LF NETWORKING

LFN Developer & Testing Forum

ONAP: DAY-1 Config Management for O-RAN components

Lena Peuker (Deutsche Telekom AG)

14.06.2022

Recording

Anti-Trust Policy Notice

- Linux Foundation meetings involve participation by industry competitors, and it is the intention of the Linux Foundation to conduct all of its activities in accordance with applicable antitrust and competition laws. It is therefore extremely important that attendees adhere to meeting agendas, and be aware of, and not participate in, any activities that are prohibited under applicable US state, federal or foreign antitrust and competition laws.

- Examples of types of actions that are prohibited at Linux Foundation meetings and in connection with Linux Foundation activities are described in the Linux Foundation Antitrust Policy available at http://www.linuxfoundation.org/antitrustpolicy. If you have questions about these matters, please contact your company counsel, or if you are a member of the Linux Foundation, feel free to contact Andrew Updegrove of the firm of Gesmer Updegrove LLP, which provides legal counsel to the Linux Foundation.

Agenda

- Intro
- General overview
- Phase 1: Planning
- Phase 2: Creation
- Phase 3: Applying
- Summary
- Outlook
- Q&A

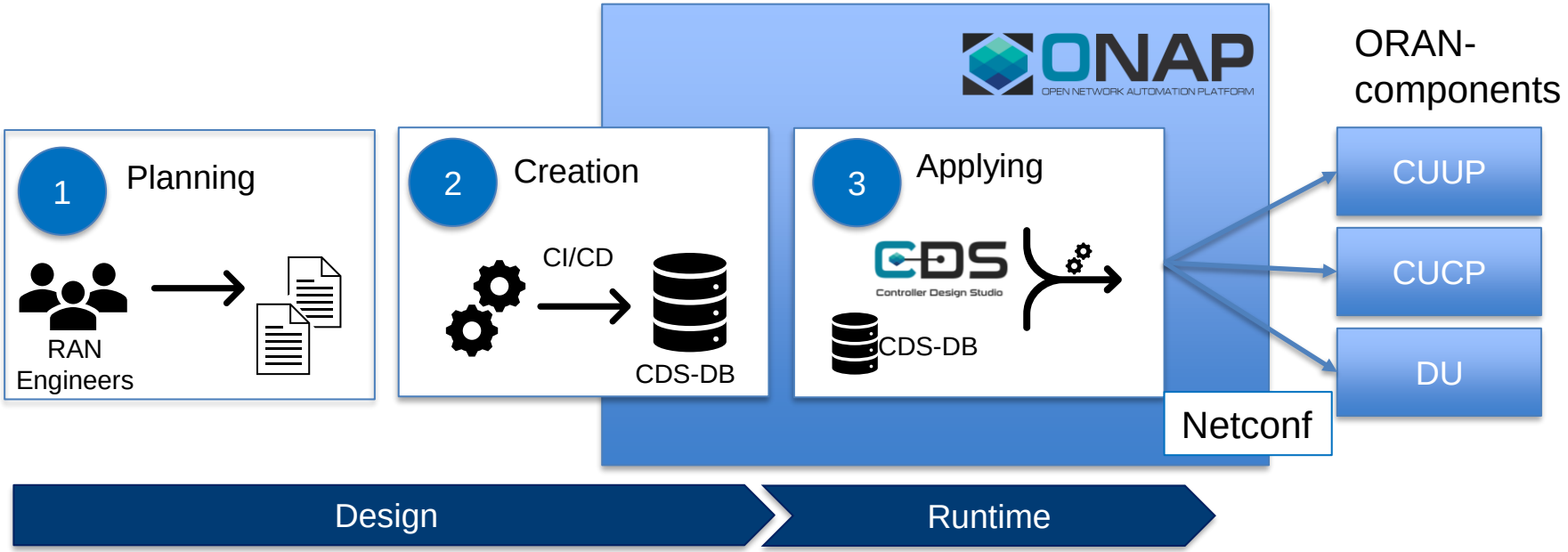
Intro

- Deutsche Telekom (DT) pursues ORAN with multiple cell sites and an integration with ONAP

- How do we want to handle the different configuration (DAY-1) for multiple cell sites with multiple network elements inside ONAP?

