LF NETWORKING
Developer & Testing Forum

ONAP: Lessons Learned

Ahmad Khalil (presenting), Ali Fouladgar
Mohammad Nurujjaman

https://lfnetworking.org

Contents

- Our Journey
- Baseline Use Case
  - Artifacts Qualification in New Releases: Lessons Learned
- Scaling & Prototyping IP-based Service Orchestration
  - Issues we face during new Prototype Use Case: Lessons Learned
- Platform Customization – Lessons Learned

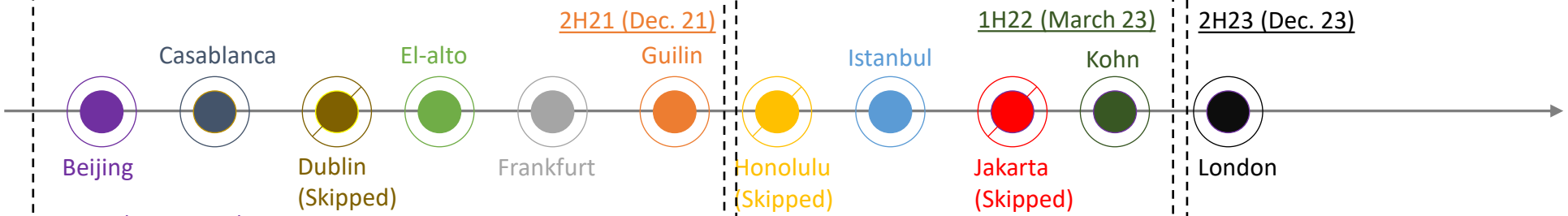
Our Journey

LFN Developer & Testing Forum

1. Evaluation & LAB POC Phase
2. MEF-based standard L2 E-Line service orchestration
3. Completed Evaluation December 2021 using Guilin (R7).

1. Post Evaluation & Production-Grade Phase (LAB)
2. Additional use case prototyping
3. Enhanced security & observability
4. Internal demos, and initiation of commercial deployment

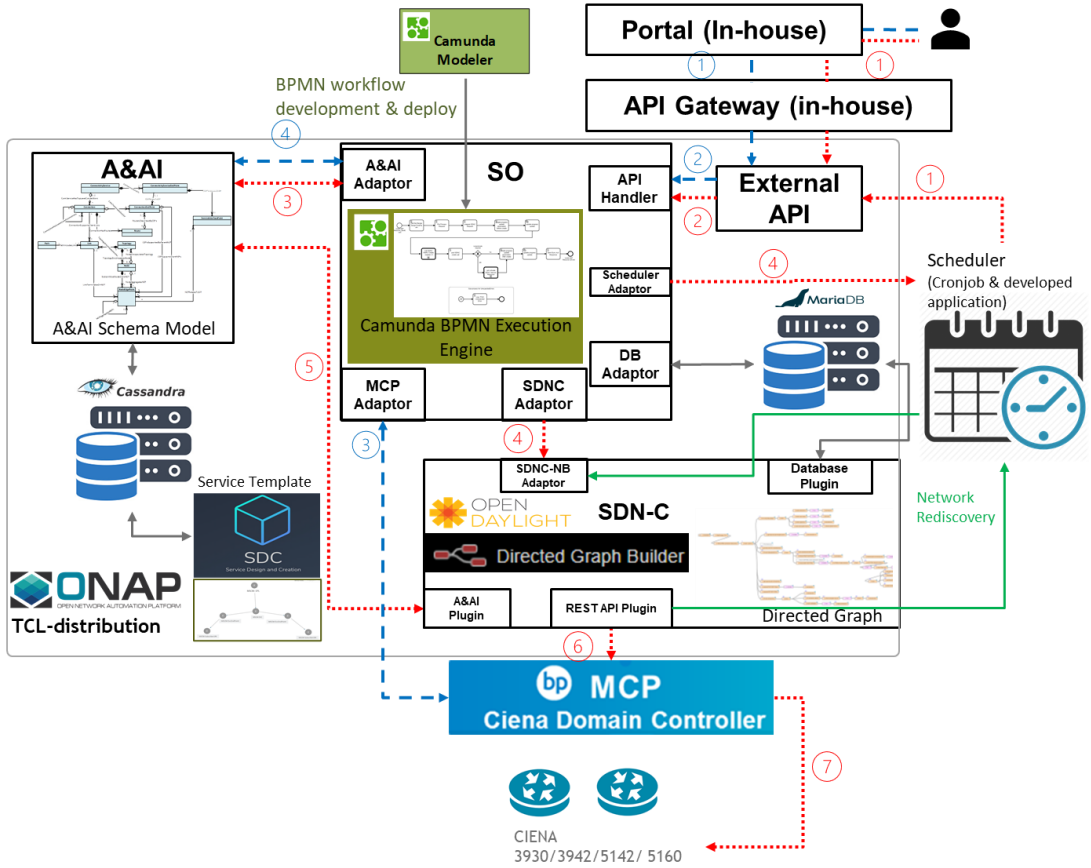
1. Commercial Deployment Phase (Production Environment)



1H19 (June 2018) – Initial non-production-grade deployment

Baseline Use Case

- Baseline use case: Demonstrate ONAP has the capabilities and flexibilities to discover and synchronize a physical Ethernet network and remotely orchestrate network Ethernet services over their lifecycles.
- Enabling BOD-Based Feasibility Check and Service Provisioning including Time-Aware Calendarization for E-Line
- Demo Test Case #1: Scheduling a BoD (Customer Experience)
  - Feasibility Check to upgrade a provisioned service (i.e., BoD)
  - Feasibility Rejection/Approval by user
  - Service Provisioning for a BoD
  - Disconnection of a provisioned BoD on its end time
- Demo Test Case #2: Network Monitoring & Troubleshooting (Tata Comm. Operation Experience)
  - Enhanced Real-time Monitoring
  - Enhanced Logging system
  - Performance monitoring: infrastructure & ONAP



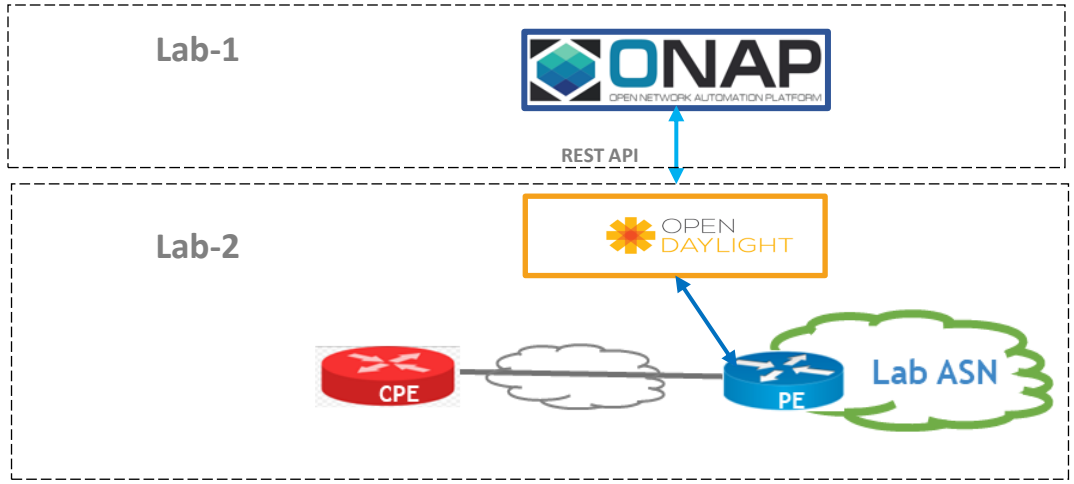
Artifacts Qualification in New Releases: Lessons Learned

- We developed artifacts for a use case prototype in earlier ONAP release.
- As new ONAP platform is released, we used to take backup from older release and restore that in newer release.
- Meanwhile in newer version of SDC, some parameters that we used in our artifacts become obsolete, category types changed.
- As a result, the service templates (YAML files) onboarding using SDC UI fails in newer ONAP releases even though the same service templates can be made available for instantiation while using backup/restoration process.
- [Lesson-learned] we adjusted our qualification process. Instead of using backup/restoration process, we onboard each service templates using SDC UI and make changes to the service templates, if necessary. This ensures that the artifacts are always up-to-date and compatible with latest ONAP release.

