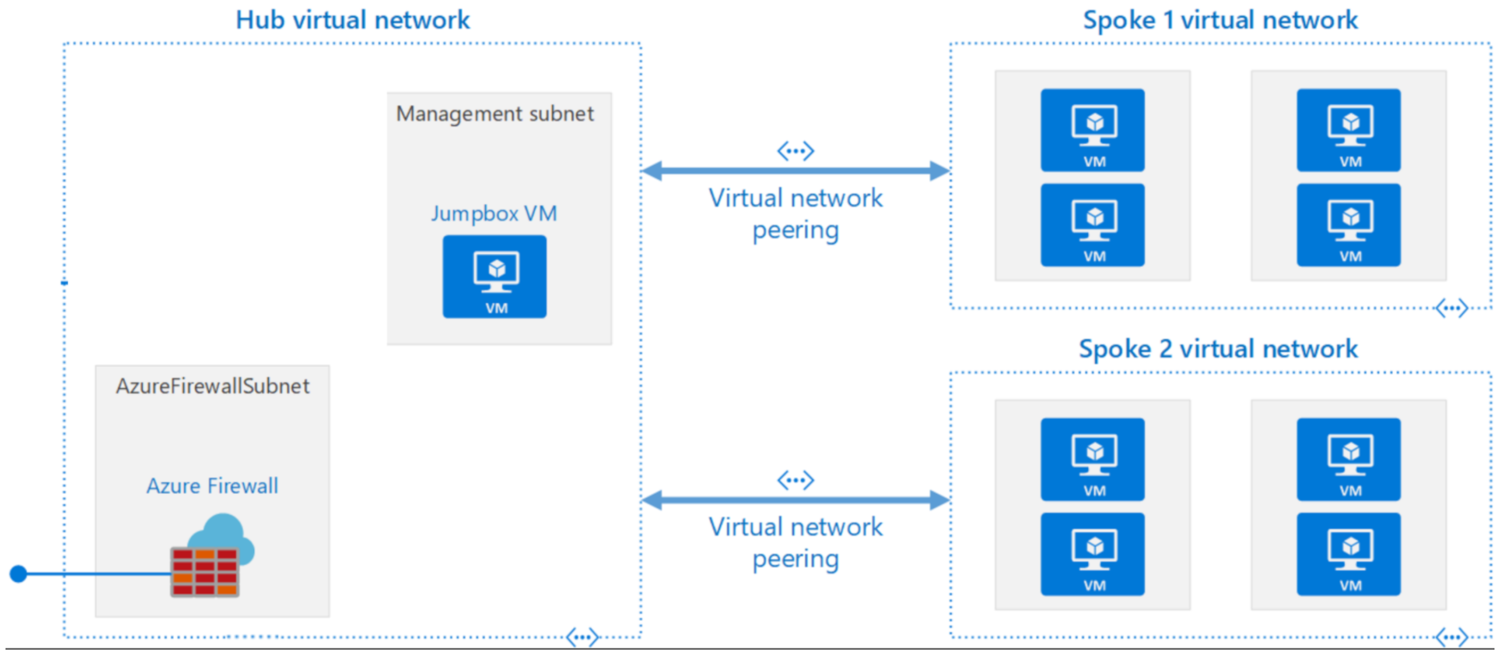
Continuous integration and deployment @ Snow