CDS-approach with Template-Engine / CICD

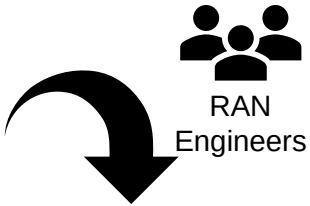
General overview



Phase 1: Planning

Golden parameters

- site_name
- gNBId
- id_cucp
- ….



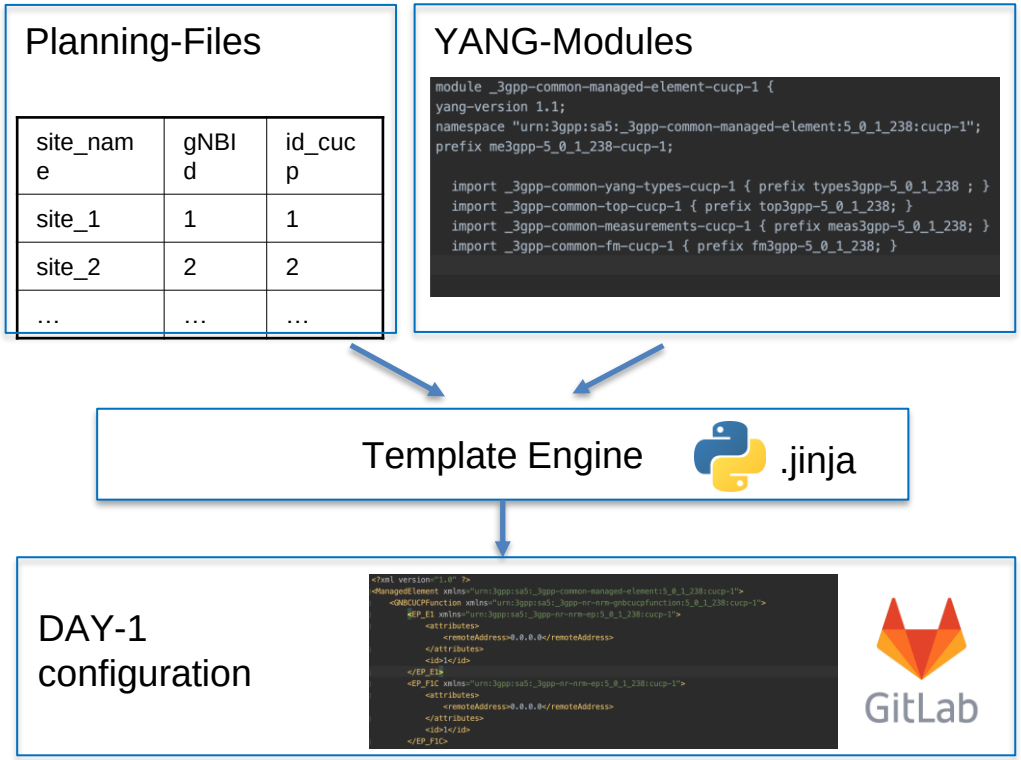
GitLab

.csv

site_name	gNBId	id_cucp
site_1	1	1
site_2	2	2
…	…	…

1. List of golden parameter that is received from the vendor based on supported YANG-models.
2. Ran Engineers / Ran Planning department are filling these parameters for the specific cell-sites.
3. Parameter-Sets are provided as .csv-Files from the planning department in GitLab.

Phase 2: Creation



1. Parameter-Sets are provided as .csv-Files from the planning department in GitLab.
2. Parameter-Sets (.csv) and YANG-models are inputs for a Template-Engine.
3. Template-Engine creates the specific configuration (DAY-1).
4. Created configurations for specific network elements are stored in GitLab.

