

# Logging Architecture

- Artifacts
- High Level Description
- ELK stack in ONAP
- Use Cases
  - Use Case 1: SDC distribution
  - Use Case 2: Triage a failed Service Deployment
- Architecture Discussion
  - Log Generation
  - Log Collection
  - Log Processing
  - Log Analytics
- Design Issues
  - DI 1: 20170803: Cross Project Collaboration
    - AAI
  - DI 2: 20170929: Volumetrics
  - DI 3: 20171010: AAI Logging API Debug
- DevOps
  - OOM Deployment
- Northbound APIs
- Discussions
- Links

## Artifacts

[JIRAs](#) | [Git \(pending\)](#) | [Gerrit \(pending\)](#) | [Nexus](#) | [Jenkins \(pending\)](#) | [Sonar](#) |

## High Level Description

There are the 3 elk containers in the log pod in kubernetes off the OOM deployment. Currently all 3 ports are exposed, logstash, elasticsearch and kibana. We use Kubernetes 1.8 with Helm 2.8 charts via the OOM project deployment. The ports are all mapped as node ports in the external 302xx namespace. We use filebeat as a side car container for all the onap components to push logs to the logstash node port.

Logstash port mapping

<https://git.onap.org/oom/tree/kubernetes/log/charts/log-logstash/templates/service.yaml>

<https://git.onap.org/oom/tree/kubernetes/log/charts/log-logstash/templates/deployment.yaml>

port numbers are here - 5044

<https://git.onap.org/oom/tree/kubernetes/log/charts/log-logstash/values.yaml>

for example SDC filebeat config to log-ls on port 5044

<https://git.onap.org/oom/tree/kubernetes/sdc/values.yaml>

the filebeat sidecar docker image for sdc-be

<https://git.onap.org/oom/tree/kubernetes/sdc/charts/sdc-be/values.yaml>

Logging configmap

<https://git.onap.org/oom/tree/kubernetes/sdc/charts/sdc-be/templates/configmap.yaml>

