ONAP Operations Manager (5/10/17)

Project Name:

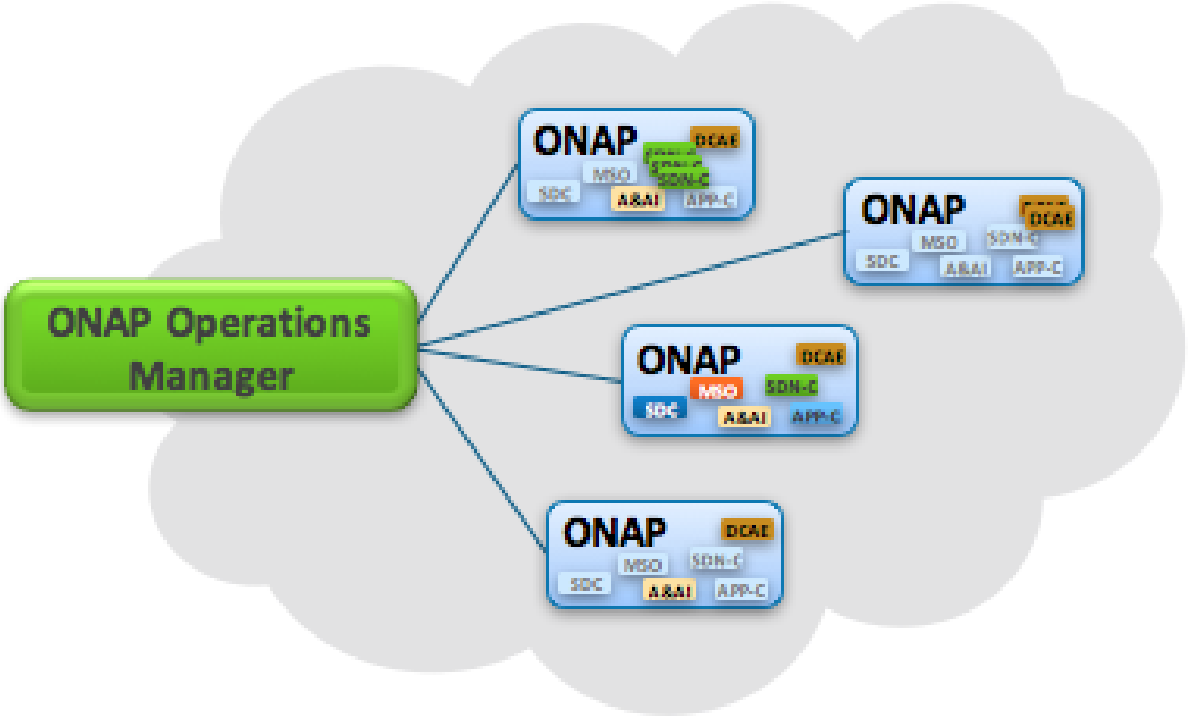
- Proposed name for the project: ONAP `Operations Manager`
- Proposed name for the repository: oom

Project description:

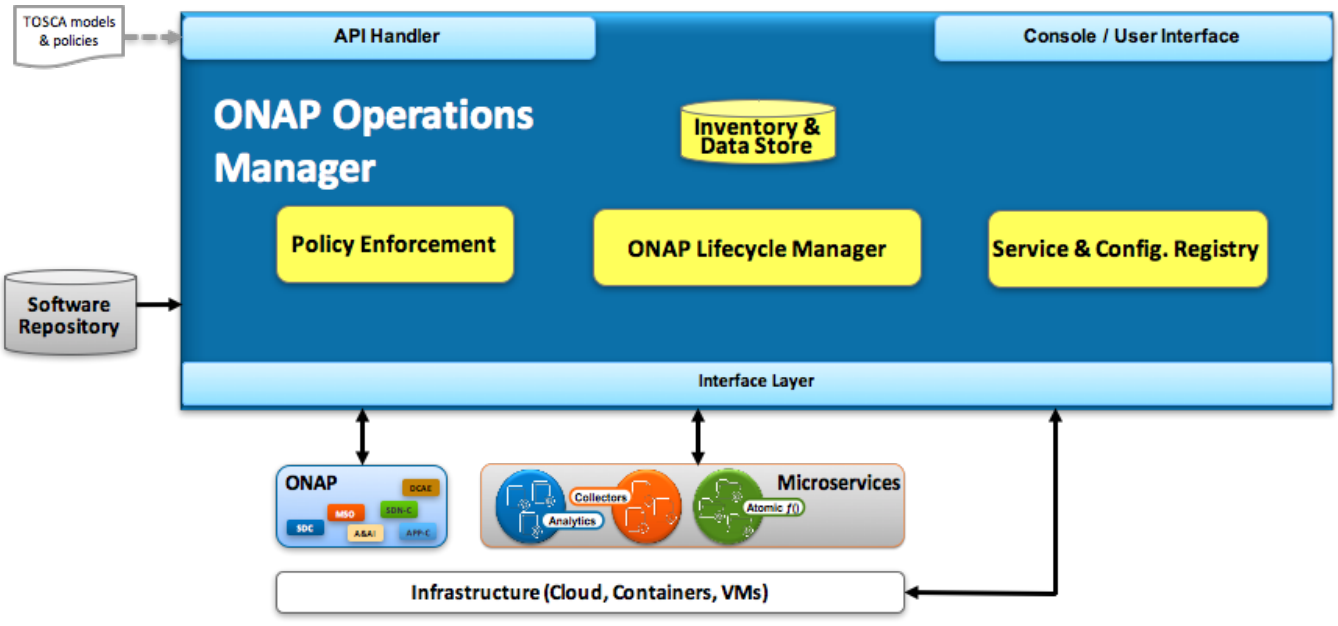
This proposal introduces the ONAP Platform OOM (ONAP Operations Manager) to efficiently Deploy, Manage, Operate the ONAP platform and its components (e.g. MSO, DCAE, SDC, etc.) and infrastructure (VMs, Containers). The OOM addresses the current lack of consistent platform-wide method in managing software 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. (see attached deck for more information on OOM's hierarchical architecture - Slides 7-8: https://wiki.onap.org/pages/worddav/preview.action?fileName=ONAP_Operations_Manager_Proposalv2.pdf&pageId=3246809.)

Scope:

- In scope: ONAP Platform lifecycle management & automation, i.e.
  - **Platform Deployment**: Automated deployment/un-deployment of ONAP instance(s) / Automated deployment/un-deployment of individual platform components
  - **Platform Monitoring & healing**: Monitor platform state, Platform health checks, fault tolerance and self-healing
  - **Platform Scaling**: Platform horizontal scalability
  - **Platform Upgrades**: Platform upgrades
  - **Platform Configurations**: Manage overall platform components configurations
  - **Platform migrations**: Manage migration of platform components
- Out of scope:

Architecture Alignment:

- How does this project fit into the rest of the ONAP Architecture?
  - The ONAP Operations Manager (OOM) is used to deploy, manage and automate ONAP platform components operations.

- How does this align with external standards/specifications?
  - At target, TOSCA should be used to model platform deployments and operations.
- Are there dependencies with other open source projects?
  - Open source technologies could be Cloudify, Kubernetes, Docker, and others.
  - The OOM will have dependency on the current proposed "Common Controller Framework" (CommServ 2) project and will leverage its software framework.
  - The OOM will require to work with the MSB project for to perform a comprehensive **Service Registry** function. At this point, the 2 projects seems to have needs to manage service registry at 2 different levels, i.e. MSB at the micro/macro service endpoint level vs OOM at the micro-service/component level.
  - The OOM will have inter-dependencies with DCAE (specifically the DCAE Controller). OOM will be built on the CCF software framework for ONAP platform management. DCAE Controller will be updated to use the same CCF software framework. In addition, OOM ("Root Node") and DCAE Controller ("DCAE Node") form the initial hierarchy of managers for ONAP platform management (as depicted in Slides 7-8 of this deck: https://wiki.onap.org/pages/worddav/preview.action?fileName=ONAP_Operations_Manager_Proposalv2.pdf&pageId=3246809). OOM Root Node is the Tier 1 manager responsible for the entire platform and delegates to the Tier 2 DCAE Controller/DCAE Node to be responsible for managing DCAE and its analytics and collector microservices.
  - The current proposed "System Integration and Testing" (Integration) Project might have a dependency on this project - use OOM to deploy/undeploy/change the test environments, including creation of the container layer.
  - This project has also a dependency on the LF infrastructure (seed code from ci-management project)

