Veritas Loom

Loom Deployment Guide

Release 0.7.0

Veritas Technologies LLC

Apr 07, 2018

CONTENTS

1 Introduction to Loom Deployment **3**

2 Loom: Deployment Models **5**
- 2.1 On Premise Deployment 5
- 2.2 Public Cloud Deployment 5
- 2.3 Hybrid Cloud Deployment 5

3 Installation Prerequisites **7**
- 3.1 Planning & Preparation 7
- 3.2 System Requirements 7
- 3.3 Verify Dependencies 7

4 Bootstrapping Loom **9**
- 4.1 On-boarding multi-tier Applications on Loom 10

5 Loom: Cloud Deployment **11**
- 5.1 Azure Deployment Guide 11
- 5.2 Overview 12
- 5.3 Oracle Deployment 17

6 Loom On-Premise Deployment **19**

7 Loom Hybrid Deployment **21**

8 Loom: De-installation Procedure **23**
- 8.1 Service Shutdown 23
- 8.2 De-installing component A 23
- 8.3 Upgrading Loom 23
- 8.4 Cleanup Scripts 23

9 Deployment Troubleshooting Guide **25**
- 9.1 Resolving Basic Loom deployment issues 25
- 9.2 Loom Service Patching 31
- 9.3 Troubleshooting Loom networking issues 33
- 9.4 Control plane Issues 33
- 9.5 Data plane Issues 33
- 9.6 Message flow between data plane/control plane 34
- 9.7 Tenant deployment issues 34
- 9.8 Security Issues 34
- 9.9 Issues related to Loom logs, Access logs 34
- 9.10 Moving to different availability set 34

9.11 Issues related to sluggish Loom Application response 34
9.12 Issues related to Loom Metrics 34
9.13 Issues related to System Load 34

10 Deployment FAQ **35**
10.1 Question A: Related to any type of Loom deployment 35

- *Introduction to Loom Deployment*
- *Loom: Deployment Models*
  - *On Premise Deployment*
  - *Public Cloud Deployment*
  - *Hybrid Cloud Deployment*
- *Installation Prerequisites*
  - *Planning & Preparation*
  - *System Requirements*
  - *Verify Dependencies*
- *Bootstrapping Loom*
  - *On-boarding multi-tier Applications on Loom*
- *Loom: Cloud Deployment*
  - *Azure Deployment Guide*
  - *Overview*
  - *Oracle Deployment*
- *Loom On-Premise Deployment*
- *Loom Hybrid Deployment*
- *Loom: De-installation Procedure*
  - *Service Shutdown*
  - *De-installing component A*
  - *Upgrading Loom*
  - *Cleanup Scripts*
- *Deployment Troubleshooting Guide*
  - *Resolving Basic Loom deployment issues*
  - *Loom Service Patching*
  - *Troubleshooting Loom networking issues*
  - *Control plane Issues*
  - *Data plane Issues*
  - *Message flow between data plane/control plane*
  - *Tenant deployment issues*
  - *Security Issues*
  - *Issues related to Loom logs, Access logs*
  - *Moving to different availability set*
  - *Issues related to sluggish Loom Application response*
  - *Issues related to Loom Metrics*

  - *Issues related to System Load*
- *Deployment FAQ*
  - *Question A: Related to any type of Loom deployment*

CHAPTER

ONE

INTRODUCTION TO LOOM DEPLOYMENT

This section will cover introduction of Loom Installation and Deployment Methodology as seen by an external user. A user refers to DevOps Admin or Enterprise Admin in charge of deploying Loom Platform.

It will also cover how this guide is organized. After introduction section, there is a section on Deployment models describing different ways in which Loom can be deployed for use in an enterprise. Next, it will cover deployment and install process for each method in detail. The last section is about general troubleshooting and FAQ for overall Loom Platform deployment and install process. In addition to the generic troubleshooting section, there will be per deployment model specific troubleshooting and FAQ guides that will cover topics specific to a particular deployment model.

See also:

This is a **PLACEHOLDER** content.

Please note, this is placeholder text according to current outline. We will refine the content in subsequent iterations as UK folks plug in the specifics for say Cloud based deployment.

CHAPTER

TWO

LOOM: DEPLOYMENT MODELS

Loom can be deployed as per the enterprise needs either on premise, or in a private cloud, or public cloud, or a hybrid environment. The platform offers flexible deployment on-premise either on bare metal servers or virtual machines.

Other details TBD in consultation with Platform team.

2.1 On Premise Deployment

- Bare Metal Server
- Virtual Machine

2.2 Public Cloud Deployment

- Azure
- Oracle
- AWS
- Others

Cloud-Provider	Deployment Guide Reference
Azure	Refer to *Azure Deployment Guide*
Oracle	TBD
AWS	TBD

2.3 Hybrid Cloud Deployment

Loom can also be deployed as a hybrid cloud. The first version of the product does not support this model. This section will be updated in subsequent releases of Loom.

CHAPTER

THREE

INSTALLATION PREREQUISITES

This section will contain the software required to deploy Loom. This is a generic section not specific to any deployment model. Deployment Model specific pre-requisites will be covered in the respective sections.

For e.g., to bring up the control plane what are the software and setup needed. See details here: Issue in Jira (IMP-964)

3.1 Planning & Preparation

Placeholder Outline of Planning & Preparation Section

3.2 System Requirements

For latest Loom System Requirements, refer to Platform Technical Specifications Section.

3.3 Verify Dependencies

- Loom Deployment requires several Open Source Software components.
- See the ing_loom_all_dep_list utilized by Loom Platform and Loom Applications.

CHAPTER

FOUR

BOOTSTRAPPING LOOM

In the previous sections of this Installation Guide, we covered various options available to an enterprise for Loom Platform deployment and pre-requisites for the same.

This section explains how Loom Platform itself is bootstrapped in various deployment models. Besides that, it lists workflows on how various Loom components and services are instantiated. Next, it covers how to on board an end-to-end multi-tiered application on Loom Platform.

At a conceptual level, there are 4 stages of Loom Platform deployment that need to be completed by the Platform, Partner and Tenant Admin before the end user can begin using various features offered by Platform and its Apps such as Visibility and Classification.