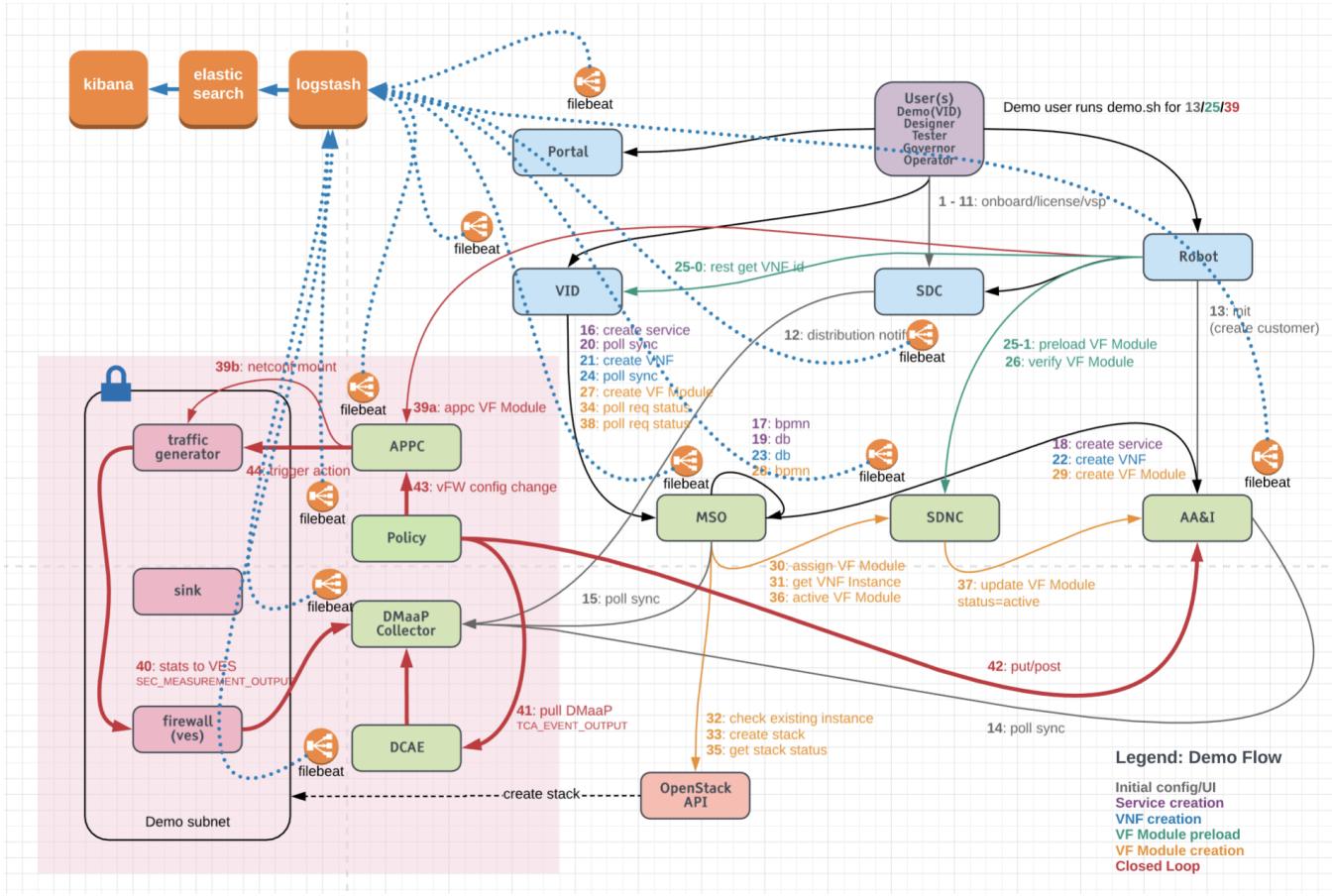
Logging filebeat deployment spec

<https://git.onap.org/oom/tree/kubernetes/sdc/charts/sdc-be/templates/deployment.yaml>

and the logback config

<https://git.onap.org/oom/tree/kubernetes/sdc/charts/sdc-be/resources/config/logging/logback.xml>

## ELK stack in ONAP



## Use Cases

### Use Case 1: SDC distribution

We can distribute manually via SDC or automatically using robot ./demo.sh distribute

The screenshot shows a web-based log analysis interface and a terminal window side-by-side.

**Log Analysis Interface:**

- Left Panel:** A sidebar with various log types: \_type, host, instance UUID, message, path, phys/virt server name, process Key, response description, service name, svc instance, tags, threadID, unused.
- Logs:**
  - September 14th 2017, 22:51:27.507:** Action: Create RequestID: - Server: INFO Partner Name: - Client IP: - path: /dockerdata-nfs/ona p/sdc/logs/SDC/SDC-BE/audit.log Begin TS: September 14th 2017, 22:51:27.507 svc instance: 95d73849-1f7d-4a37-9408-fd1d2a8595a1 @version: 1 host: ip-172-31-93-122 Log level: - Response code: cs0008 Server IP: - Severity: - message: 2017-09-15T02:51:27.507Z|||95d73849-1f7d-4a37-9408-fd1d2a8595a1|c a62c539-37f9-4352-8c85-9577e9147513|qtp1013423070-1393653|||SDC-BE|cs0008|||||INFO||10.42.15.233||10.4
  - September 14th 2017, 22:51:26.746:** RequestID: - Server: INFO Partner Name: Create VSP Client IP: - path: /dockerdata-nfs/onap/sdc/lo gs/SDC/SDC-BE/audit.log Begin TS: September 14th 2017, 22:51:26.746 svc instance: - @version: 1 host: ip-172-31-93-122 Log level: - Response code: - Server IP: - Severity: - message: 2017-09-15T02:51:26.746Z|||qtp1013423070-1395434|||Create VSP|SDC-BE|||||N/A|INFO|||10.42.15.233|||o.o.s.v.r.s.VendorSoftwareProductsImpl|||ActivityType=audit>, Desc=< --Audit-- Create VSP. VSP Name: 913abald-a2

