LF NETWORKING
Developer & Testing Forum

ONAP Component Build and Deployment Thru CD

ONAP Streamlining – The Process

November 2023

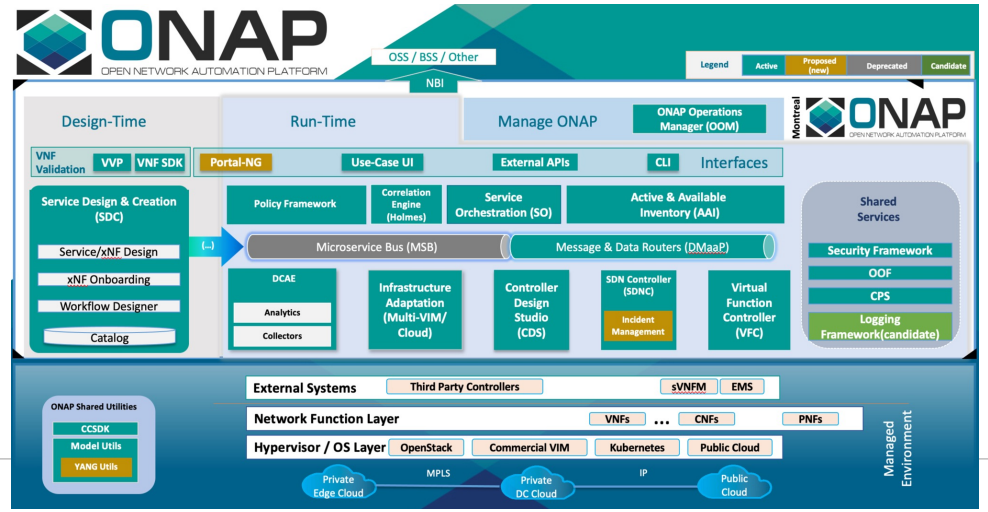
Presenter:
Andreas Geissler (DT) – OOM PTL
Byung-Woo Jun (Ericsson) – ARCCOM Chair, TSC

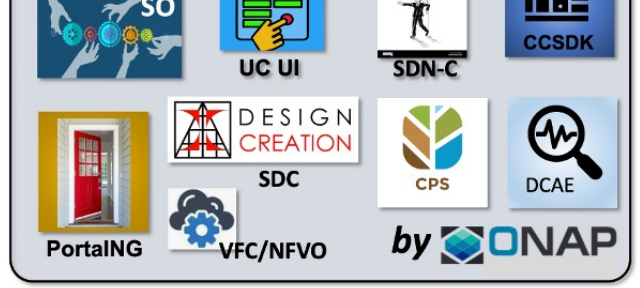
https://lfnetworking.org

ONAP Streamlining – Transformation

LFN Developer & Testing Forum

- Thru ONAP Streamlining, **ONAP is no longer a platform**, rather it provides various network automation functions, and security reference configuration in LFN.
- ONAP enables individual ONAP function build, and component deployment thru CD
- Build use cases for repository-based E2E service, NS, CNF and CNA onboarding, and CD-based ONAP component triggering mechanisms with abstracted interfaces for choreography (that is shown in Nephio architecture)
  - Standard-based abstracted interfaces with declarative APIs
  - Each component is autonomous and invoked from any level of Network Automation, by leveraging CD mechanisms – e.g., GITOps and CD readiness
- ONAP will become more intent-based and declarative, and bring in more AI, conforming to standards such as 3GPP, TMForum, ETSI, IETF, ORAN, etc.
  - Extend UUI User Intent support and AI-based natural language translation, by applying coming 3GPP and TMForum models and APIs
  - Delegate resource-level orchestration to functions from the external community
- ONAP continues to support the Service Mesh, Ingress, OAuth2, IdAM-based authentication and authorization, and consider sidecar-less solution for NF security.