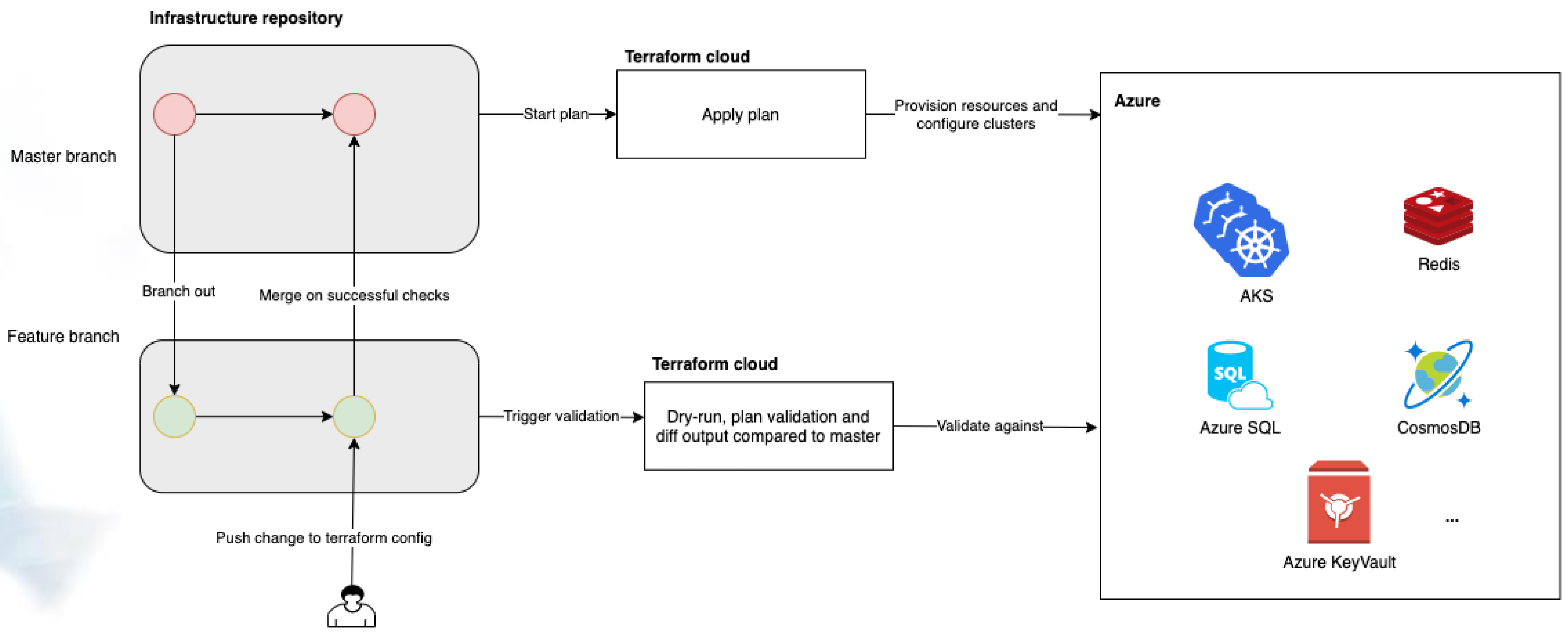
Hub and spoke architecture

- All workloads running on Azure
- Shared resources in the hub with workload isolation in the spokes
  - Such as DNS, Firewall and container registries
  - Azure subscription separation of in the spokes
- Environment subscription separation
  - Workload isolation in spokes with VNet pairing only with hub
- Multi-region resource group separation
  - Resource naming convention is important and enforced