Initial Implementation Proposal: ONAP on Containers

Description:

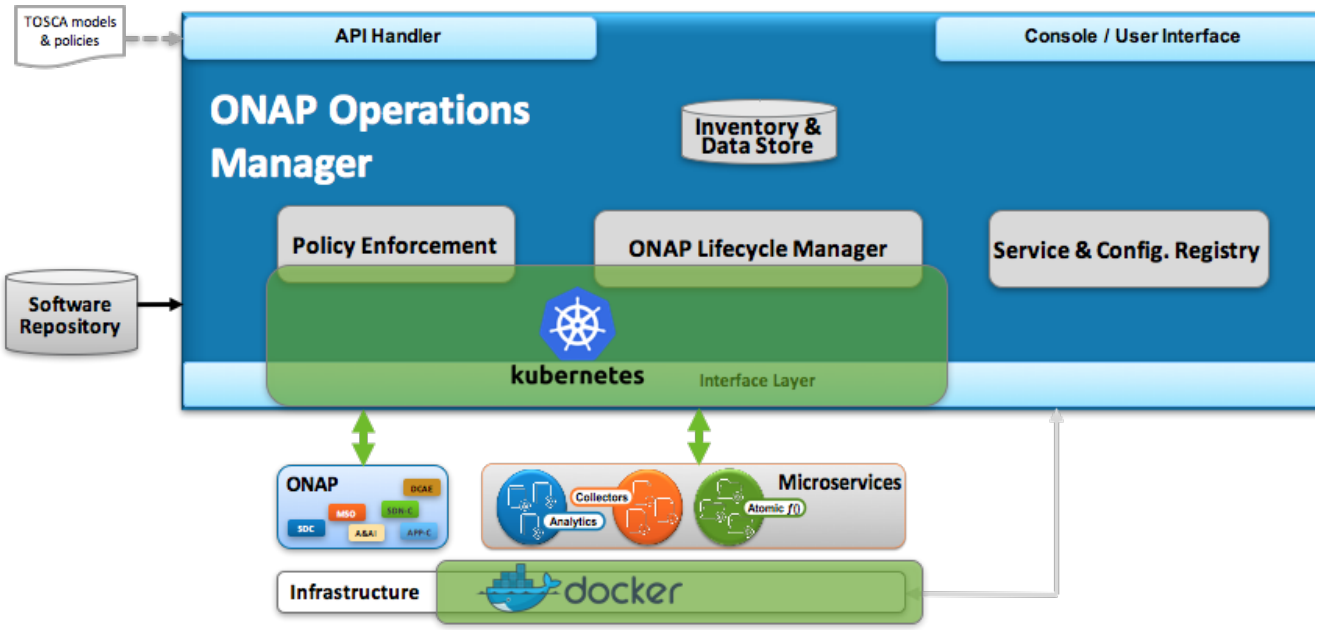
This milestone describes a deployment and orchestration option for the ONAP platform components (MSO, SDNC, DCAE, etc.) based on Docker containers and the open-source Kubernetes container management system. This solution removes the need for VMs to be deployed on the servers hosting ONAP components and allows Docker containers to directly run on the host operating system. As ONAP uses Docker containers presently, minimal changes to existing ONAP artifacts will be required.

- The primary benefits of this approach are as follows:
  - **Life-cycle Management**. Kubernetes is a comprehensive system for managing the life-cycle of containerized applications. Its use as a platform manager will ease the deployment of ONAP, provide fault tolerance and horizontal scalability, and enable seamless upgrades.
  - **Hardware Efficiency**. As opposed to VMs that require a guest operating system be deployed along with the application, containers provide similar application encapsulation with neither the computing, memory and storage overhead nor the associated long term support costs of those guest operating systems. An informal goal of the project is to be able to create a development deployment of ONAP that can be hosted on a laptop.
  - **Deployment Speed**. Eliminating the guest operating system results in containers coming into service much faster than a VM equivalent. This advantage can be particularly useful for ONAP where rapid reaction to inevitable failures will be critical in production environments.
  - **Cloud Provider Flexibility**. A Kubernetes deployment of ONAP enables hosting the platform on multiple hosted cloud solutions like Google Compute Engine, AWS EC2, Microsoft Azure, CenturyLink Cloud, IBM Bluemix and more.