Phase 2: Creation

DAY-1 configuration

GitLab

stored in CDS-DB via specific workflow-step

CDS
Controller Design Studio

resolution_key	artifact_name
cucp-1_3gpp.xml_1.0	day-1
cucp-1_3gpp.xml_1.0.2	day-1
cucp-1_vendor.xml_1.0	day-1
cucp-1_vendor.xml_1.0.2	day-1
cucp-1000_3gpp.xml_1.0	day-1
cucp-1000_vendor.xml_1.0	day-1
cucp-1001_3gpp.xml_1.0	day-1
cucp-1001_3gpp.xml_1.0	day-1
cucp-1001_vendor.xml_1.0	day-1
cucp-1001_vendor.xml_1.0	day-1

1. Parameter-Sets are provided as .csv-Files from the planning department in GitLab.
2. Parameter-Sets (.csv) and YANG-models are inputs for a Template-Engine.
3. Template-Engine creates the specific configuration (DAY-1).
4. Created configurations for specific network elements are stored in GitLab.
5. Created configurations for the specific network elements in a specific version are stored inside the CDS-DB to be later used at runtime.

Phase 2: Creation

LFN Developer & Testing Forum

Storing of configurations inside CDS via workflow-steps

```
"store-config-day-1" : {
  "steps" : {
    "activate-process" : {
      "description" : "Store Day 1 Config",
      "target" : "store-config-day-1",
      "activities" : [ {
        "call_operation" : ""
      } ]
    }
  },
  "inputs" : {
    "resolution-key" : {
      "required" : true,
      "type" : "string"
    },
    "store-result" : {
      "required" : true,
      "type" : "boolean"
    },
    "config-file" : {
      "required" : true,
      "type" : "string"
    },
    "store-config-day-1-properties" : {
      "description" : "Dynamic PropertyDefinition for workflow(store-config-day-1).",
      "required" : true,
      "type" : "dt-store-config-day-1-properties"
    }
  }
},
```

```
{
    "actionIdentifiers": {
        "mode": "sync",
        "blueprintName": "CBA_CUCP_CNF",
        "blueprintVersion": "1.0.0",
        "actionName": "store-config-day-1"
    },
    "payload": {
        "store-config-day-1-request": {
            "resolution-key": "cucp-1_3gpp.xml_1.0",
            "store-config-day-1-properties": {
                "resolution-key": "cucp-1_3gpp.xml_1.0",
                "store-result": true,
                "config-file": "<config>"
            }
        }
    },
    "commonHeader": {
        "subRequestId": "143748f9-3cd5-4910-81c9-a4601ff2ea58",
        "requestId": "e5eb1f1e-3386-435d-b290-d49d8af8db4c",
        "originatorId": "SDNC_DG"
    }
}
```

Phase 3: Applying

1

Service-instantiation

Instance Name: cucp-1

Instance Parameters: day0_version:1.0,day1_version:1.0

2

AAI


```
"service-instance-id": "ed7214b1-d7ac-4b5e-8596-204e06c3ba9f",
"service-instance-name": "cucp-1",
"service-type": "cucp",
"service-role": "cucp",
"environment-context": "General_Revenue-Bearing",
"workload-context": "1.0,1.0",
"model-invariant-id": "c6e0cc5c-7471-4e00-8412-da79e8d309e2",
"model-version-id": "6f0fefae-c94f-4dae-a86d-03a63406664e",
"resource-version": "1651035723935",
```


Service-Instantiation of an ORAN-component

1. At instantiation a specific config-version is passed as input-parameter.
2. This config-version is stored inside AAI during instantiation process.