ONAP focuses on Network Automation Functions

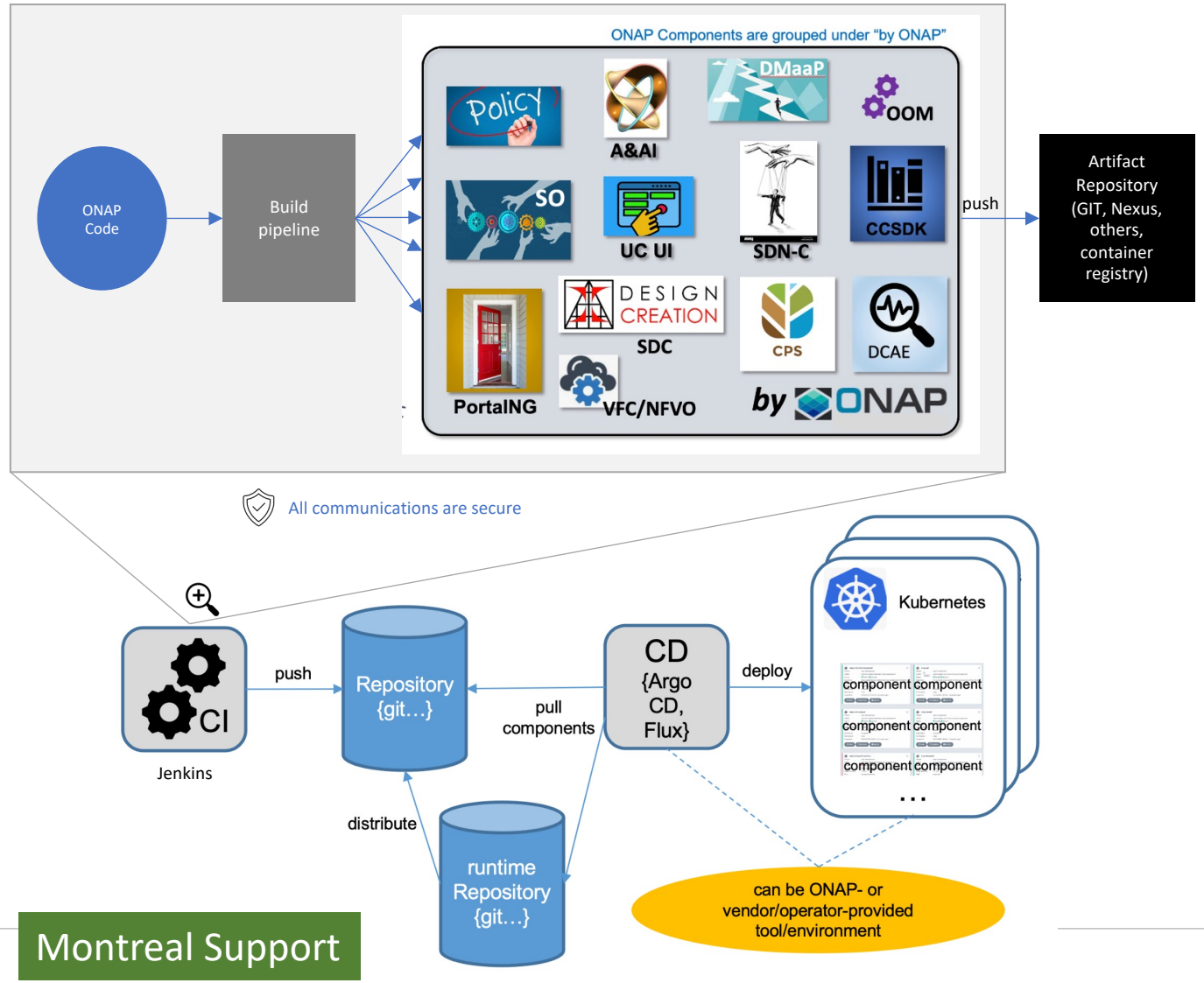
- Modular
- individual
- interface abstraction
- loose coupling
- Extensibility
- Interchangeability
- Autonomous
- Declarative
- CI / CD ONAP components and E2E Service, NS, CNF & CAN handling

ONAP Streamlining – Component Design, Build & Deployment

LFN Developer & Testing Forum

- ONAP Components are independently deployable pieces of software, built out of one or more microservices
  - Modular
  - Autonomous
  - Extensible and substitutional
- ONAP Network Automation processes will manage more intent-based operations using AI/ML.
  - Manage user and other Intents and translations
  - Study on TMForum & 3GPP Intent models and APIs
- ONAP components conform to the standards and de facto specifications to enable plug and play and pick-and-choose facilitation.