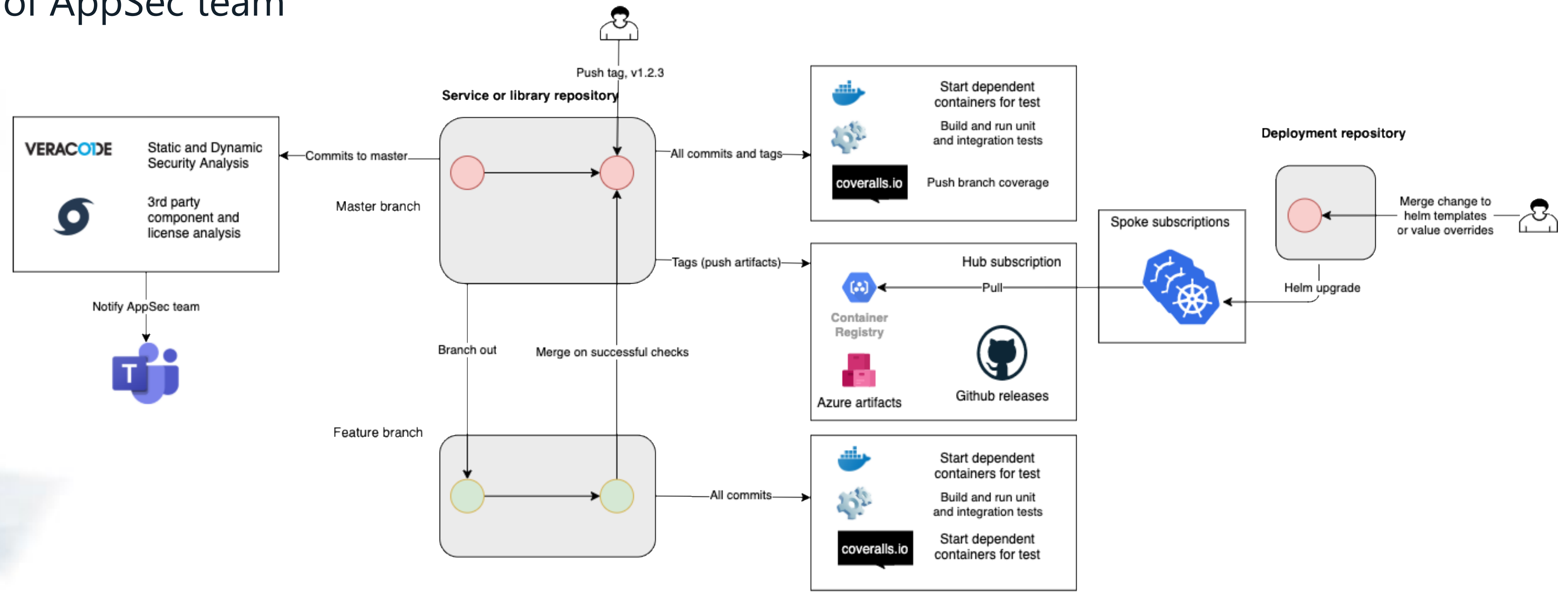
Terraform deployment

- Mono repository with terraform configuration for the infrastructure
- Provisions storage, AKS, load balancers and more
- Adds k8s namespaces to the clusters
    - Assigns namespace RBAC access through Azure AD
- Adds config maps and secrets from the provisioned resources to the clusters and namespaces based on checked in configuration


Infrastructure repository
Master branch
Branch out
Merge on successful checks
Feature branch
Push change to terraform config
Start plan
Terraform cloud
Apply plan
Provision resources and configure clusters
Trigger validation
Terraform cloud
Dry-run, plan validation and diff output compared to master
Validate against
Azure
AKS
Redis
Azure SQL
CosmosDB
Azure KeyVault
...