Phase 3: Applying

3 CBA-Workflowstep

```
"actionIdentifiers": {
    "mode": "sync",
    "blueprintName": "CBA_CUCP_CNF",
    "blueprintVersion": "1.0.0",
    "actionName": "config-deploy-day-1"
},
"payload": {
    "config-deploy-day-1-request": {
        "nfId": "cucp-1",
        "version": "1.0"
    }
},
"commonHeader": {
    "subRequestId": "143748f9-3cd5-4910-81c9-a4601ff2ea58",
    "requestId": "e5eb1f1e-3386-435d-b290-d49d8af8db4c",
    "originatorId": "SDNC_DG"
}
```

4 Config-retrieval

```
suspend fun getConfigFromDB(resolution_key: String, artifact_name: String): String {
    try {
        val payload = storedContentFromResolvedArtifactNa(resolution_key, artifact_name)
        return config
    } catch (e: EmptyResultDataAccessException) {
        val blueprintName = bluePrintRuntimeService.bluePrintContext().name()
        val blueprintVersion = bluePrintRuntimeService.bluePrintContext().version()
        throw BluePrintProcessorException("Artifact (resolution_key: $resolution_key, artifact_name:
    }
}
```

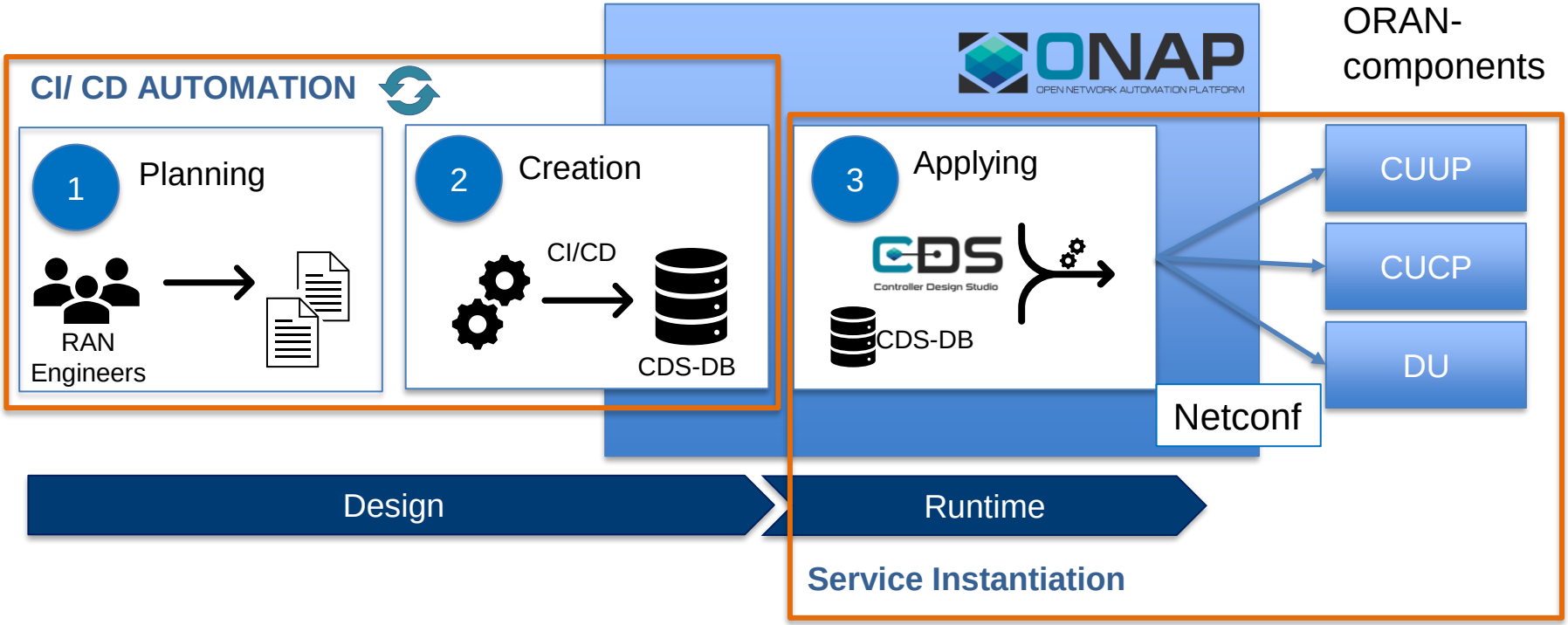
cucp-1_3gpp.xml_1.0

day-1