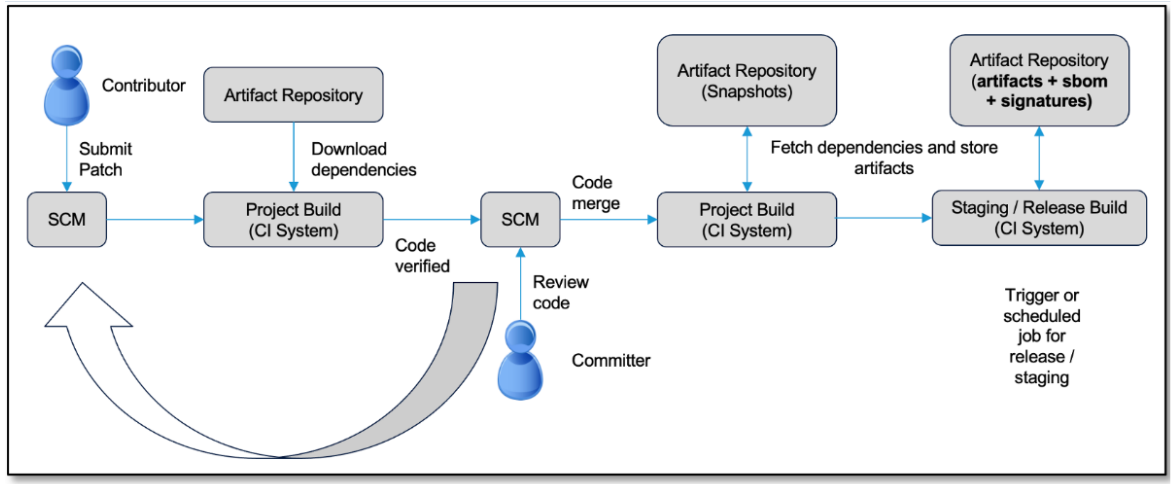
- ONAP repository-based SW management enables smaller imperative actions that can be triggered by different events in the orchestration and SW LCM flow.
  - Events can trigger different types of deployment automation jobs or chains of automation jobs (pipelines).
- In Jenkins, ONAP OOM build scripts will be used for ONAP component builds and will store built ONAP components into the Artifact Repository (e.g., Nexus). This can be changed.
- CD (e.g., ArgoCD, Flux, others) will be used to pick and choose ONAP components.



Montreal Support

ONAP Component Individual Build

- Leveraging the existing LF-based CI pipeline, builds ONAP components individually
  - Check-in ONAP component code and triggering build processes
  - Thru the CI pipeline, each ONAP component will be built by scripts (e.g., modified OOM, or project-own scripts), along with SBOM
  - Secure CI pipeline will be applied.
- Project Helm chart separation from the master Helm chart, and adding individual versioning
  - Currently, all the ONAP component helm charts have the same version number (e.g., 13.0.0). for a start,
    - e.g., projects with PTLs can start with 13.0.0. as the major Montreal release, and they can play with minor version(s) based on their release cycle, e.g., 13.0.1, 13.1.0… Projects without PTLs (or no improvement) will have the major Montreal version, e.g., 13.0.0
    - Other options: see, Break ONAP's monolithic version schema (by Florian Bachmann), https://wiki.onap.org/display/DW/Proposal%3A+Break+ONAP%27s+monolithic+version+schema
- PTLs determine granularities of project function exposures, e.g., exposing sub-components, use of flags for sub-component installation
- A common job will create all the ONAP charts.
- Having a separate job per component will be investigated