**Terminal Window:**

```

onap — root@ip-172-31-93-122:/dockerdockerdata-nfs/onap/robot — ssh ubuntu@dev.onap.info — 138x32
root@vm1... ... root@zld... ... root@vm1... ... root@vmf... ... root@vm1... ... ubuntu@l... root@ip-1... root@ip-1... ubuntu@l... >> + 51:26.540
- Delete the module created by instantiateVFW
root@ip-172-31-93-122:/dockerdockerdata-nfs/onap/robot# ./demo-k8s.sh distribute
Starting Xvfb on display :89 with res 1280x1024x24
Executing robot tests at log level TRACE
=====
OpenCOMP ETE
=====
OpenCOMP ETE.Robot
=====
OpenCOMP ETE.Robot.Testsuites
=====
OpenCOMP ETE.Robot.Testsuites.Demo :: Executes the VNF Orchestration Test ...
=====
Initialize Models | PASS |
OpenCOMP ETE.Robot.Testsuites.Demo :: Executes the VNF Orchestrat... | PASS |
1 critical test, 1 passed, 0 failed
1 test total, 1 passed, 0 failed
=====
OpenCOMP ETE.Robot.Testsuites | PASS |
1 critical test, 1 passed, 0 failed
1 test total, 1 passed, 0 failed
=====
OpenCOMP ETE.Robot | PASS |
1 critical test, 1 passed, 0 failed
1 test total, 1 passed, 0 failed
=====
OpenCOMP ETE | PASS |
1 critical test, 1 passed, 0 failed
1 test total, 1 passed, 0 failed
=====
Output: /var/opt/OpenCOMP_ETE/html/logs/demo/InitDistribution/output.xml

```

## Use Case 2: Triaging a failed Service Deployment

## Architecture Discussion

Log Generation

Log Collection

Log Processing

Log Analytics

## Design Issues

DI 1: 20170803: Cross Project Collaboration

### AAI

<https://git.onap.org/aailogging-service/>

Audit and Metrics log formats are different

```

audit.log
return startTimeString + " | " + // 1 start time
endTimeString + " | " + // 2 end time
getMdcValue(MdcContext.MDC_REQUEST_ID) + " | " + // 3 transaction id
getMdcValue(MdcContext.MDC_SERVICE_INSTANCE_ID) + " | " + // 4 service instance
Thread.currentThread().getName() + " | " + // 5 thread id
getMdcValue(MdcContext.MDC_SERVER_FQDN) + " | " + // 6 physical/virtual server name
getMdcValue(MdcContext.MDC_SERVICE_NAME) + " | " + // 7 service name
getMdcValue(MdcContext.MDC_PARTNER_NAME) + " | " + // 8 partner name
fieldValue(DefinedFields.STATUS_CODE) + " | " + // 9 status code
fieldValue(DefinedFields.RESPONSE_CODE) + " | " + // 10 response code
fieldValue(DefinedFields.RESPONSE_DESCRIPTION) + " | " + // 11 response description
fieldValue(DefinedFields.INSTANCE_UUID) + " | " + // 12 instance UUID
level + " | " + // 13 log level
fieldValue(DefinedFields.SEVERITY) + " | " + // 14 log severity
fieldValue(DefinedFields.SERVER_IP) + " | " + // 15 server ip
elapsedTimeString + " | " + // 16 elapsed time
getMdcValue(MdcContext.MDC_SERVER_FQDN) + " | " + // 17 server name
getMdcValue(MdcContext.MDC_CLIENT_ADDRESS) + " | " + // 18 client ip address
fieldValue(DefinedFields.CLASS_NAME) + " | " + // 19 class name
" | " + // 20 deprecated
fieldValue(DefinedFields.PROCESS_KEY) + " | " + // 21 process key
fieldValue(DefinedFields.CUSTOM_1) + " | " + // 22 custom 1
fieldValue(DefinedFields.CUSTOM_2) + " | " + // 23 custom 2
fieldValue(DefinedFields.CUSTOM_3) + " | " + // 24 custom 3
fieldValue(DefinedFields.CUSTOM_4) + " | " + // 25 custom 4
message; // 26 details
}

metrics.log
return startTimeString + " | " + // 1 start time
endTimeString + " | " + // 2 end time
getMdcValue(MdcContext.MDC_REQUEST_ID) + " | " + // 3 transaction id
getMdcValue(MdcContext.MDC_SERVICE_INSTANCE_ID) + " | " + // 4 service instance
Thread.currentThread().getName() + " | " + // 5 thread id
getMdcValue(MdcContext.MDC_SERVER_FQDN) + " | " + // 6 physical/virtual server name
getMdcValue(MdcContext.MDC_SERVICE_NAME) + " | " + // 7 service name
getMdcValue(MdcContext.MDC_PARTNER_NAME) + " | " + // 8 partner name
fieldValue(DefinedFields.TARGET_ENTITY) + " | " + // 9 target entity
fieldValue(DefinedFields.TARGET_SVC_NAME) + " | " + // 10 target service
fieldValue(DefinedFields.STATUS_CODE) + " | " + // 11 status code
fieldValue(DefinedFields.RESPONSE_CODE) + " | " + // 12 response code
fieldValue(DefinedFields.RESPONSE_DESCRIPTION) + " | " + // 13 response description
fieldValue(DefinedFields.INSTANCE_UUID) + " | " + // 14 instance UUID
level + " | " + // 15 log level
fieldValue(DefinedFields.SEVERITY) + " | " + // 16 log severity
fieldValue(DefinedFields.SERVER_IP) + " | " + // 17 server ip
elapsedTimeString + " | " + // 18 elapsed time
getMdcValue(MdcContext.MDC_SERVER_FQDN) + " | " + // 19 server name
fieldValue(DefinedFields.CLIENT_IP) + " | " + // 20 client ip address
fieldValue(DefinedFields.CLASS_NAME) + " | " + // 21 class name
" | " + // 22 deprecated
fieldValue(DefinedFields.PROCESS_KEY) + " | " + // 23 process key
fieldValue(DefinedFields.TARGET_ENTITY) + " | " + // 24 target virtual entity
fieldValue(DefinedFields.CUSTOM_1) + " | " + // 25 custom 1
fieldValue(DefinedFields.CUSTOM_2) + " | " + // 26 custom 2
fieldValue(DefinedFields.CUSTOM_3) + " | " + // 27 custom 3
fieldValue(DefinedFields.CUSTOM_4) + " | " + // 28 custom 4
message; // 29 detail message
}

