Overview

<table>
<thead>
<tr>
<th>Project Name</th>
<th>Enter the name of the project</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target Release Name</td>
<td>Amsterdam</td>
</tr>
<tr>
<td>Project Lifecycle State</td>
<td>Incubation</td>
</tr>
<tr>
<td>Participating Company</td>
<td>Bell, Amdocs, AT&amp;T, ZTE, Huawei</td>
</tr>
</tbody>
</table>

Scope

What is this release trying to address?

ONAP Operations Manager will be used to efficiently Deploy, Manage, Operate the ONAP platform and its components (e.g. MSO, DCAE, SDC, etc.) and infrastructure (VMs, Containers). The OOM addresses the current lack of consistent platform-wide method in managing software components, their health, resiliency and other lifecycle management functions. With OOM, service providers will have a single dashboard/UI to deploy & un-deploy the entire (or partial) ONAP platform, view the different instances being managed and the state of each component, monitor actions that have been taken as part of a control loop (e.g., scale in-out, self-heal), and trigger other control actions like capacity augments across data centers.

In the Amsterdam release, OOM will be used to automatically deploy and re-deploy all ONAP platform components using containers technology, to monitor components state and to automatically heal broken platform components when required.

Use Cases
OOM functionality is platform wide and is therefore not directly tied to specific use cases. However the vFW use case will be used as the test bed for OOM.

**Minimum Viable Product**

- **ONAP platform 1-click deployment on kubernetes for key components:**
  - onap-aai
    - aai-service
    - hbase
    - model-loader-service
  - onap-appc
    - dbhost
    - dgbuilder
    - sdncldb01
    - sdncldb02
    - sdnhost
  - onap-message-router
    - dmaap
    - global-kafka
    - zookeeper
  - onap-ms0
    - mariadb
    - mso
  - onap-policy
    - brmsgw
    - drools
    - mariadb
    - nexus
    - pap
    - pdp
    - pypdp
    - portalapps
    - portaldb
    - vnc-portal
  - onap-robot
    - robot
  - onap-sdc
    - sdc-be
    - sdc-cs
    - sdc-es
    - sdc-fe
    - sdc-kb
  - onap-sdnc
    - dbhost
    - sdnc-dgbuilder
    - sdnc-portal
    - sdncldb01
    - sdncldb02
    - sdnhost
  - onap-vid
    - vid-mariadb
    - vid-server
  - AAF
  - DCAE
  - MSB
  - Multi-VIM
  - VFC
- **vFirewall demo on docker/kubernetes**
- **OOM Guidelines:**
  - Provide guidelines to the community teams for bringing components to oom/kubernetes (all component teams can bring themselves their components to OOM).
- **Platform containers monitoring:** provide visibility to the state of each components
- **Platform components auto-healing mvp:** auto-restart of platform containers on shutdown (may require a second level of policies on top of kubernetes to avoid infinite restarts)
- **Platform Configuration management to deploy on different environments**
  - Technical architecture & technology POCs for tools like Ansible, Chef, Cloudify for config management
  - Platform configuration parameters externalization to enable custom deployments (i.e. remove all hardcoded parameters).
  - Critical parameters like openstack deployments and etc in an environment file.
  - Component-level configuration & version management: allow to easily change configurations and manage different versions of the configuration files for each component.
  - Platform configuration management examples/demos so that the community knows how to use oom. Would require 2 examples
    - 1 would be for lab deployment
    - 1 for a production deployment
- **Platform Upgradability:**
  - Teams must expose endpoints for rollbacks and ensure backward compability. We need to provide requirements to each component teams.
  - Ability to deploy multiple instances of ONAP on different versions.
  - Manage individual components versions - Ability to upgrade 1 component at the time.
Platform infrastructure & multi-deployment options support (TBD)
  - Orchestrating infrastructure deployment and more deployment options with Cloudify - will determine if it is MVP or stretch or M2 scope. Maybe we lay the foundation in M1 Amsterdam and then grow.