- **STAGE1:** As a first step, the Platform and its components need to be downloaded and initiated (for an on-premise deployment). In case of a Cloud based deployment, this first step is usually carried out either by Veritas or one of its Partners. During this phase, infrastructure components including persistent databases, micro-services comprising of the Platform and Control Plane constituents are instantiated.
- **STAGE2:** Next, the Data Planes need to be setup for every tenant or customer organization and user accounts created in the context of each tenant. This can be done only by tenant Admin, so Loom Platform deployment Admin must setup Partner Admin accounts (if a Partner such as Cloud Provider is hosting Loom Platform) or create respective Tenant Admin to carry out Data Plane orchestration. The Data Plane setup requires credentials that are internal to the Tenant organization - for e.g., Azure access credentials, AWS customer credentials etc. Besides this, the Tenant Admin also needs to create user accounts and assign roles such as Storage/IT Admin, end users, and others. The Platform Admin or the Partner Admin is also required to define the enterprise wide Data Access policies to enable data data scans, audits and tracking for various enterprise data assets.
- **STAGE3:** After, User account definition and Data Plane configuration, the Tenant Admin or assigned user with Storage/IT Admin role can plug in different data sources within the organization that need to be analyzed and scanned for gaining data insights and visibility. This may require access to various filers, cloud storage and other data sources within the enterprise. (Question: Does that mean that there may be **unknown** data sources that are being managed and expenses made for storage that may not even be known to an enterprise? There is no auto detection of filers / storage sources in Loom Platform - right?) The
- **STAGE4:** Once users are defined, policies and data sources configured, end users can begin using the Platform and access Platform Apps that are deployed in their context. Note, not all Platform Apps may be visible to all tenant accounts. The default ones accessible to all include Platform UI App, Visibility & Classification App. Also, not all user roles may be allowed to view all kinds of enterprise information. For e.g., only a certain kinds of privileged users may have access to all PII stored within the enterprise. Loom Platform can scan and tag all data provided it to via policies, data sources and other filters and classify information based on filters and tags.

Content-TBD Add a high level picture with 4 blobs and arrows pointing from first stage to the fourth one. Jason-help-needed!

Figure below goes to the next level of detail describing Loom Platform bootstrapping at a conceptual level. Different color highlight different stages. You can see various Platform Admin roles and personas that carry out each of these steps to ensure Platform services are up and running for end users to begin interacting with it.

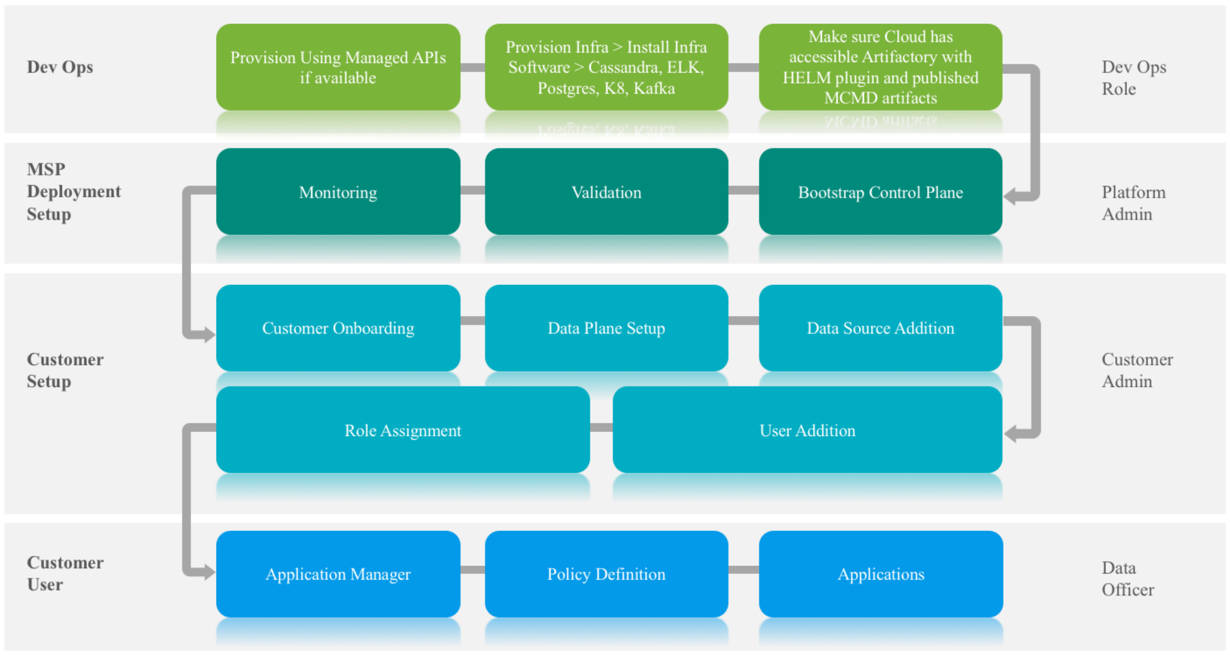


Fig. 4.1: Figure: Stages of Loom Platform Bootstrapping Process

See also:

This is a **DOC-CONTRIB** Request

This section contains placeholder for content that requires shared contribution by Loom dev / owner personnel. As part of Loom Deployment section, we need to add the step by step instructions on how an end user would actually deploy Loom. Use the following JIRA EPIC pointers for more information and user stories regarding the same.

- User Onboarding section: Refer to Jira 887
- Control Plane Bootstrapping: Refer to Jira 966
- How to deploy a data plane (on-premise): Refer to Jira 993
- How to register a deployed data plane with control plane: Refer to Jira 995

4.1 On-boarding multi-tier Applications on Loom

Describe how a multi-tier application can on-board Loom Platform. This includes third party applications such as ??? (find list from Platform team). What about in-built apps (if any)? Choosing and installing via Marketplace?

- Generic 3 tier apps
- Specific App – from Marketplace
- 3rd party apps
- SaaS Hosted Apps

CHAPTER

FIVE

LOOM: CLOUD DEPLOYMENT

Introduction

Veritas Loom Cloud Deployment Model is designed to work on any public cloud provider such as Microsoft Azure, Amazon AWS, Google GCP, Oracle Cloud, IBM Cloud and other cloud providers. In the very first release, Loom is available as a pre-deployed service hosted on Microsoft Azure. In future releases, Veritas Partners will be able to deploy Loom themselves on a cloud of their choice.