```

## DCAE

1.0.0 (deprecated) - <https://git.onap.org/dcae/operation/utils/>

## DMaaS

## DMaP API

<https://wiki.onap.org/download/attachments/11928249/DR-R1-Pub-Del-API-v1.5.pdf?version=2&modificationDate=1501879648000&api=v2>

## DI 2: 20170929: Volumetrics

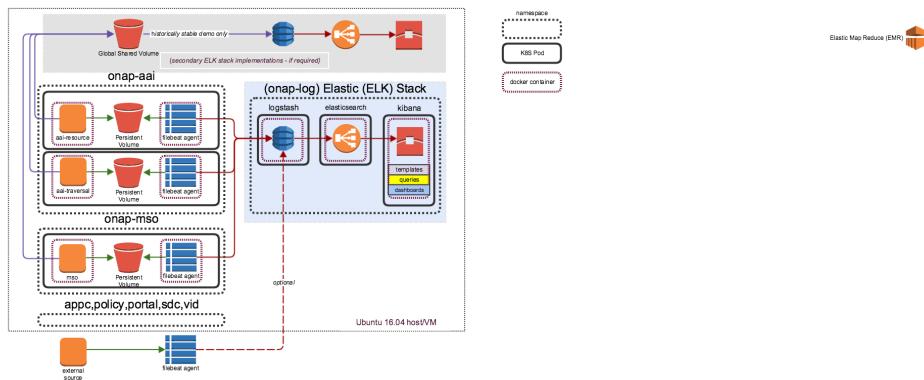
Per component traffic in a real set of use cases over time (# of requests, volume of requests, rate of requests)

## DI 3: 20171010: AAI Logging API Debug

```
public class PreAaiAjscInterceptor implements AjscInterceptor {  
  
    public boolean allowOrReject(HttpServletRequest req, HttpServletResponse resp, Map<?, ?> paramMap)  
  
        LoggingContext.requestId(req.getHeader("X-TransactionId"));  
  
        LoggingContext.partnerName(req.getHeader("X-FromAppId"));  
  
    public class AAIApplServletContextListener implements ServletContextListener {  
  
        private static final EELFLogger LOGGER = EELFManager.getInstance().getLogger  
(AAIApplServletContextListener.class.getName());  
        public void contextInitialized(ServletContextEvent arg0)  
        {  
  
            System.setProperty("org.openecomp.aai.serverStarted", "false");  
  
            LOGGER.info("****AAI Server initialization started...");  
    }
```

## DevOps

### OOM Deployment



Southbound APIs

## Northbound APIs

<https://github.com/opentracing>

<http://zipkin.io/>

<https://www.cncf.io/blog/2016/10/11/opentracing-joins-the-cloud-native-computing-foundation/>

<https://github.com/jaegertracing/jaeger>

## Discussions

[Re: Carrier Grade Requirements \(consolidated\)](#)

## Links

Reference ELK stack - [LOG-50](#) - Getting issue details... [STATUS](#)

<https://kubernetes.io/docs/concepts/cluster-administration/logging/>

Operate