Stretch goal
  - ONAP platform 1-click deployment on kubernetes for additional components:
    - CLI (Stretch)
    - CLAMP (Stretch)
    - Holmes (Stretch)
  - Platform Configuration management - Scaled deployment
    - Distributed platform deployment using config management to determine where to deploy each individual components across geographies (stretch)
    - Advanced Multi-data-center platform deployment policies, e.g. applying anti-affinity rules for different components.

Out of scope
  - Platform components/containers auto-scaling auto scaling

Functionalities
List the functionalities that this release is committing to deliver by providing a link to JIRA Epics and Stories. In the JIRA Priority field, specify the priority (either High, Medium, Low). The priority will be used in case de-scoping is required. Don't assign High priority to all functionalities.

Epics

<table>
<thead>
<tr>
<th>Key</th>
<th>Summary</th>
<th>T Created</th>
<th>Updated</th>
<th>Due</th>
<th>Assignee</th>
<th>Reporter</th>
<th>P</th>
<th>Status</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>OOM-1958</td>
<td>Upgrade k8s and helm for El Alto</td>
<td>Jul 02, 2019</td>
<td>Dec 09, 2020</td>
<td></td>
<td>Marco Platania</td>
<td>Mike Elliott</td>
<td></td>
<td>CLOSED</td>
<td>Done</td>
</tr>
<tr>
<td>OOM-1568</td>
<td>Define configuration overrides for target deployment environment</td>
<td>Dec 21, 2018</td>
<td>Feb 23, 2022</td>
<td></td>
<td>Mike Elliott</td>
<td>Dominic Lunanuova</td>
<td></td>
<td>CLOSED</td>
<td>Won't Do</td>
</tr>
<tr>
<td>OOM-2062</td>
<td>Centralized Registry for K8s Cloud Regions</td>
<td>Aug 21, 2019</td>
<td>Apr 28, 2021</td>
<td></td>
<td>Mike Elliott</td>
<td>Mike Elliott</td>
<td></td>
<td>CLOSED</td>
<td>Won't Do</td>
</tr>
<tr>
<td>OOM-1172</td>
<td>Align Software Versions for ONAP Release</td>
<td>Jun 21, 2018</td>
<td>Jul 02, 2019</td>
<td></td>
<td>Mike Elliott</td>
<td>Roger Maitland</td>
<td></td>
<td>CLOSED</td>
<td>Done</td>
</tr>
<tr>
<td>OOM-1878</td>
<td>Secure k8s Cluster Access</td>
<td>May 23, 2019</td>
<td>May 07, 2020</td>
<td></td>
<td>Krzysztof Opasiak</td>
<td>Mike Elliott</td>
<td></td>
<td>CLOSED</td>
<td>Done</td>
</tr>
<tr>
<td>OOM-1933</td>
<td>Secure Credentials</td>
<td>Jun 20, 2019</td>
<td>Sep 14, 2021</td>
<td></td>
<td>Krzysztof Opasiak</td>
<td>Mike Elliott</td>
<td></td>
<td>CLOSED</td>
<td>Done</td>
</tr>
<tr>
<td>OOM-1240</td>
<td>Helm Chart Ownership Transfer</td>
<td>Jun 26, 2018</td>
<td>Apr 20, 2020</td>
<td></td>
<td>Mike Elliott</td>
<td>Roger Maitland</td>
<td></td>
<td>CLOSED</td>
<td>Won't Do</td>
</tr>
<tr>
<td>OOM-1946</td>
<td>Reducing Operational Complexity</td>
<td>Jun 26, 2019</td>
<td>Feb 23, 2022</td>
<td></td>
<td>Mike Elliott</td>
<td>Mike Elliott</td>
<td></td>
<td>CLOSED</td>
<td>Won't Do</td>
</tr>
<tr>
<td>OOM-1227</td>
<td>Enhance ONAP Storage Architecture</td>
<td>Jun 25, 2018</td>
<td>Apr 20, 2020</td>
<td></td>
<td>Sylvain Desbureaux</td>
<td>Roger Maitland</td>
<td></td>
<td>CLOSED</td>
<td>Done</td>
</tr>
<tr>
<td>OOM-1175</td>
<td>Shared Database Instances</td>
<td>Jun 25, 2018</td>
<td>Jun 09, 2022</td>
<td></td>
<td>Sylvain Desbureaux</td>
<td>Roger Maitland</td>
<td></td>
<td>IN PROGRESS</td>
<td>Unresolved</td>
</tr>
<tr>
<td>OOM-1944</td>
<td>Disaster Recovery for ONAP</td>
<td>Jun 25, 2019</td>
<td>Feb 23, 2022</td>
<td></td>
<td>Mike Elliott</td>
<td>Mike Elliott</td>
<td></td>
<td>CLOSED</td>
<td>Won't Do</td>
</tr>
<tr>
<td>OOM-1504</td>
<td>Health monitoring for ONAP Platform</td>
<td>Nov 02, 2018</td>
<td>Apr 20, 2020</td>
<td></td>
<td>Mike Elliott</td>
<td>Mike Elliott</td>
<td></td>
<td>CLOSED</td>
<td>Done</td>
</tr>
<tr>
<td>OOM-1237</td>
<td>Source Helm Charts from ONAP Repo</td>
<td>Jun 26, 2018</td>
<td>Nov 29, 2021</td>
<td></td>
<td>Sylvain Desbureaux</td>
<td>Roger Maitland</td>
<td></td>
<td>CLOSED</td>
<td>Done</td>
</tr>
<tr>
<td>OOM-1503</td>
<td>Platform Security (Network Policies)</td>
<td>Nov 01, 2018</td>
<td>Apr 20, 2020</td>
<td></td>
<td>Mike Elliott</td>
<td>Mike Elliott</td>
<td></td>
<td>CLOSED</td>
<td>Won't Do</td>
</tr>
<tr>
<td>OOM-1506</td>
<td>Platform support for multi-site application deployments</td>
<td>Nov 02, 2018</td>
<td>Nov 07, 2019</td>
<td></td>
<td>Mike Elliott</td>
<td>Mike Elliott</td>
<td></td>
<td>CLOSED</td>
<td>Done</td>
</tr>
</tbody>
</table>
Longer term roadmap