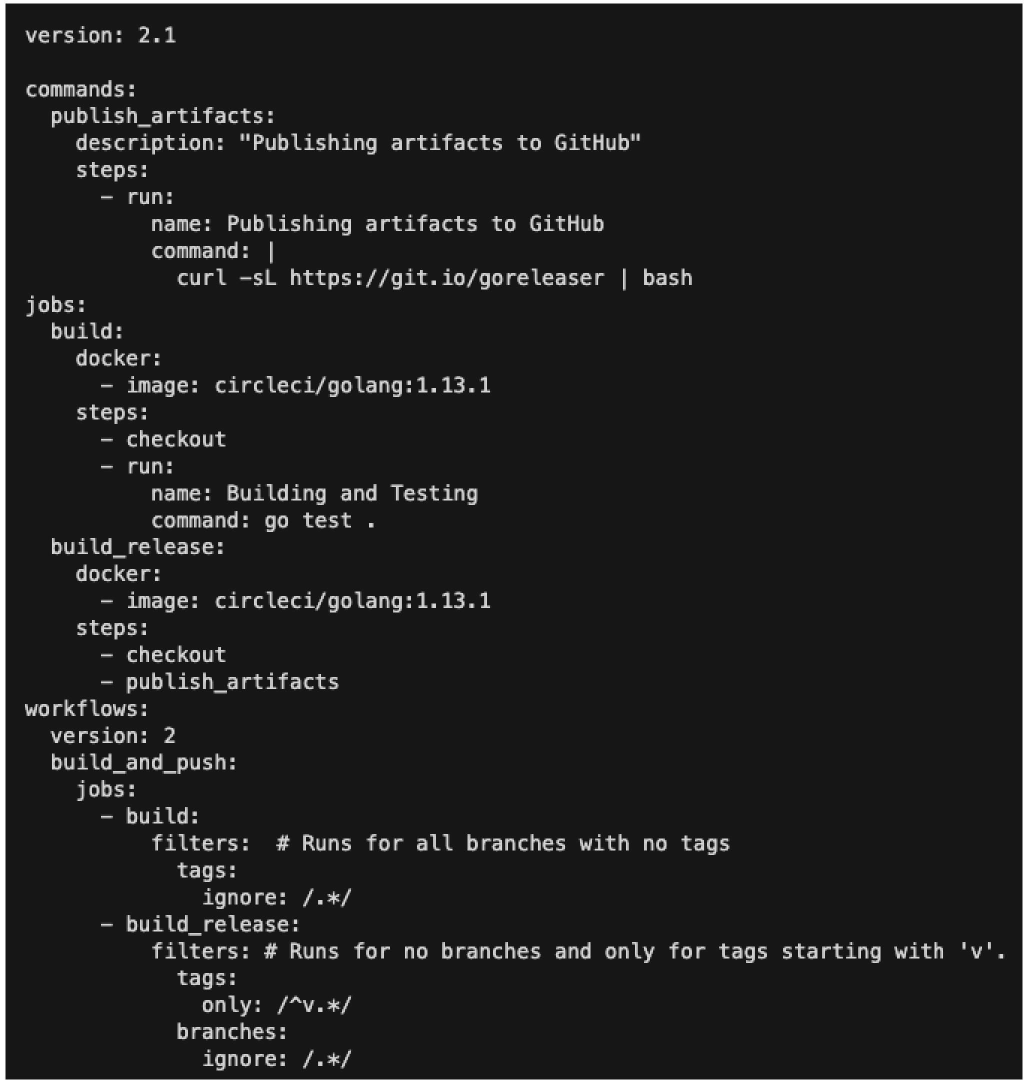
Continuous Integration - Overview

- Builds runs in CircleCI on commits and tags
- Build configuration checked into each repository (example on next slide)
- Builds, unit tests and integration tests on each commit
- Container builds, Nuget packages, NPM packages and go releases published on new tags
- Static, dynamic (Veracode) and 3rd party component security scan (CycloneDX) runs on master branch commits and publishes delta score into MS Teams channel of AppSec team


Push tag, v1.2.3
Service or library repository
VERACODE
Static and Dynamic Security Analysis
3rd party component and license analysis
Commits to master
Master branch
All commits and tags
Start dependent containers for test
Build and run unit and integration tests
coveralls.io
Push branch coverage
Deployment repository
Merge change to helm templates or value overrides
Spoke subscriptions
Helm upgrade
Tags (push artifacts)
Hub subscription
Pull
Container Registry
Azure artifacts
Github releases
Notify AppSec team
Branch out
Merge on successful checks
Feature branch
All commits
Start dependent containers for test
Build and run unit and integration tests
coveralls.io
Start dependent containers for test

Continuous Integration – CircleCI example configuration

- Global configuration in contexts where the parameters can be mapped in the configuration (not shown here)
- Workflow – Set of rules for defining jobs and ordering
- Job: A set of commands to be run in sequence
- ORB – Reusable package of YAML configuration
    - No support for private ORBs as of yet

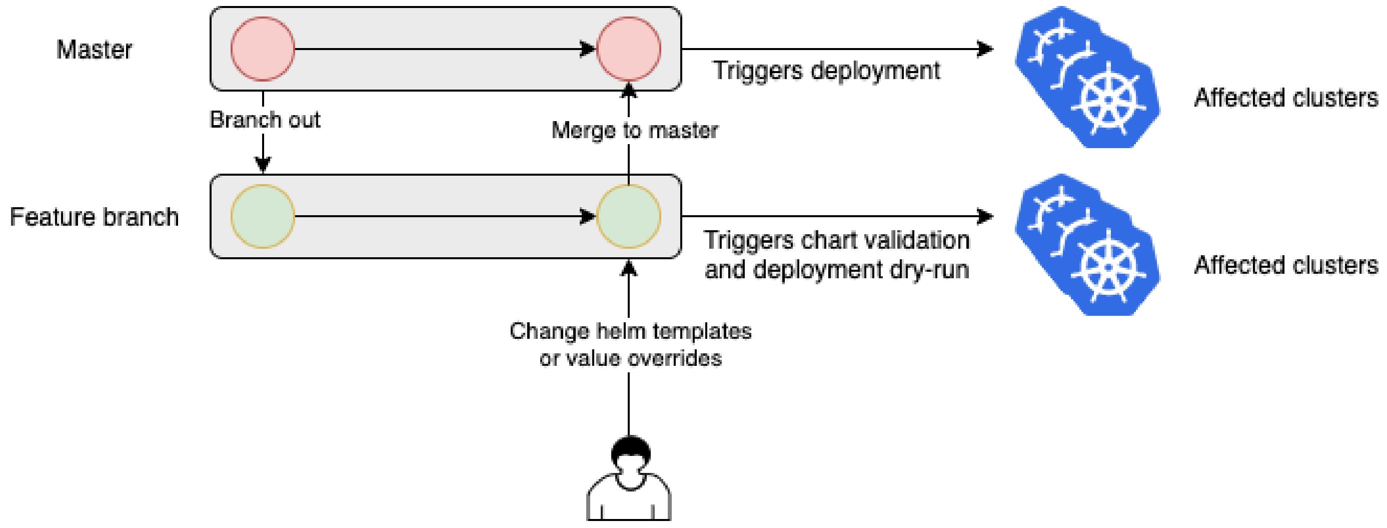


```yaml
version: 2.1

commands:
  publish_artifacts:
    description: "Publishing artifacts to GitHub"
    steps:
      - run:
          name: Publishing artifacts to GitHub
          command: |
            curl -sL https://git.io/goreleaser | bash
jobs:
  build:
    docker:
      - image: circleci/golang:1.13.1
    steps:
      - checkout
      - run:
          name: Building and Testing
          command: go test .
  build_release:
    docker:
      - image: circleci/golang:1.13.1
    steps:
      - checkout
      - publish_artifacts
workflows:
  version: 2
  build_and_push:
    jobs:
      - build:
          filters:  # Runs for all branches with no tags
            tags:
              ignore: /.*/
      - build_release:
          filters: # Runs for no branches and only for tags starting with 'v'.
            tags:
              only: /^v.*/
            branches:
              ignore: /.*/
```