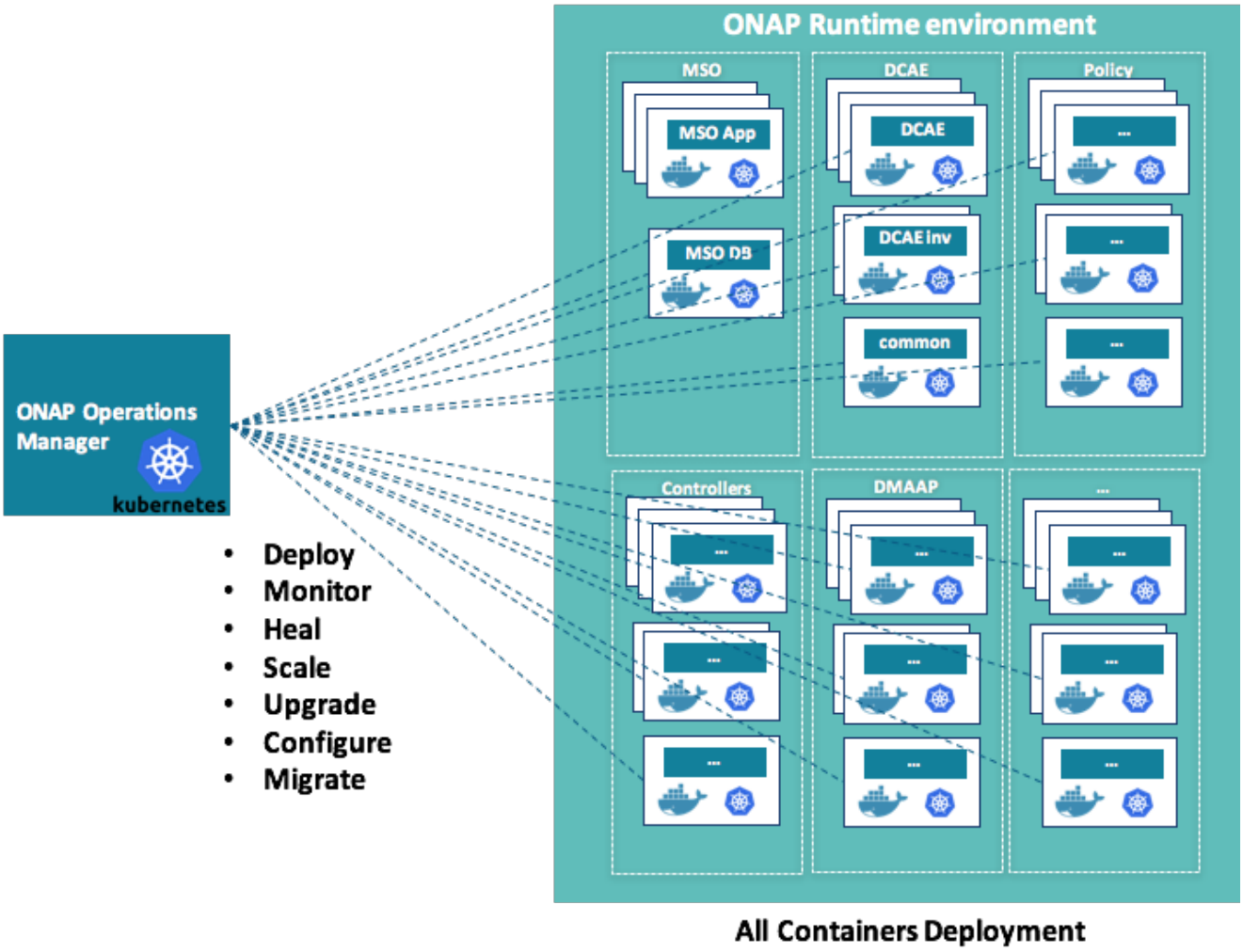
In no way does this project impair or undervalue the VM deployment methodology currently used in ONAP. Selection of an appropriate deployment solution is left to the ONAP user.