ONAP Platform OOM (ONAP Operations Manager) will be used to efficiently Deploy, Manage, Operate the ONAP platform and its components (e.g. MSO, DCAE, SDC, etc.) and infrastructure (VMs, Containers). The OOM addresses the current lack of consistent platform-wide method in managing software components, their health, resiliency and other lifecycle management functions. With OOM, service providers will have a single dashboard/UI to deploy & un-deploy the entire (or partial) ONAP platform, view the different instances being managed and the state of each component, monitor actions that have been taken as part of a control loop (e.g., scale in-out, self-heal), and trigger other control actions like capacity augments across data centers.

The primary benefits of this approach are as follows:

- **Flexible Platform Deployment** - While current ONAP deployment automation enables the entire ONAP to be created, more flexibility is needed to support the dynamic nature of getting ONAP instantiated, tested and operational. Specifically, we need the capability to repeatedly deploy, un-deploy, and make changes onto different environments (dev, system test, DevOps, production), for both platform as a whole or on an individual component basis. To this end, we are introducing the ONAP Operations Manager with orchestration capabilities into the deployment, un-deployment and change management process associated with the platform.

- **State Management of ONAP platform components** – Our initial health checking of Components and software modules are done manually and lack consistency. We are proposing key modules/services in each ONAP Component to be able to self-register/discovered into the ONAP Operations Manager, which in turn performs regular health checks and determines the state of the Components/software.

- **Platform Operations Orchestration / Control Loop Actions** – Currently there is a lack of event-triggered corrective actions defined for platform components. The ONAP Operations Manager will enable DevOps to view events and to manually trigger corrective actions. The actions might be simple initially – stop, start or restart the platform component. Over time, more advanced control loop automation, triggered by policy, will be built into the ONAP Operations Manager.
Proposed ONAP Operations Manager Functional Architecture:

- **UI/Dashboard** – this provides DevOps users a view of the inventory, events and state of what is being managed by the ONAP Operations Manager, and the ability to manually trigger corrective actions on a component. The users can also deploy ONAP instances, a component, or a change to a software module within a component.
- **API handler** – this supports NB API calls from external clients and from the UI/Dashboard
- **Inventory & data store** – tracks the inventory, events, health, and state of the ONAP instances and individual components
- **ONAP Lifecycle Manager** – this is a model-driven orchestration engine for deploying/un-deploying instances and components. It will trigger downstream plugin actions such as instantiate VMs, create containers, stop/restart actions, etc. Target implementation should aim at TOSCA as the master information model for deploying/managing ONAP Platform components.
- **SB Interface Layer** – these are a collection of plugins to support actions and interactions needed by the ONAP Operations Manager to ONAP instances and other external cloud related resources – plugins may include Openstack, Docker, Kubernetes, Chef, Ansible, etc.
- **Service & Configuration Registry** – this function performs the registry and discovery of components/software to be managed as well as the subsequent health check on each registered component/software
- **Hierarchical OOM architecture for scaling and specialization** – OOM’s architecture allows for a hierarchical implementation for scale as volume /load and scope of the underlying ONAP platform instance(s) increases.

Release Deliverables

Indicate the outcome (Executable, Source Code, Library, API description, Tool, Documentation, Release Note…) of this release.

<table>
<thead>
<tr>
<th>Deliverable Name</th>
<th>Deliverable Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OOM/ONAP Lifecycle Manager</td>
<td>ONAP platform 1-click deploy on K8S</td>
</tr>
<tr>
<td>OOM/ONAP Service &amp; Config Registry</td>
<td>ONAP Platform configuration management on K8S</td>
</tr>
<tr>
<td>OOM/Policy Enforcement</td>
<td>ONAP containers monitoring and auto-healing policy enforcement on K8S</td>
</tr>
<tr>
<td>OOM User guide</td>
<td>Provide guidelines to the community teams for bringing components to oom/kubernetes (all component teams can bring themselves their components to OOM)</td>
</tr>
</tbody>
</table>

Sub-Components

List all sub-components part of this release. Activities related to sub-component must be in sync with the overall release.

Sub-components are repositories are consolidate in a single centralized place. Edit the Release Components name for your project in the centralized page.
ONAP Dependencies

As OOM manages all of the ONAP components it has dependencies on all other projects. OOM's first goal is to manage all of the components from the 1.0 release and will incorporate newer components as they mature. In order for a project to be managed by OOM deployment artifacts that must be created - specifically a kubernetes deployment specification(s) and project configuration/environment data - in formats provided by this project.

Architecture

See ONAP Operations Manager (5/10/17) and OOM User Guide page.
See also detailed diagrams on the use of TOSCA Model Driven orchestration across container and non-container deployment options.

High level architecture diagram

OOM User Guide

API Incoming Dependencies

List the API this release is expecting from other releases.
Prior to Release Planning review, Team Leads must agreed on the date by which the API will be fully defined. The API Delivery date must not be later than the release API Freeze date.
Prior to the delivery date, it is a good practice to organize an API review with the API consumers.

<table>
<thead>
<tr>
<th>API Name</th>
<th>API Description</th>
<th>API Definition Date</th>
<th>API Delivery date</th>
<th>API Definition link (i.e. swagger)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSB Service Registry</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

API Outgoing Dependencies

API this release is delivering to other releases.

<table>
<thead>
<tr>
<th>API Name</th>
<th>API Description</th>
<th>API Definition Date</th>
<th>API Delivery date</th>
<th>API Definition link (i.e. swagger)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Helm</td>
<td>Helm API</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>KubeCTL</td>
<td>Kubernetes control</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Third Party Products Dependencies

Third Party Products mean products that are mandatory to provide services for your components. Development of new functionality in third party product may or not be expected.
List the Third Party Products (OpenStack, ODL, RabbitMQ, ElasticSearch,Crystal Reports, ...).

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kubernetes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rancher</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cloudify (Potential)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Consul</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Postgres</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In case there are specific dependencies (Centos 7 vs Ubuntu 16. Etc.) list them as well.
Testing and Integration Plans

Provide a description of the testing activities (unit test, functional test, automation,...) that will be performed by the team within the scope of this release.

Describe the plan to integrate and test the release deliverables within the overall ONAP system.

Confirm that resources have been allocated to perform such activities.

Gaps

This section is used to document a limitation on a functionality or platform support. We are currently aware of this limitation and it will be delivered in a future Release.

List identified release gaps (if any), and its impact.

<table>
<thead>
<tr>
<th>Gaps identified</th>
<th>Impact</th>
</tr>
</thead>
<tbody>
<tr>
<td>To fill out</td>
<td>To fill out</td>
</tr>
</tbody>
</table>

Known Defects and Issues

Provide a link toward the list of all known project bugs.

<table>
<thead>
<tr>
<th>Key</th>
<th>Summary</th>
<th>T Created</th>
<th>Updated</th>
<th>Due</th>
<th>Assignee</th>
<th>Reporter</th>
<th>P Status</th>
<th>Status</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>OOM-761</td>
<td>Move oomk8s artifacts to nexus3</td>
<td>Feb 22, 2018</td>
<td>Apr 11, 2022</td>
<td></td>
<td>Sylvain Desbureaux</td>
<td>Mike Elliott</td>
<td>CLOSED</td>
<td>Done</td>
<td></td>
</tr>
<tr>
<td>OOM-2155</td>
<td>Mariadb galera creates an unused PVC</td>
<td>Oct 28, 2019</td>
<td>Apr 11, 2022</td>
<td></td>
<td>Sylvain Desbureaux</td>
<td>Sylvain Desbureaux</td>
<td>CLOSED</td>
<td>Done</td>
<td></td>
</tr>
<tr>
<td>OOM-1577</td>
<td>Non-breaking UTF-8 space in K8 slave NFS script breaks NFS fstab mounting</td>
<td>Jan 02, 2019</td>
<td>Dec 09, 2020</td>
<td></td>
<td>Mike Elliott</td>
<td>John Lotoski</td>
<td>CLOSED</td>
<td>Done</td>
<td></td>
</tr>
<tr>
<td>OOM-1817</td>
<td>Use of global.repository inconsistent across Helm Charts</td>
<td>Apr 18, 2019</td>
<td>Apr 28, 2021</td>
<td></td>
<td>Akansha Dua</td>
<td>Mike Elliott</td>
<td>CLOSED</td>
<td>Done</td>
<td></td>
</tr>
<tr>
<td>OOM-1978</td>
<td>Misrendered enumerated lists</td>
<td>Jul 10, 2019</td>
<td>Dec 09, 2020</td>
<td></td>
<td>Pawe Wieczorek</td>
<td>Pawe Wieczorek</td>
<td>CLOSED</td>
<td>Done</td>
<td></td>
</tr>
<tr>
<td>OOM-2057</td>
<td>race condition: readiness-check pod ready.py script fails to detect shared Cassandra is ready</td>
<td>Aug 14, 2019</td>
<td>Dec 09, 2020</td>
<td></td>
<td>Mike Elliott</td>
<td>Keong Lim</td>
<td>CLOSED</td>
<td>Done</td>
<td></td>
</tr>
<tr>
<td>OOM-2114</td>
<td>helm deploy of sub-project fails and re-installs onap on new release name</td>
<td>Sep 18, 2019</td>
<td>Dec 09, 2020</td>
<td></td>
<td>Mike Elliott</td>
<td>Brian Freeman</td>
<td>CLOSED</td>
<td>Done</td>
<td></td>
</tr>
<tr>
<td>OOM-2121</td>
<td>Helm Deploy not working for Charts with Components</td>
<td>Oct 02, 2019</td>
<td>Dec 09, 2020</td>
<td></td>
<td>Mike Elliott</td>
<td>Mike Elliott</td>
<td>CLOSED</td>
<td>Not a Bug</td>
<td></td>
</tr>
<tr>
<td>OOM-2230</td>
<td>Missing requests/limits for some PODS</td>
<td>Nov 29, 2019</td>
<td>Dec 09, 2020</td>
<td></td>
<td>Sylvain Desbureaux</td>
<td>Sylvain Desbureaux</td>
<td>CLOSED</td>
<td>Done</td>
<td></td>
</tr>
<tr>
<td>OOM-2234</td>
<td>common postgres is not service mesh compliant</td>
<td>Dec 05, 2019</td>
<td>Dec 09, 2020</td>
<td></td>
<td>Sylvain Desbureaux</td>
<td>Sylvain Desbureaux</td>
<td>CLOSED</td>
<td>Done</td>
<td></td>
</tr>
<tr>
<td>OOM-2236</td>
<td>Two unit tests failing in dbcapci</td>
<td>Dec 06, 2019</td>
<td>Dec 09, 2020</td>
<td></td>
<td>Dominic Lunanuova</td>
<td>Dominic Lunanuova</td>
<td>CLOSED</td>
<td>Not a Bug</td>
<td></td>
</tr>
<tr>
<td>OOM-2245</td>
<td>Stumbling over kubernetes submodules</td>
<td>Dec 12, 2019</td>
<td>Dec 09, 2020</td>
<td></td>
<td>Dominic Lunanuova</td>
<td>Dominic Lunanuova</td>
<td>CLOSED</td>
<td>Done</td>
<td></td>
</tr>
<tr>
<td>OOM-2311</td>
<td>SSH Tunneling Failed on worker node</td>
<td>Feb 25, 2020</td>
<td>May 20, 2020</td>
<td>Feb 28, 2020</td>
<td>Himesh Shukla</td>
<td>Himesh Shukla</td>
<td>CLOSED</td>
<td>Done</td>
<td></td>
</tr>
</tbody>
</table>
Risks