Overview

The following figure gives a high level overview of a typical Loom Cloud deployment. It comprises of a starter VM that helps to jump-start a Kubernetes cluster with requisite Loom services that are needed to bring up various Loom capabilities such as managing Loom Users, managing connectivity to various enterprise content repositories, administration of Loom Deployment, managing data classification jobs and their status, tracking data accesses, setting up of enterprise specific data policies and more.

Note: To be updated

This section will contain image / diagram / figures explaining how a Loom Cloud deployment is done, various components and where each is deployed. How the enterprise / MSP can set up Loom on cloud, which components to interact with and more. Looking up to Branislav / Adam to provide a high level sketch that can be added here.

Note, this diagram will be generic to any cloud deployment. Specific cloud deployments will be covered in individual deployment guides dedicated to each supported cloud provider.

5.1 Azure Deployment Guide

This guide covers the design and implementation details of deploying Veritas Loom on Microsoft Azure cloud. It is intended for DevOps personnel that are entrusted with deploying Loom for Veritas or its Partner Enterprise. The very first release of Veritas Loom is *only* available as a SaaS deployment hosted on Microsoft Azure and managed by Veritas. In future, Veritas Partners can bring up their own Loom SaaS deployment on Microsoft Azure and onboard customer organizations. For deploying Loom on other cloud providers, refer to Cloud Provider specific Loom deployment Guide.

The following sections cover Loom deployment on Microsoft Azure and how to bring up all the services to ensure deployment status is healthy and ready for further usage. For details on how to use Loom, refer to the Loom User Getting Started Guide.

Loom Azure Deployment comprises of four key areas: Cloud Configuration, Loom Infrastructure Setup, High Availability(HA) and Cluster Deployment, Security and Access Control and other Azure specific deployment details. The last section of this guide addresses some of the common issues faced during Loom deployment on Microsoft Azure and troubleshooting tips.

5.2 Overview

The figure below captures a very high level diagram of a typical Loom SaaS deployment on Microsoft Azure based cloud infrastructure.

> **Caution:** The overview section is under the process of refinement / review and updates.
>
> The figure below is based on the one provided by Engineering DevOps on Confluence Page titled Azure Topology. The internal confluence reference will be eliminated from the final draft of this guide, once the details are crystallized and ready for sharing with external customers.

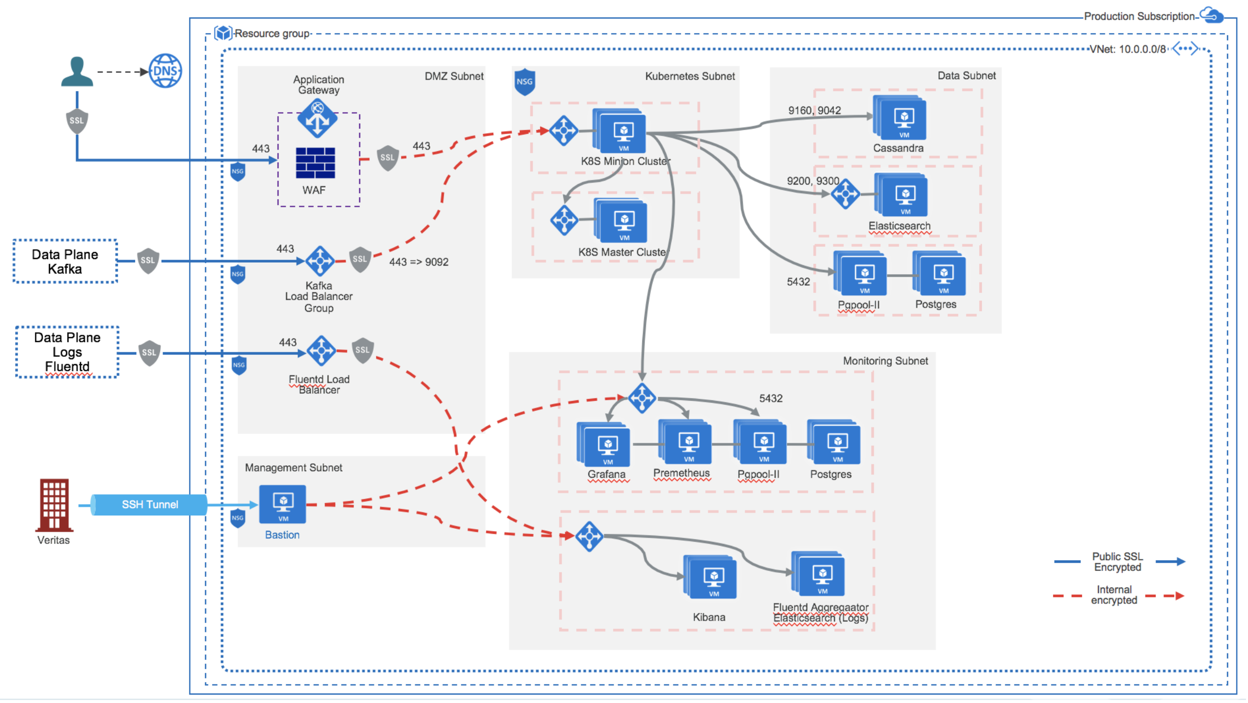


Fig. 5.1: Figure: Loom (Alpha2) Control Plane Deployment Topology on Microsoft Azure

> **Caution:** The figure below shows the Loom Azure SaaS Data Plane Deployment Details. Parking it here for the next refresh cycle of the Loom Deployment Guide.