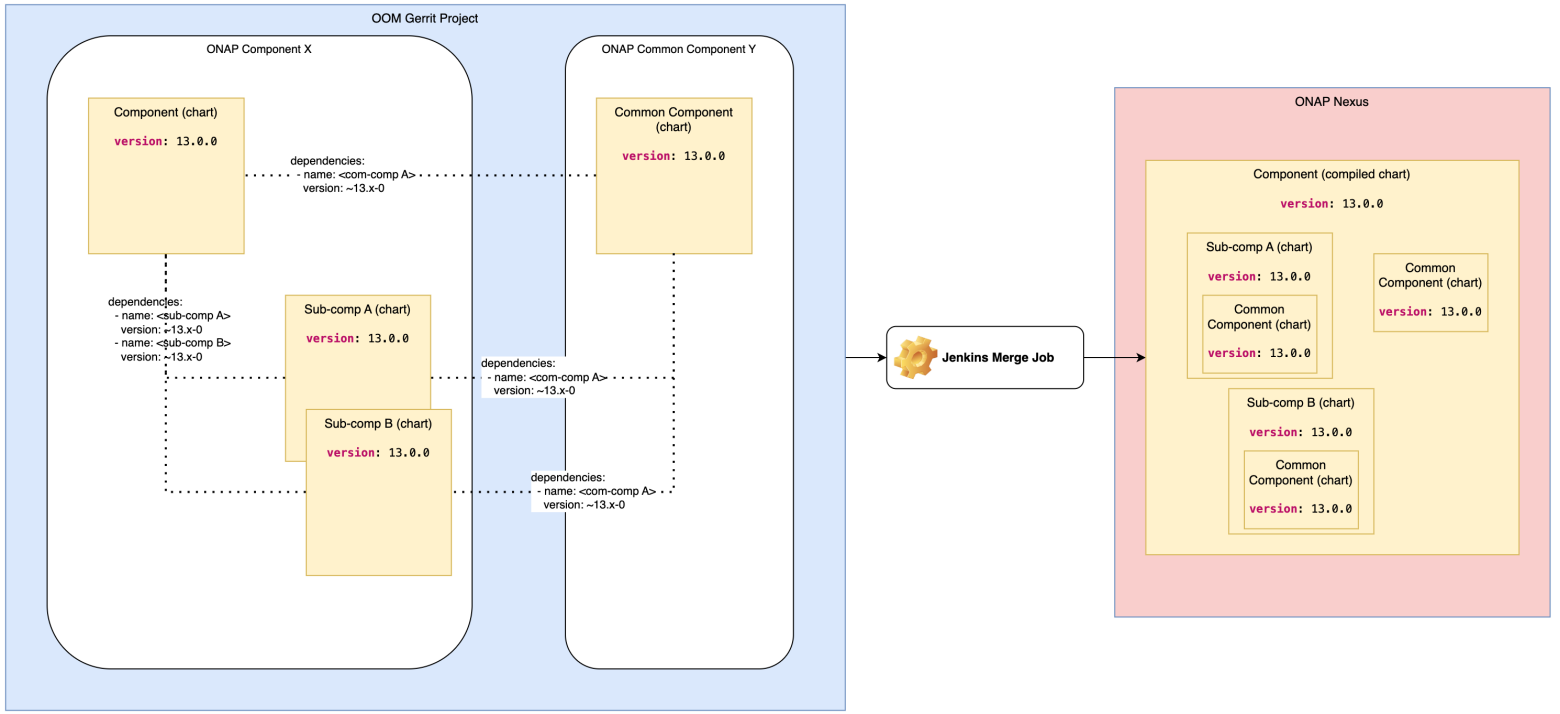



```yaml
apiVersion: v2
description: ONAP Service Orchestrator
name: so
version: 13.0.0

dependencies:
  - name: common
    version: ~13.x-0
    # local reference to common chart, as it is
    # a part of this chart's package and will not
    # be published independently to a repo (at this point)
    repository: '@local'
  - name: readinessCheck
    version: ~13.x-0
    repository: '@local'
  - name: mariadb-galera
    version: ~13.x-0
    repository: '@local'
    condition: global.mariadbGalera.localCluster
  - name: repositoryGenerator
```

ONAP Helm Chart Dependencies

- Past State:
  - All charts are managed in one project: https://git.onap.org/oom/
  - Chart dependencies are using wildcards (main chart will include the latest version of a subchart or common charts)
  - All charts are built (and stored in Nexus) after a merge
  - Currently, “appVersion” is not used in the charts
- Decision:
  - Use wildcards (e.g., ~13.x-0) in dependencies
  - Start with <rel>.0.0 every ONAP release for all charts (e.g., 13.0.0)
  - When a “common” component (template, DB chart) is changed, it will b4e included automatically in a component chart.
- Guidelines:
  - Component chart version update needs to done in a separate patch; all component chart versions (main + subchart) have to be updated
  - It is recommended to also set and maintain the “appVersion” in a chart
  - Merge Job
    - Currently, common job to create all ONAP charts
    - possible enhancement: separate jobs per component → to be investigated



https://wiki.onap.org/display/DW/ONAP+Helm+chart+dependencies

ONAP Helm Versioning Plan

LFN Developer & Testing Forum

- At ARCCOM, Florian Bachmann (DT) presented:
  - **ONAP version Schema options**, https://wiki.onap.org/display/DW/Proposal%3A+Break+ONAP%27s+monolithic+version+schema
    - Three possible solutions were proposed
    - Option 1: Separate Marketing version (e.g., Montreal / 13) and component version
      - The component version can be evolved based on the component feature / API changes individually, by following the SemVer scheme (SemVer.org)
      - It could be a target goal, but considering ONAP build impacts, the Option 2 is preferred for Montreal
    - Option 2: Use Marketing version as the MAJOR version
      - The Marketing/Major version will represent the usual ONAP update cycle, e.g., Montreal/13
      - All ONAP components excluding unmaintained ones will start with the same Marketing/Major.Minor.Patch version (e.g., 13.0.0) at the beginning
      - Within the marketing/major release, each ONAP component can have multiple minor / patch versions depending on its development cycle(s)
    - Option 3: Leave it as is. - this is “not” applicable for ONAP Streamlining
- For Montreal release, the Option 2 is chosen. It would be a study item for migrating from the Option 2 to the Option 1 in the future

Short-term plan (e.g., Montreal)

- for Montreal (Marketing version 13, and therefore all MAJOR numbers are 13 as well)
  - AAI: 13.4.6
  - SDC: 13.2.1
  - CDS: 13.69.0
  - Unmaintained Component: 13.69.1
  - Kafka: 7.1.4
  - Keycloak: 9.1.4

Long-term plan

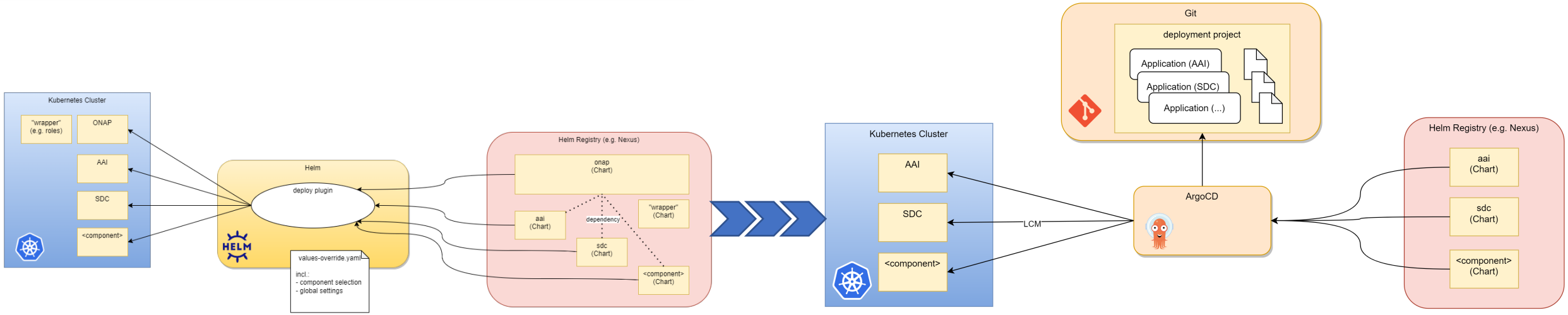
- for a future release (Marketing version xyz)
  - AAI: 29.4.6
  - SDC: 43.2.1
  - CDS: 89.69.0
  - Unmaintained Component: 6.69.1
  - Kafka: 7.1.4
  - Keycloak: 9.1.4

Deployment evolution

LFN Developer & Testing Forum

- Current “platform” deployment in “OOM”:
  - Using a “umbrella” ONAP chart with component dependencies
  - Usage of “common” wrappers (roles, registry,…)
  - Helm deployment using a selfmade “deploy” plugin

- Target:
  - Individual deployment the “GitOps” way by using tools like ArgoCD, Flex
  - Already now working (DT)

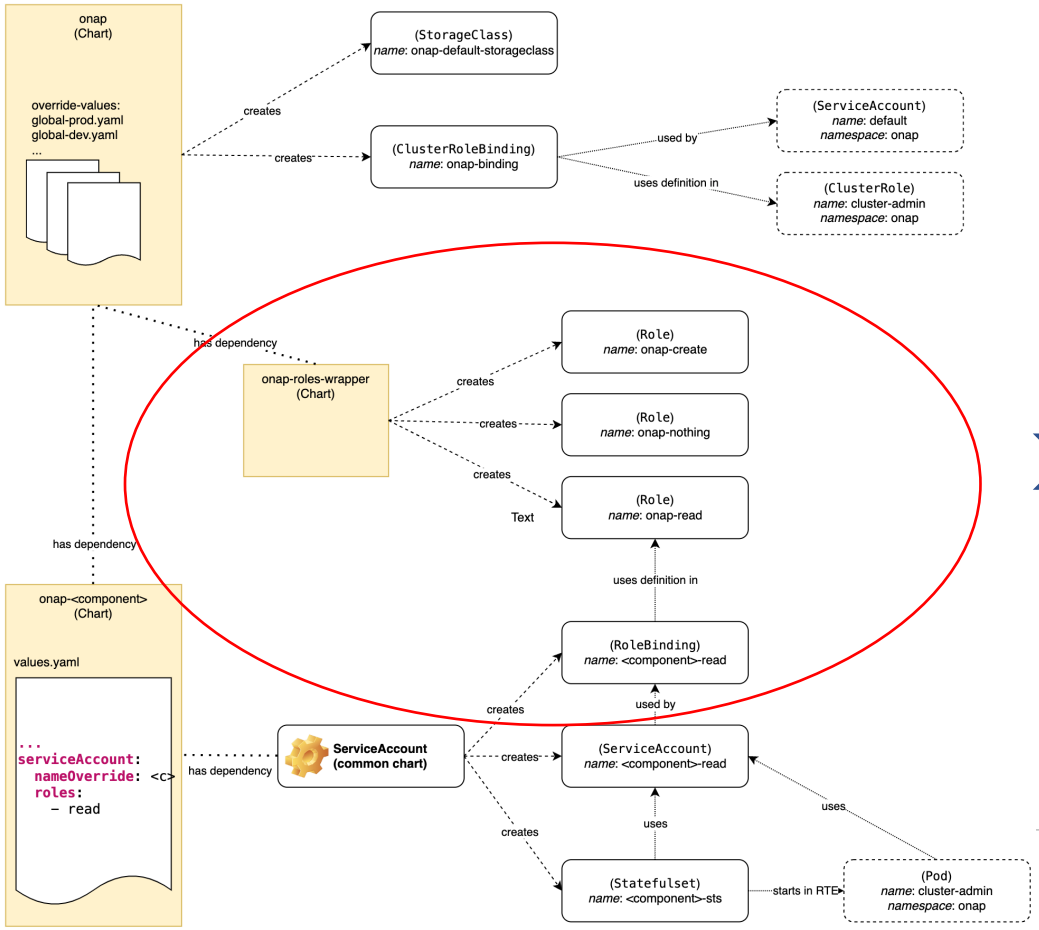


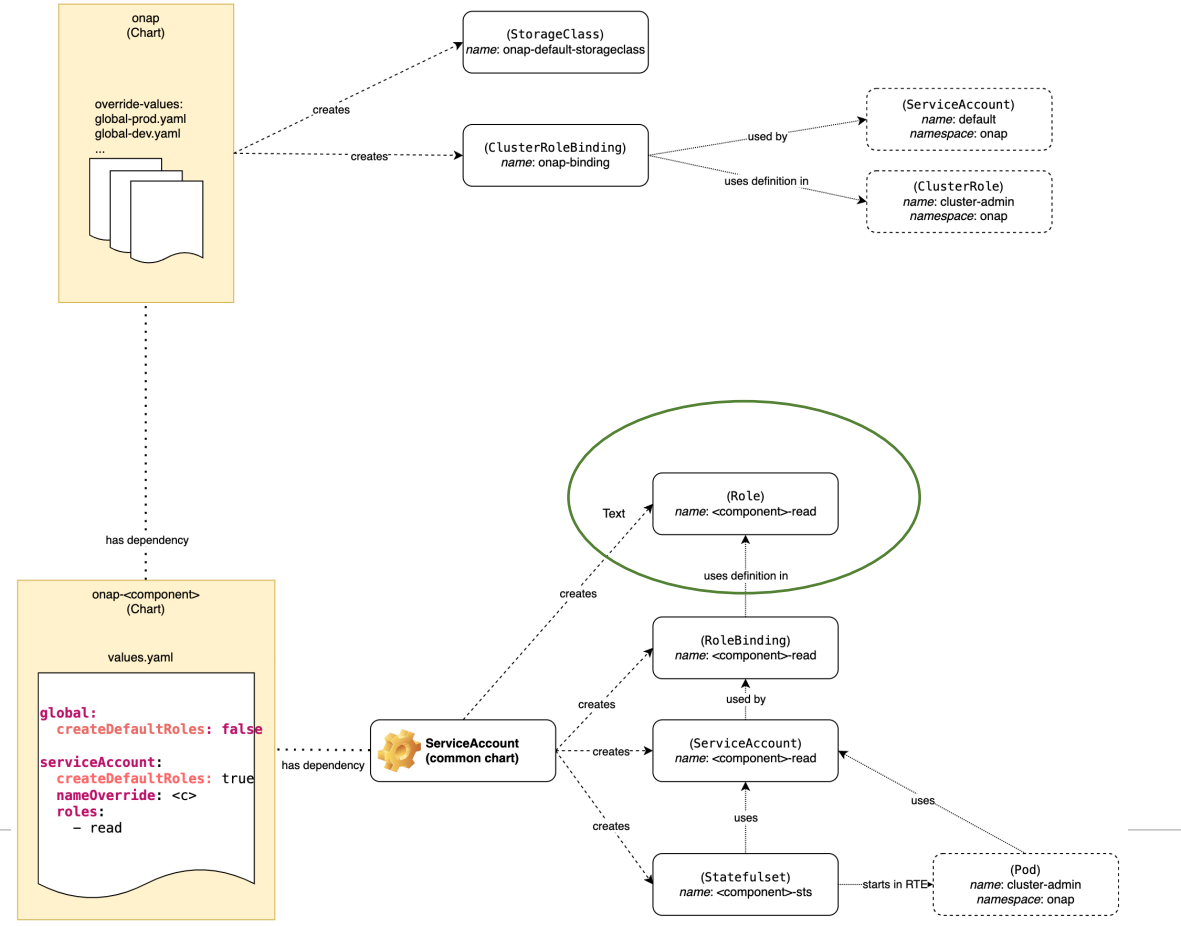
See: https://wiki.onap.org/display/DW/ONAP+deployment+evolution