List the risks identified for this release along with the plan to prevent the risk to occur (mitigation) and the plan of action in the case the risk would materialized (contingency).

<table>
<thead>
<tr>
<th>Risk identified</th>
<th>Mitigation Plan</th>
<th>Contingency Plan</th>
</tr>
</thead>
<tbody>
<tr>
<td>To fill out</td>
<td>To fill out</td>
<td>To fill out</td>
</tr>
</tbody>
</table>

Resources

Fill out the Resources Committed to the Release centralized page.

Release Milestone

The milestones are defined at the Release Level and all the supporting project agreed to comply with these dates.

Team Internal Milestone

This section is optional and may be used to document internal milestones within a project team or multiple project teams. For instance, in the case the team has made agreement with other team to deliver some artifacts on a certain date that are not in the release milestone, it is recommended to provide these agreements and dates in this section.

It is not expected to have a detailed project plan.

<table>
<thead>
<tr>
<th>Date</th>
<th>Project</th>
<th>Deliverable</th>
</tr>
</thead>
<tbody>
<tr>
<td>To fill out</td>
<td>To fill out</td>
<td>To fill out</td>
</tr>
</tbody>
</table>

Documentation, Training

- Highlight the team contributions to the specific document related to the project (Config guide, installation guide...).
- Highlight the team contributions to the overall Release Documentation and training asset.
- High level list of documentation, training and tutorials necessary to understand the release capabilities, configuration and operation.
- Documentation includes items such as:
  - Installation instructions
Other Information

Vendor Neutral

If this project is coming from an existing proprietary codebase, ensure that all proprietary trademarks, logos, product names, etc. have been removed. All ONAP deliverables must comply with this rule and be agnostic of any proprietary symbols.

Free and Open Source Software

FOSS activities are critical to the delivery of the whole ONAP initiative. The information may not be fully available at Release Planning, however to avoid late refactoring, it is critical to accomplish this task as early as possible.

List all third party Free and Open Source Software used within the release and provide License type (BSD, MIT, Apache, GNU GPL,...).

In the case non Apache License are found inform immediately the TSC and the Release Manager and document your reasoning on why you believe we can use a non Apache version 2 license.

Each project must edit its project table available at Project FOSS.

Charter Compliance

The project team comply with the ONAP Charter.