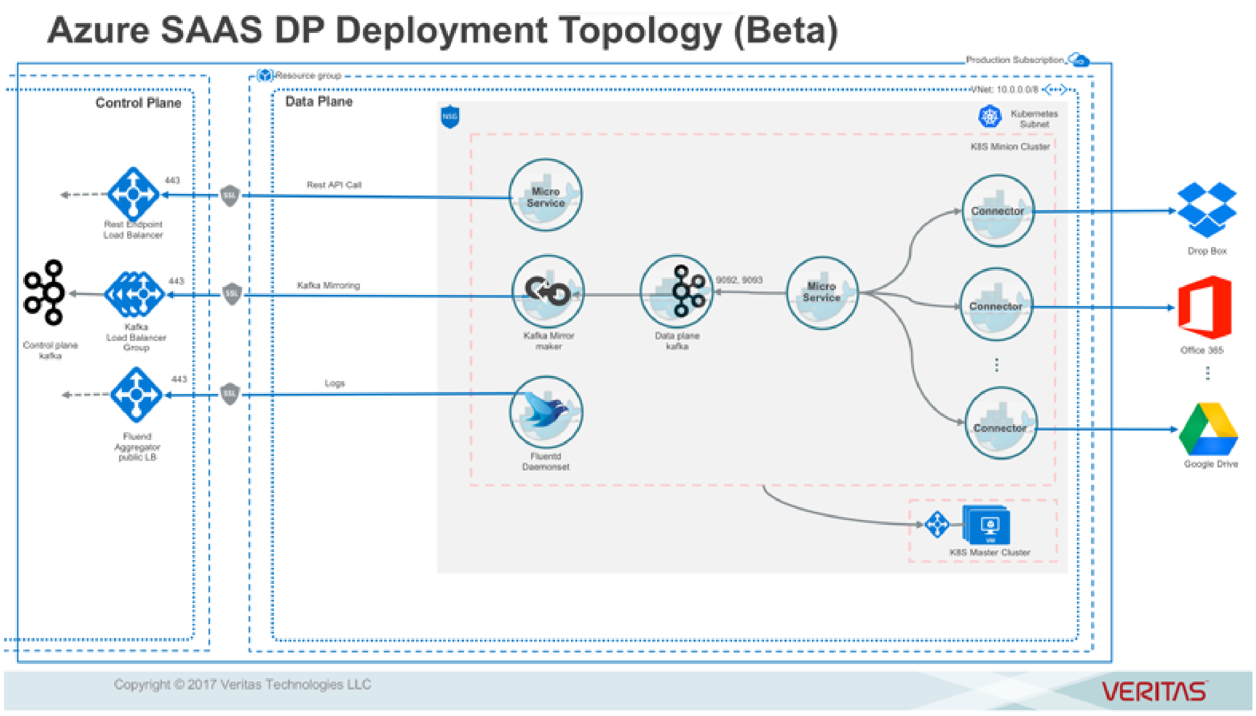


Fig. 5.2: Figure: Loom (Alpha2) Data Plane Deployment Topology on Microsoft Azure

5.2.1 Azure Configuration

1. Identification of region/s
   - Number of Availability set required and define how different component uses it.
     - Rack id : will this be used in no SQL Engines?
2. Identification of Azure account
3. Identification of regions to be deployed
4. Identification of domain and sub-domain name(if required)
   - DNS
   - Static IP required?
5. VPN connection across regions
6. Deployment Topology
   - Azure | Oracle
7. General
   - Network connectivity
     - Subnets
       * Database
       * DMZ zone
       * CP/DP communication
     - VNET peering

- Traffic flow
- Bandwidth/latency
- Bastion server - Management server
- WAF for all external network calls

8. Network flow diagram and port access across system boundaries
   - UI accessing LB
   - Communication from CP to DP, Ports opened
   - Communication from DP to CP, Ports opened
   - Network flow: tenant isolation depicted diagram
   - IPSec or HTTPS/SSL from CP to DP
9. Firewall/NAT services in front of load balancer
   - Learning, Blocking, slow connection
10. Front end load balancer : Azure| oracle load balancer
    - SSL offloading - Terminate at load balancer?
    - Data at motion security consideration has to be finalized and updated
    - NGINX HA

5.2.2 Loom Infrastructure Setup

Following is the suggested outline for this sub-section. Information related to these topics need to be fleshed out for Loom Infrastructure Setup in the context of Azure deployment.

Deploying infra services

1. DMZ zone
   - Application gateway
   - Firewall
     - which firewall and its configuration management
     - Different Security group
     - Different subnet
   - Management
     - Bastion server
       * Different Security Group
       * Different subnet
2. Infra components
   - Cassandra
     - Partitioning strategy

  - Adding a node
  - Decommissioning node
  - Re-balancing
  - Database backup/recovery
  - version update
- Elastic search
  - Master and data node deployment
  - Add a node
  - De-commission node
  - Identifying bottle necks, Split brain, Data loss, reconciliation
  - Version update
- Postgres
  - Master/Slave
  - Virtual IP/DNS
    - Switching between Master/slave
  - Capacity planning with data growth
  - version update
- K8 cluster
  - Integration with Veritas LDAP user ?
  - Namespace design
  - Across region deployment (if required)??
    - VPC peering kind of in azure
    - Availability set
- Kafka
  - Ports opened
  - Partition
  - Number of Brokers
    - Addition
    - Decommission
- RabbitMQ
  - Federation required across region(if required)
  - SSL
- System crashes
  - Handling of VM/EC2 crashes
    - Volume strategy
    - recovery

    - Corruption of volumes/recovery mechanism

  - DR/BR
    - Database backup
      - Incremental
    - Geographic Disaster

5.2.3 HA / Cluster Deployment

This is the suggested outline for HA /Cluster Deployment section in case of Azure Deployment.

- Define availability set
- Define number of nodes that will be supported by default and adding incremental node
- VPN connection across regions
  - IPSec or HTTPS/SSL from CP to DP
  - Artifactory server access to upgrade

5.2.4 Security & Access Control

Here is a recommended outline for Security and Access Control Setup in Azure Deployment scenario of MCDM Platform (Loom). We need to flesh out this section along with diagrams / figures.

- Access control to deployed system
  - Via Bastion server
  - Least privileges
  - Key management and rotation policy
  - Audit log
- Security
  - Handle - when security is comprised
    - Patch update/services
    - recovery
  - Intrusion detection system - Firewall
    - SQL injection prevention, XSS attacks
    - WAF ?
    - Define golden configuration
  - OS hardening
    - Docker user and access control
      - Restriction File access, encrypted volume
    - Host firewall

- Network traffic
  - Network ACL, Security groups, Routing table on Azure/oracle/AWS
    - Golden configuration
  - List all ports that are white listed
- Auditing/Incident notification
  - Integration with messaging/email
- Key management - Data at motion, Data at rest
  - Encryption

5.2.5 Miscellaneous Setups and Azure Tuning for Loom

This section covers other miscellaneous configuration settings and tuning required to ensure that Veritas Loom is successfully up and running normally in Microsoft Azure Cloud. Need help from Adam and Branislav to populate this section. ~

5.2.6 Azure Deployment Troubleshooting Guide

Loom Deployment on Microsoft Azure Cloud might face certain deployment issues. This guide contains some of the frequently encountered issues and fixes, tips on how to solve them to attain a healthy Loom Deployment status.