The ONAP on Containers project is part of the ONAP Operations Manager project and focuses on (as shown in green):

- Converting ONAP components deployment to docker containers
- Orchestrating ONAP components lifecycle using Kubernetes



As part of the OOM project, it will manage the lifecycle of individual containers on the ONAP runtime environment.



All Containers Deployment

Challenges

Although the current structure of ONAP lends itself to a container based manager there are challenges that need to be overcome to complete this project as follows:

- **Duplicate containers** – The VM structure of ONAP hides internal container structure from each of the components including the existence of duplicate containers such as Maria DB.
- **DCAE** - The DCAE component not only is not containerized but also includes its own VM orchestration system. A possible solution is to not use the DCAE Controller but port this controller's policies to Kubenetes directly, such as scaling CDAP nodes to match offered capacity.
- **Ports** - Flattening the containers also expose port conflicts between the containers which need to be resolved.
- **Permanent Volumes** - One or more permanent volumes need to be established to hold non-ephemeral configuration and state data.
- **Configuration Parameters** - Currently ONAP configuration parameters are stored in multiple files; a solution to coordinate these configuration parameters is required. Kubernetes Config Maps may provide a solution or at least partial solution to this problem.
- **Container Dependencies** – ONAP has built-in temporal dependencies between containers on startup. Supporting these dependencies will likely result in multiple Kubernetes deployment specifications.

Scope:

- In scope: ONAP Operations Manager implementation **using docker containers and kubernetes**, i.e.
  - **Platform Deployment**: Automated deployment/un-deployment of ONAP instance(s) / Automated deployment/un-deployment of individual platform components using docker containers & kubernetes
  - **Platform Monitoring & healing**: Monitor platform state, Platform health checks, fault tolerance and self-healing using docker containers & kubernetes
  - **Platform Scaling**: Platform horizontal scalability through using docker containers & kubernetes
  - **Platform Upgrades**: Platform upgrades using docker containers & kubernetes
  - **Platform Configurations**: Manage overall platform components configurations using docker containers & kubernetes
  - **Platform migrations**: Manage migration of platform components using docker containers & kubernetes
- Out of scope: support of container networking for VNFs. The project is about containerization of the ONAP platform itself.

Architecture Alignment:

- How does this project fit into the rest of the ONAP Architecture?
  - Please Include architecture diagram if possible
  - What other ONAP projects does this project depend on?
  - ONAP Operations Manager (OOM) [Formerly called ONAP Controller]: The ONAP on Containers project is a sub-project of OOM focusing on docker/kubernetes management of the ONAP platform components
  - The current proposed "System Integration and Testing" (Integration) Project might have a dependency on this project - use OOM to deploy/undeploy/change the test environments, including creation of the container layer.
  - This project has also a dependency on the LF infrastructure (seed code from ci-management project)
- How does this align with external standards/specifications?
  - N/A
- Are there dependencies with other open source projects?
  - Docker
  - Kubernetes

Resources:

- Primary Contact Person:
  - Mike Elliott (Amdocs)
- Committers and Contributors:
  - OOM Team

Other Information:

- link to seed code:
  - oom: https://gerrit.onap.org/r/gitweb?p=oom.git
- Vendor Neutral
  - if the proposal is coming from an existing proprietary codebase, have you ensured that all proprietary trademarks, logos, product names, etc., have been removed?
- Meets Board policy (including IPR)

Use the above information to create a key project facts section on your project page

Key Project Facts

Project Name:

- JIRA project name: ONAP Operations Manager
- JIRA project prefix: oom

Repo name: oom
Lifecycle State: Approved
Primary Contact: Mike Elliott (Amdocs)
Project Lead: Mike Elliott (Amdocs)
mailing list tag oom
Committers: OOM Team

*Link to TSC approval:
Link to approval of additional submitters: