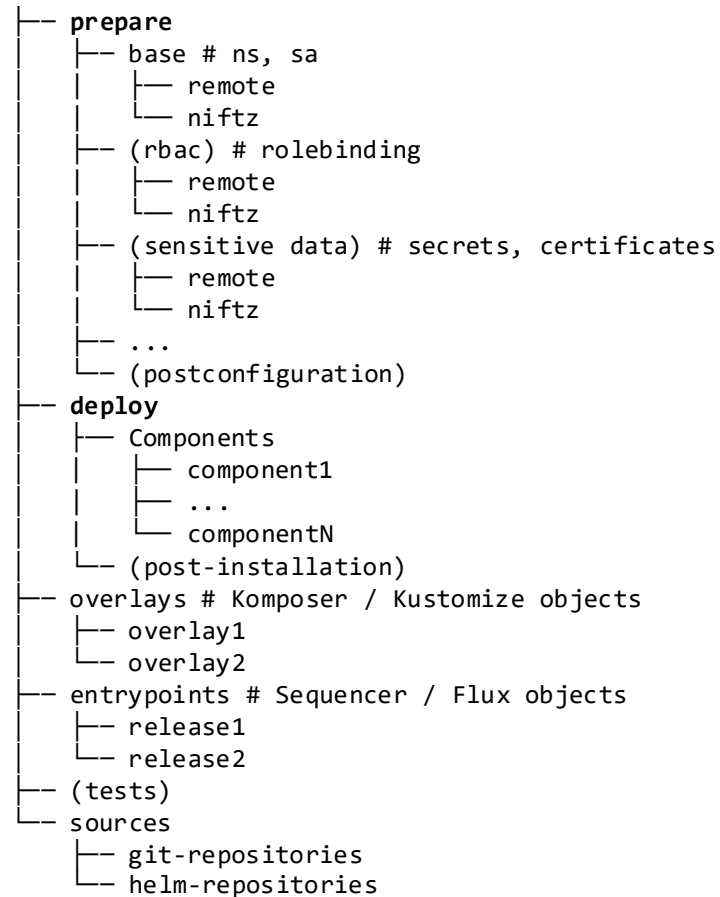
Git Intents shall allow us to

- Onboard a CNF – put an existing Network function under Git control
- Install a CNF (from scratch)
- Terminate a CNF (what does it mean...)
- Upgrade a CNF
- Perform CNF Change management
- Execute Tests

For the CNF

- We want to
 - Reference versions we validate (not all the published versions)
 - Avoid having multiple sources of truth
 - Manage multiple versions/instances in //
 - Be able to apply a patch to any version
 - Be able to compare easily the difference between the versions
 - Write intents that can be shared with operations (not necessarily GitOps experts)
 - Be clear on what is deployed on the automation enabler / what is deployed on the target cluster