TBD - Need help from Branislav, Dinesh, Adam and others to populate this section.

5.3 Oracle Deployment

TBD

CHAPTER

SIX

LOOM ON-PREMISE DEPLOYMENT

This is not supported in the initial release.

This guide is a placeholder for now. It will be covered once the functionality is added in Loom platform.

CHAPTER

SEVEN

LOOM HYBRID DEPLOYMENT

This mode of deployment is not supported in the initial release of Loom.

This Guide is a placeholder until this functionality is supported in Loom.

CHAPTER

EIGHT

LOOM: DE-INSTALLATION PROCEDURE

TBD

8.1 Service Shutdown

TBD

8.2 De-installing component A

TBD

8.3 Upgrading Loom

TBD

8.4 Cleanup Scripts

TBD

CHAPTER

NINE

DEPLOYMENT TROUBLESHOOTING GUIDE

This guide lists down some of the key issues faced during Loom Deployment and provides useful tips and insights on how to address Loom Deployment issues. The audience for this guide are the Loom Platform Admins, Loom Application developers and others trying to deploy Loom and its components.

Warning: The following content is primarily focused on Loom Deployment in SaaS Model on Microsoft Azure Cloud. This is the only supported deployment as of Loom Alpha releases.

In future, this guide may also contain information related to other kinds of Loom Deployment. For details on different kinds of Loom Deployments see *Loom: Deployment Models*.

- *Resolving Basic Loom deployment issues*
- *Loom Service Patching*
- *Troubleshooting Loom networking issues*
- *Control plane Issues*
- *Data plane Issues*
- *Message flow between data plane/control plane*
- *Tenant deployment issues*
- *Security Issues*
- *Issues related to Loom logs, Access logs*
- *Moving to different availability set*
- *Issues related to sluggish Loom Application response*
- *Issues related to Loom Metrics*
- *Issues related to System Load*

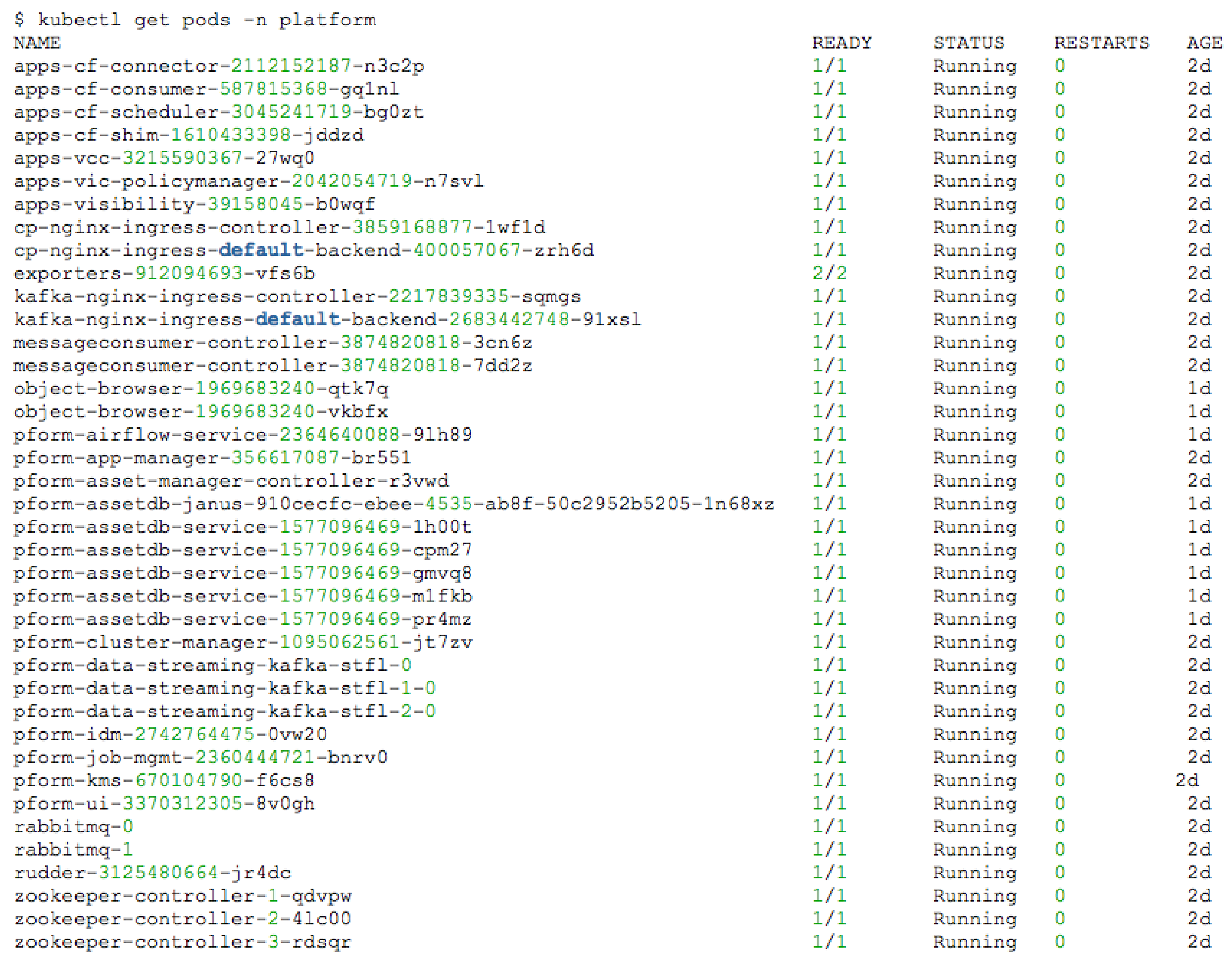
9.1 Resolving Basic Loom deployment issues

This section lists some of the Loom deployment related issues faced in early builds, alpha1, alpha2 and how to resolve the same.

9.1.1 How to begin Loom deployment troubleshooting?

The first step is to check the status of the Control And Data Planes in a Loom deployment.

Following figures show all the pods in healthy state. The first figure refers to a healthy control plane showing all the Loom pods that make up a Loom Control Plane.



```
$ kubectl get pods -n platform
NAME                                                              READY     STATUS    RESTARTS   AGE
apps-cf-connector-2112152187-n3c2p                                1/1       Running   0          2d
apps-cf-consumer-587815368-gq1nl                                  1/1       Running   0          2d
apps-cf-scheduler-3045241719-bg0zt                                1/1       Running   0          2d
apps-cf-shim-1610433398-jddzd                                     1/1       Running   0          2d
apps-vcc-3215590367-27wq0                                         1/1       Running   0          2d
apps-vic-policymanager-2042054719-n7svl                           1/1       Running   0          2d
apps-visibility-39158045-b0wqf                                    1/1       Running   0          2d
cp-nginx-ingress-controller-3859168877-1wf1d                      1/1       Running   0          2d
cp-nginx-ingress-default-backend-400057067-zrh6d                  1/1       Running   0          2d
exporters-912094693-vfs6b                                         2/2       Running   0          2d
kafka-nginx-ingress-controller-2217839335-sqmgs                   1/1       Running   0          2d
kafka-nginx-ingress-default-backend-2683442748-91xsl              1/1       Running   0          2d
messageconsumer-controller-3874820818-3cn6z                       1/1       Running   0          2d
messageconsumer-controller-3874820818-7dd2z                       1/1       Running   0          2d
object-browser-1969683240-qtk7q                                   1/1       Running   0          1d
object-browser-1969683240-vkbfx                                   1/1       Running   0          1d
pform-airflow-service-2364640088-91h89                            1/1       Running   0          1d
pform-app-manager-356617087-br551                                 1/1       Running   0          2d
pform-asset-manager-controller-r3vwd                              1/1       Running   0          2d
pform-assetdb-janus-910cecfc-ebee-4535-ab8f-50c2952b5205-1n68xz   1/1       Running   0          1d
pform-assetdb-service-1577096469-1h00t                            1/1       Running   0          1d
pform-assetdb-service-1577096469-cpm27                            1/1       Running   0          1d
pform-assetdb-service-1577096469-gmvq8                            1/1       Running   0          1d
pform-assetdb-service-1577096469-m1fkb                            1/1       Running   0          1d
pform-assetdb-service-1577096469-pr4mz                            1/1       Running   0          1d
pform-cluster-manager-1095062561-jt7zv                            1/1       Running   0          2d
pform-data-streaming-kafka-stfl-0                                 1/1       Running   0          2d
pform-data-streaming-kafka-stfl-1-0                               1/1       Running   0          2d
pform-data-streaming-kafka-stfl-2-0                               1/1       Running   0          2d
pform-idm-2742764475-0vw20                                        1/1       Running   0          2d
pform-job-mgmt-2360444721-bnrv0                                   1/1       Running   0          2d
pform-kms-670104790-f6cs8                                         1/1       Running   0         2d
pform-ui-3370312305-8v0gh                                         1/1       Running   0          2d
rabbitmq-0                                                        1/1       Running   0          2d
rabbitmq-1                                                        1/1       Running   0          2d
rudder-3125480664-jr4dc                                           1/1       Running   0          2d
zookeeper-controller-1-qdvpw                                      1/1       Running   0          2d
zookeeper-controller-2-41c00                                      1/1       Running   0          2d
zookeeper-controller-3-rdsqr                                      1/1       Running   0          2d
```

Fig. 9.1: Figure: Loom Control Plane - all pods are healthy in this figure.

The figure below refers to a healthy data plane showing all the Loom pods that make up a Loom Data Plane.

9.1.2 Where to look for more details is a Pod is not healthy?

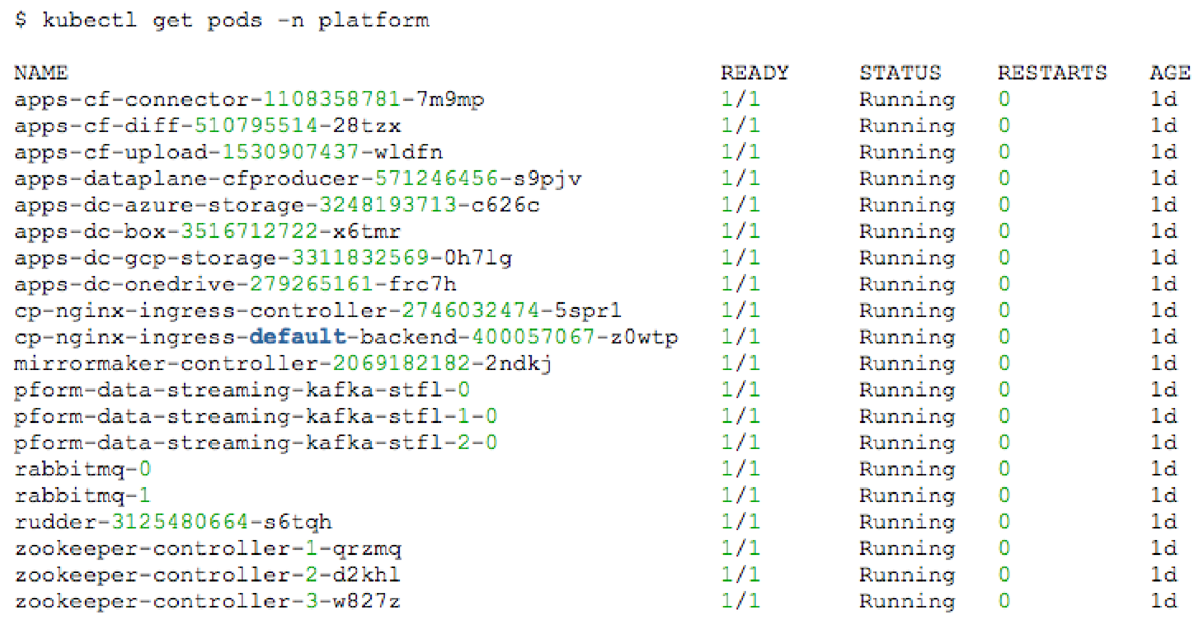
If a pod looks like it is not running properly the next step is to look at it's log file using the following command:

```
kubectl logs <podname> -n platform
```

Figure below shows the command and its output.