Removal of shared “wrappers”

- Current issues:
  - ClusterRoleBinding “onap-binding” is missing and need to be added before deployment
  - “onap-roles-wrapper” is required, as onap-* roles are used by the “ServiceAccount” chart
  - ServiceAccount chart uses a naming schema for the default ”role” binding

- Solution:
  - The common chart “ServiceAccount” is extended to support the “default” Role creation
  - A new parameter “createDefaultRoles” are used
  - https://wiki.onap.org/display/DW/ONAP+deployment+dependencies





GitOps - Cloud Native Agility and Reliability

- See DDF (06/22) Florian Bachmann https://wiki.lfnetworking.org/pages/viewpage.action?pageId=68792723

Solution

GitOps is a set of modern best practices for deploying and managing cloud native infrastructure and applications.

Based on our experience operating a full cloud native stack

GitOps manages the whole stack:

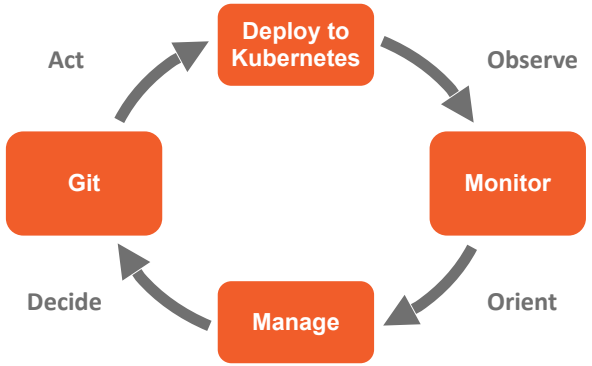
- Cluster and application versioned configuration
- Security and policy enforcement
- Monitoring and observability
- Continuous Deployment of workloads

Benefits

- **Complete platform**: Single platform for infrastructure, core components and applications.
- **Productivity**: Dramatically increase deployments and faster feedback and control loop,
- **Reliability**: Enables cluster and application operator model with standardised tooling.
- **Compliance and Security**: Enforces standard security policy and an audit trail
- **Multi-cloud and on-premise**: Deploy a complete cluster from git with all applications.

Vision

- All application deployments, application operations and cluster management operations under one platform with a common workflow.