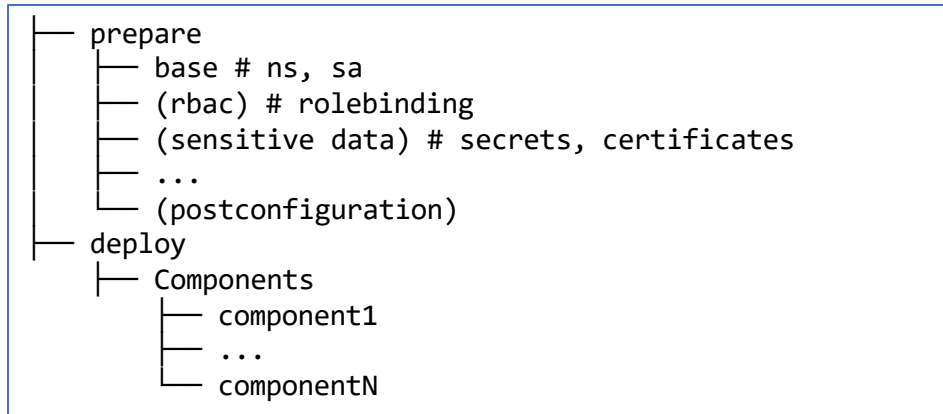
Git Structure: all in one to start



Usually the vendor approach as they want to "sell" GitOps packages

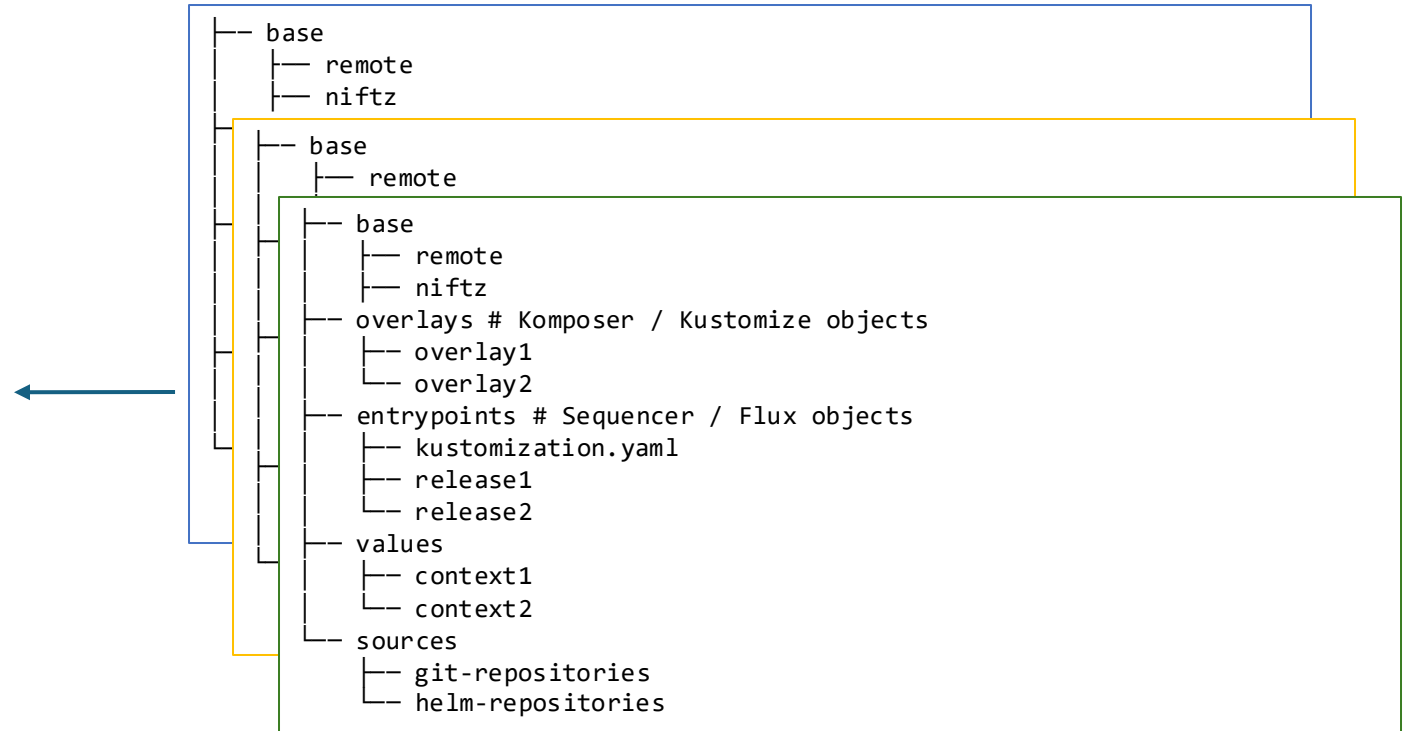
Problem: we couple here the CNF and its configuration => risk to create as many Sources of Truth as Repositories

Git Structure: 2 repositories approach



Generic ~ helmreleases / vendor operators

Responsibility: Technical Skill Center



Specific: values/secrets, config, post-configuration
Target Cluster

Could be 1 repo for all deployments but
RACI/RBAC/Readability issues => 1 specific repo
pointing to generic / operation organization