9.1.3 How to determine which software version is deployed for each pod?

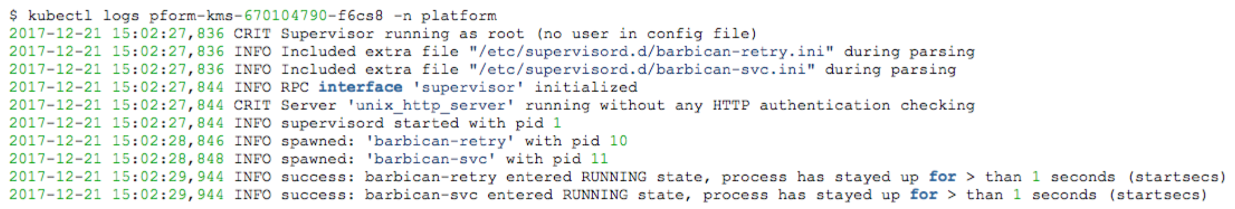
If you want to check the version of a particular pod, use the command:



```
$ kubectl get pods -n platform

NAME                                                READY     STATUS     RESTARTS    AGE
apps-cf-connector-1108358781-7m9mp                  1/1       Running    0           1d
apps-cf-diff-510795514-28tzx                        1/1       Running    0           1d
apps-cf-upload-1530907437-wldfn                     1/1       Running    0           1d
apps-dataplane-cfproducer-571246456-s9pjv           1/1       Running    0           1d
apps-dc-azure-storage-3248193713-c626c              1/1       Running    0           1d
apps-dc-box-3516712722-x6tmr                        1/1       Running    0           1d
apps-dc-gcp-storage-3311832569-0h7lg                1/1       Running    0           1d
apps-dc-onedrive-279265161-frc7h                    1/1       Running    0           1d
cp-nginx-ingress-controller-2746032474-5spr1        1/1       Running    0           1d
cp-nginx-ingress-default-backend-400057067-z0wtp    1/1       Running    0           1d
mirrormaker-controller-2069182182-2ndkj             1/1       Running    0           1d
pform-data-streaming-kafka-stfl-0                   1/1       Running    0           1d
pform-data-streaming-kafka-stfl-1-0                 1/1       Running    0           1d
pform-data-streaming-kafka-stfl-2-0                 1/1       Running    0           1d
rabbitmq-0                                          1/1       Running    0           1d
rabbitmq-1                                          1/1       Running    0           1d
rudder-3125480664-s6tqh                             1/1       Running    0           1d
zookeeper-controller-1-qrzmq                        1/1       Running    0           1d
zookeeper-controller-2-d2khl                        1/1       Running    0           1d
zookeeper-controller-3-w827z                        1/1       Running    0           1d
```

Fig. 9.2: Figure: Loom Data Plane - all pods are healthy in this figure.

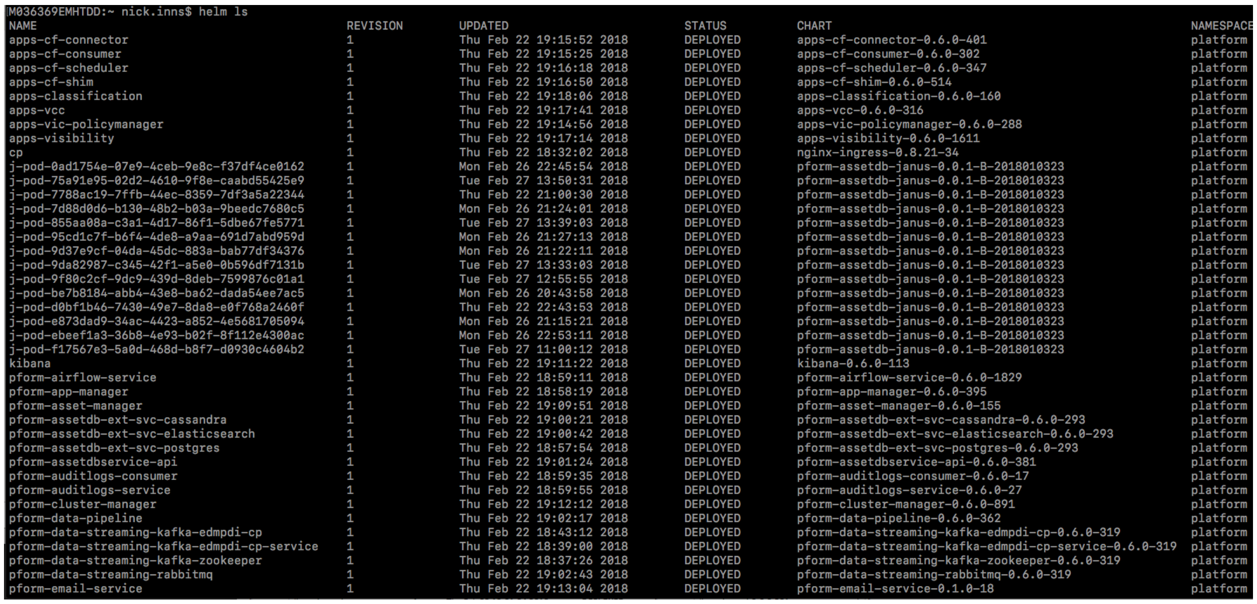


```
$ kubectl logs pform-kms-670104790-f6cs8 -n platform
2017-12-21 15:02:27,836 CRIT Supervisor running as root (no user in config file)
2017-12-21 15:02:27,836 INFO Included extra file "/etc/supervisord.d/barbican-retry.ini" during parsing
2017-12-21 15:02:27,836 INFO Included extra file "/etc/supervisord.d/barbican-svc.ini" during parsing
2017-12-21 15:02:27,844 INFO RPC interface 'supervisor' initialized
2017-12-21 15:02:27,844 CRIT Server 'unix_http_server' running without any HTTP authentication checking
2017-12-21 15:02:27,844 INFO supervisord started with pid 1
2017-12-21 15:02:28,846 INFO spawned: 'barbican-retry' with pid 10
2017-12-21 15:02:28,848 INFO spawned: 'barbican-svc' with pid 11
2017-12-21 15:02:29,944 INFO success: barbican-retry entered RUNNING state, process has stayed up for > than 1 seconds (startsecs)
2017-12-21 15:02:29,944 INFO success: barbican-svc entered RUNNING state, process has stayed up for > than 1 seconds (startsecs)
```