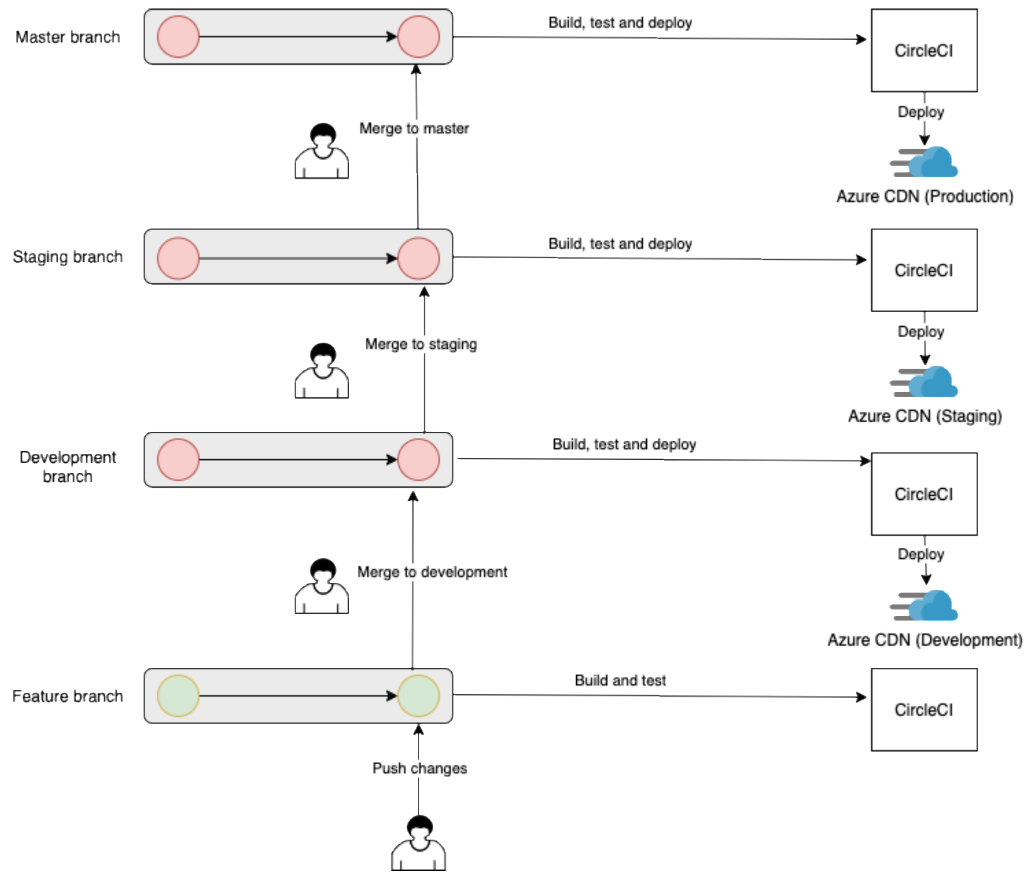
Continuous deployment - Services

- Mono repository with helm charts and value overrides for all services
  - Overrides on global level (all clusters), environment (prod, dev, stage) and individual clusters
  - Config map and secret references mapped to environment parameters in chart templates
  - Other secrets checked in encrypted using helm secrets with SOPS and deployed through helm
- Helm upgrade used with helm secrets plugin during deployment
- Validations and dry-runs on the development branch and cluster deployments on merge to master



Continuous deployment – Static content

- Azure CDN is used for static content
- Built, tested and deployed through CircleCI



Current challenges and upcoming items

- Helm upgrade problem (https://github.com/helm/helm/issues/5595)
  - After a failed deployment (deployment with state Failed), following helm upgrades fails
    - Adding --force to the upgrade was advised in the thread which seemed a bit dangerous for various reasons (especially when there are persistent volumes involved)
    - Workaround we have used on this condition is to manually reset the deployment status for the failed release from FAILED to DEPLOYED and re-deploy under controlled circumstances.
    - Supposed to be fixed in Helm v3.2.1 which was released on the 8th of May 2020 (Issue created 12th of April 2019) – But people where still reporting the similar issues in the thread on this version (on cancelled state)
- Upcoming plans for evaluating spinnaker (https://spinnaker.io/) and Flux (https://github.com/fluxcd/flux) as an alternatives to current deployment with helm as the templating engine
- Need to find a good way to do automatic promotion of services from development to staging to production after successful checks in each environment
- An urgent need to speed up security scans up to x10 times and run them as part of the regular CI/CD flow.

snow

Thank you

info@snowsoftware.com
snowsoftware.com