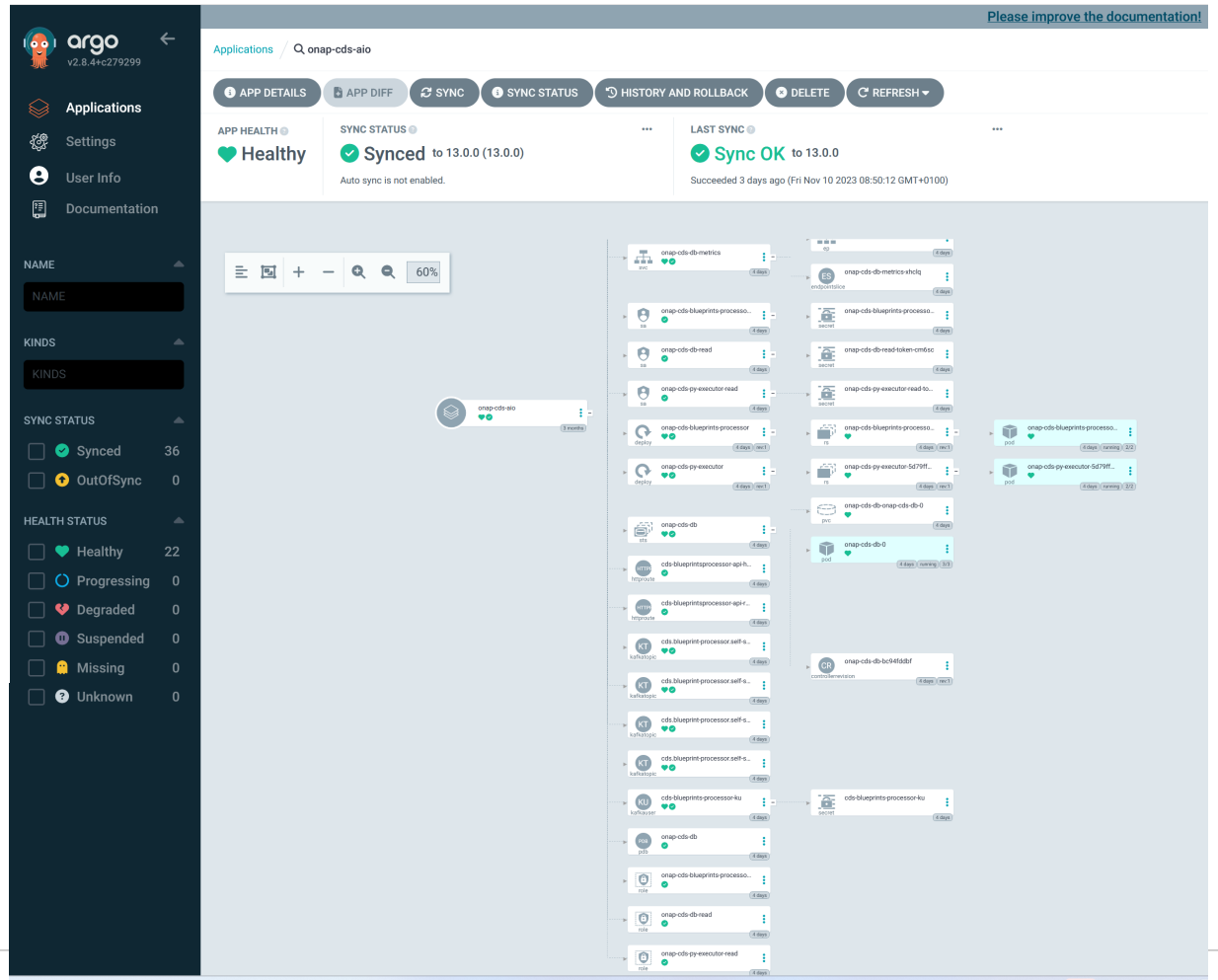


ArgoCD deployment

LFN Developer & Testing Forum

cds.yaml 1.22 KiB — Edit | Lock | Replace | Delete

```yaml
---
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: onap-cds
  namespace: argocd
  annotations:
    argocd.argoproj.io/sync-wave: "-70"
spec:
  project: argo-management
  syncPolicy:
    automated:
      prune: false
      selfHeal: true
  destination:
    server: https://kubernetes.default.svc
    namespace: onap
  source:
#    repoURL: https://artifactory.devops.telekom.de/artifactory/tnap-helmcharts/
#    chart: cds
#    targetRevision: 9.0.0
    repoURL: https://gitlab.devops.telekom.de/tnap/operations/infra/helmcharts/application-charts/onap-oom.git
    targetRevision: v11.0.0
    path: cds
    helm:
      parameters:
#       following properties added in order to bootstrap one-noded Galera cluster from Node number 0. Without this setting Galera will not start after POD resta
        - name: "mariadb-galera.galera.bootstrap.forceSafeToBootstrap"
          value: "true"
        - name: "mariadb-galera.galera.bootstrap.bootstrapFromNode"
          value: "0"
        - name: "global.cpsdata.password"
          value: "tj61KoH9"
        - name: "cds-blueprints-processor.workflow.storeEnabled"
          value: "false"
      valueFiles:
      - ../onap/values-global.yaml
      - ../onap/values-cds.yaml
```



Q & A

LFN NETWORKING
LFN Developer & Testing Forum

Thank you !