Service-Instantiation of a ORAN-component – Applying configuration via CDS / CBA

3. Workflow-step to apply DAY-1 configuration is triggered automatically during instatiation-process.
4. Based on instance-name and config-version the DAY-1-configuration is retrieved from the CDS-DB.
5. Config is applied via netconf-executor inside CDS / CBA.

Summary

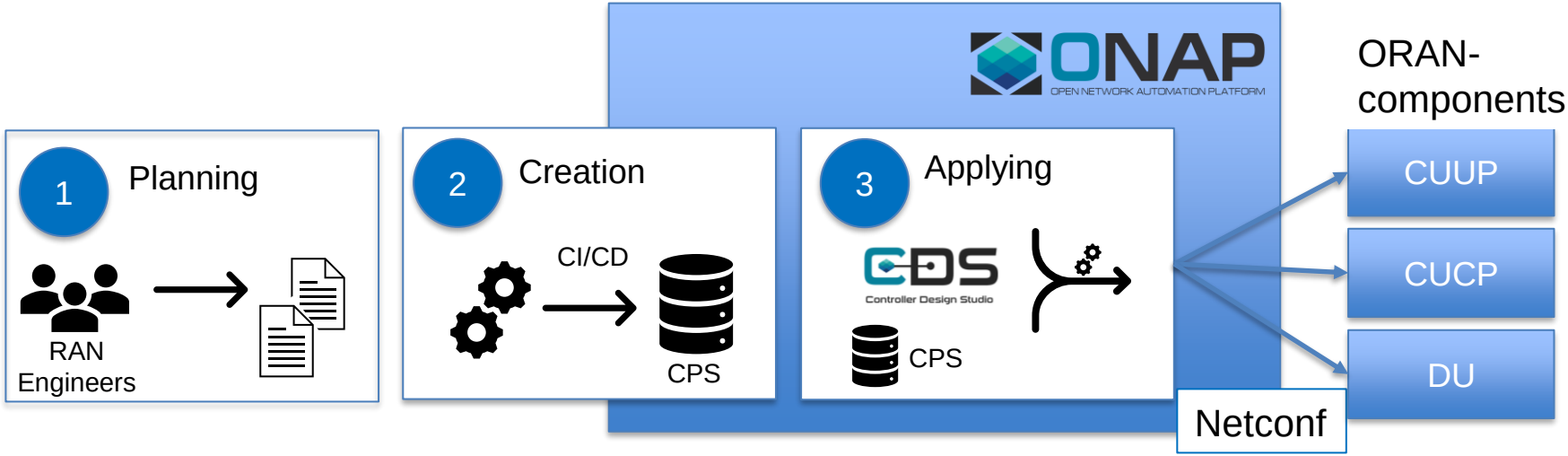

CI/ CD AUTOMATION
ONAP
OPEN NETWORK AUTOMATION PLATFORM
ORAN-components
1
Planning
RAN Engineers
2
Creation
CI/CD
CDS-DB
3
Applying
CDS
Controller Design Studio
CDS-DB
CUUP
CUCP
DU
Netconf
Design
Runtime
Service Instantiation

Summary / Outlook

CDS-approach with Template-Engine / CICD

- Existing mechanisms inside CDS to store configuration inside CDS
- Actual investigations to use CPS instead of CDS-DB to store config artifacts

Outlook



- Onboarding of YANG-Modules for ORAN-components inside CPS
- Automated storing of configs inside CPS
- CDS would retrieve the configs from CPS instead of the CDS-DB

→ gives us the possibility to handle planned and actual configuration

LF NETWORKING

LFN Developer & Testing Forum