Scaling & Prototyping IP-based Service Orchestration

LFN Developer & Testing Forum

- **OBJECTIVE**: To demonstrate ONAP flexibilities to scale beyond our initial baseline use case.
- We planned to discover/synchronize an underlay physical network and orchestrate lifecycle of IP-based services.
- We used Open Daylight as IP domain controller
- Artifacts are developed in ONAP platform to support IP use case and integrated with ODL for service configuration



- Test 1: Network Discovery –
  - 1.a – show in inventory before & after adding domain controller
  - 1.b – show inventory after adding new device to domain controller
- Test 2: Provision IP standard ILL service –
  - 2.a – verification: Perform manual ping test from CPE to PE OR any public Internet server
  - 2.b – show inventory
- Test 3: Terminate IP Standard ILL service
  - 3.a – verification: Perform manual ping test from CPE to PE OR any public Internet server
  - 3.b – show inventory

Issues we face during new Prototype Use Case: Lessons Learned

- Service template and artifacts distribution using SDC
  - Artifacts distribution using SDC does not work.
  - Even the community provided use case artifacts distribution most often fails.
  - Community documentations do not provide enough explanation & guidelines.

- Authentication for NBI APIs
  - No authentication mechanism available for making API calls through NBI(external-api)

- Delete operation through NBI (External-API)
  - Deleting a service instance using external APIs wouldn’t work. Seen in London release.
  - Since external-API is no longer an active project, bug reporting through community was not a choice.

- AAI client library in SO
  - AAI Client provided by the library in SO wouldn’t work in London release.
  - Most likely because the AAI north-bound APIs support HTTP protocol only and the client implementation uses HTTPS protocol

Platform Customization: Lessons Learned (1)

- High availability at ONAP component level is not considered and hard to achieve due to lack of proper application-level HA mechanism and it is mostly relying on Kubernetes capabilities. Most of ONAP components have single point of failure in their implementation. For example, the common mariadb database (even when it is deployed in a cluster of 3 instances), has a single of failure (if bootstrap fails) which will bring down the database service.

- Backup/restoration solution (based on Velero) provided by ONAP community does not work due to lack of proper integration with Kubernetes objects and NFS solution. There are no backup/restoration solution provided for partial ONAP failure recovery

- Offline installation solution in community has not been maintained properly for last few releases and it is very hard to implement and make it work.

- Due to lack of support from community to resolve the issues offline installation and no particular mechanism to add new components or migrate customized components, it makes the provided offline installer solution practically unreliable

- Lack of a clear minor/major upgrade path from one release to new one. This is applicable from infrastructure level all the way to ONAP components and sub-components.

Platform Customization: Lessons Learned (2)

- Latest ONAP releases are significantly lagging on implementing the latest security patches and mitigations provided by open-source communities. This is applicable to testing and verification of specific versions of Docker, Kubernetes, Helm, etc…

- Replacing all the default hardcoded credentials is practically impossible. Also, there is no standard, policy or best practices in place within community to enforce each ONAP project development to implement a universal approach to be able to add/edit credentials in a practical way

- Lack of a proper structure and consistency on role, permissions, policies, grouping, etc., for both inter- and intra-component communications. A typical commercial deployment would require robust role-based authentication system such as creating multiple users, assigning various roles to different users, allowing sub-set of use case operations for certain group of users while blocking others, etc… These features are not readily available for use case specific operations

- Lack of a standard way to practice and enforce role-based access control (RBAC) for new or custom components (like adding monitoring tools Kibana, Grafana and etc.)

- There is no solution for securing ONAP’s Kubernetes infrastructure like k8s role-Based Access Control (RBAC) and defining network policies to isolate workloads as well as security requirements for OS-level hardening such as setting proper firewall rules and listing all the necessary TCP/UDP ports needed for the platform to work

LF NETWORKING

LFN Developer & Testing Forum

Thank You - Questions?