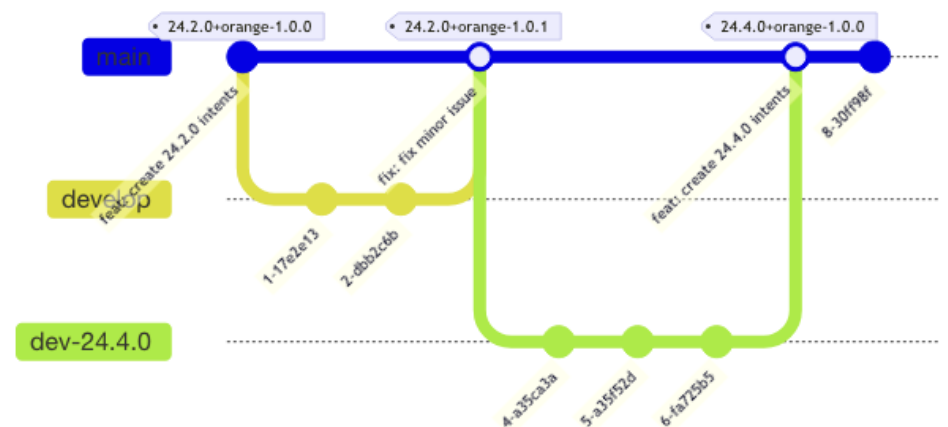
Responsibility: Operational Skill Center

Branches & tags

- The GitOps system must have clear Git references
 - Branch
 - Tags
- Both approaches are possible:
 - release branches versus releases associated with tags
 - We can protect branches and tags
- Versioning shall be self explicit (all the info in the tag or the branch)
- We like to view directly what is deployed when looking in GitLab and not guessing which branch is the right one
- Tag generation is easy with semantic release and can be adapted for Vendor inputs even with double versioning
- Tags are well adapted for renovate. MR will be suggested on tag changes

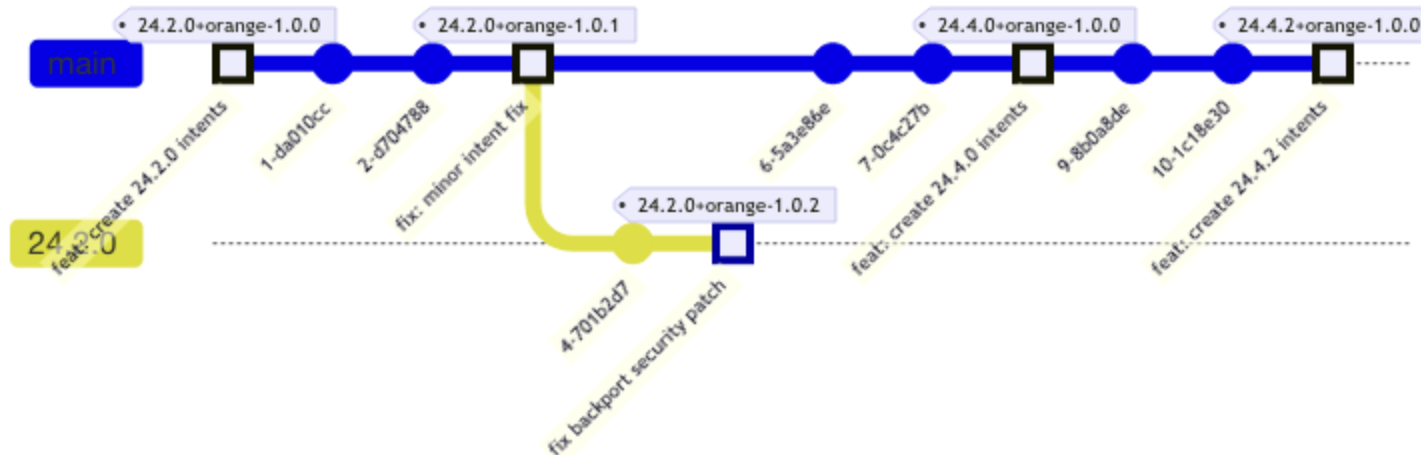
Branches & tags

- Protected branch: main
- Tags: lots of tags following semantic release (fix, feat, breaking changes)
- Tags: double versioning format
 - <vendor version>+<orange version>, vendor version set in a txt file (MR needed to modify it), orange version managed by semantic
 - e.g, 24.2.0+orange-1.0.1, 24.7.2+orange-1.2.3
- Dev branches (upgrade, patch, feature change)



The 1-protected branch problem

- If we have 1 branch hosting lots of tags, how to apply changes in "old tags"
- Note as we split Helmrelease versions and configuration version, the problem deals only with Vendor inputs not with configuration changes (that have a simple semantic versioning x.y.z)
- In this case we MUST branch and protect the new branch



The 1-protected branch problem

- Vendor Backporting is rare for the moment
- Upgrading is usually the solution, we may expect that we have already a tag corresponding to the upgrade as we validated the version