Fig. 9.3: Figure: Examine Pod Log for details

```
helm ls
```

The figure below shows all services are at version B-2017122115 - apart from Airflow which has been patched to version B-2017122123.



```
M036369EMHTDD:~ nick.inns$ helm ls
NAME                                            REVISION        UPDATED                         STATUS          CHART                                                   NAMESPACE
apps-cf-connector                               1               Thu Feb 22 19:15:52 2018        DEPLOYED        apps-cf-connector-0.6.0-401                             platform
apps-cf-consumer                                1               Thu Feb 22 19:15:25 2018        DEPLOYED        apps-cf-consumer-0.6.0-302                              platform
apps-cf-scheduler                               1               Thu Feb 22 19:16:18 2018        DEPLOYED        apps-cf-scheduler-0.6.0-347                             platform
apps-cf-shim                                    1               Thu Feb 22 19:16:50 2018        DEPLOYED        apps-cf-shim-0.6.0-514                                  platform
apps-classification                             1               Thu Feb 22 19:18:06 2018        DEPLOYED        apps-classification-0.6.0-160                           platform
apps-vcc                                        1               Thu Feb 22 19:17:41 2018        DEPLOYED        apps-vcc-0.6.0-316                                      platform
apps-vic-policymanager                          1               Thu Feb 22 19:14:56 2018        DEPLOYED        apps-vic-policymanager-0.6.0-288                        platform
apps-visibility                                 1               Thu Feb 22 19:17:14 2018        DEPLOYED        apps-visibility-0.6.0-1611                              platform
cp                                              1               Thu Feb 22 18:32:02 2018        DEPLOYED        nginx-ingress-0.8.21-34                                 platform
j-pod-0ad1754e-07e9-4ceb-9e8c-f37df4ce0162      1               Mon Feb 26 22:45:54 2018        DEPLOYED        pform-assetdb-janus-0.0.1-B-2018010323                  platform
j-pod-75a91e95-02d2-4610-9f8e-caabd55425e9      1               Tue Feb 27 13:50:31 2018        DEPLOYED        pform-assetdb-janus-0.0.1-B-2018010323                  platform
j-pod-7788ac19-7ffb-44ec-8359-7df3a5a22344      1               Thu Feb 22 21:00:30 2018        DEPLOYED        pform-assetdb-janus-0.0.1-B-2018010323                  platform
j-pod-7d88d0d6-b130-48b2-b03a-9beedc7680c5      1               Mon Feb 26 21:24:01 2018        DEPLOYED        pform-assetdb-janus-0.0.1-B-2018010323                  platform
j-pod-855aa08a-c3a1-4d17-86f1-5dbe67fe5771      1               Tue Feb 27 13:39:03 2018        DEPLOYED        pform-assetdb-janus-0.0.1-B-2018010323                  platform
j-pod-95cd1c7f-b6f4-4de8-a9aa-691d7abd959d      1               Mon Feb 26 21:27:13 2018        DEPLOYED        pform-assetdb-janus-0.0.1-B-2018010323                  platform
j-pod-9d37e9cf-04da-45dc-883a-bab77df34376      1               Mon Feb 26 21:22:11 2018        DEPLOYED        pform-assetdb-janus-0.0.1-B-2018010323                  platform
j-pod-9da82987-c345-42f1-a5e0-0b596df7131b      1               Tue Feb 27 13:33:03 2018        DEPLOYED        pform-assetdb-janus-0.0.1-B-2018010323                  platform
j-pod-9f80c2cf-9dc9-439d-8deb-7599876c01a1      1               Tue Feb 27 12:55:55 2018        DEPLOYED        pform-assetdb-janus-0.0.1-B-2018010323                  platform
j-pod-be7b8184-abb4-43e8-ba62-dada54ee7ac5      1               Mon Feb 26 20:43:58 2018        DEPLOYED        pform-assetdb-janus-0.0.1-B-2018010323                  platform
j-pod-d0bf1b46-7430-49e7-8da8-e0f768a2460f      1               Thu Feb 22 22:43:53 2018        DEPLOYED        pform-assetdb-janus-0.0.1-B-2018010323                  platform
j-pod-e873dad9-34ac-4423-a852-4e5681705094      1               Mon Feb 26 21:15:21 2018        DEPLOYED        pform-assetdb-janus-0.0.1-B-2018010323                  platform
j-pod-ebeef1a3-36b8-4e93-b02f-8f112e4300ac      1               Mon Feb 26 22:53:11 2018        DEPLOYED        pform-assetdb-janus-0.0.1-B-2018010323                  platform
j-pod-f17567e3-5a0d-468d-b8f7-d0930c4604b2      1               Tue Feb 27 11:00:12 2018        DEPLOYED        pform-assetdb-janus-0.0.1-B-2018010323                  platform
kibana                                          1               Thu Feb 22 19:11:22 2018        DEPLOYED        kibana-0.6.0-113                                        platform
pform-airflow-service                           1               Thu Feb 22 18:59:11 2018        DEPLOYED        pform-airflow-service-0.6.0-1829                        platform
pform-app-manager                               1               Thu Feb 22 18:58:19 2018        DEPLOYED        pform-app-manager-0.6.0-395                             platform
pform-asset-manager                             1               Thu Feb 22 19:09:51 2018        DEPLOYED        pform-asset-manager-0.6.0-155                           platform
pform-assetdb-ext-svc-cassandra                 1               Thu Feb 22 19:00:21 2018        DEPLOYED        pform-assetdb-ext-svc-cassandra-0.6.0-293               platform
pform-assetdb-ext-svc-elasticsearch             1               Thu Feb 22 19:00:42 2018        DEPLOYED        pform-assetdb-ext-svc-elasticsearch-0.6.0-293           platform
pform-assetdb-ext-svc-postgres                  1               Thu Feb 22 18:57:54 2018        DEPLOYED        pform-assetdb-ext-svc-postgres-0.6.0-293                platform
pform-assetdbservice-api                        1               Thu Feb 22 19:01:24 2018        DEPLOYED        pform-assetdbservice-api-0.6.0-381                      platform
pform-auditlogs-consumer                        1               Thu Feb 22 18:59:35 2018        DEPLOYED        pform-auditlogs-consumer-0.6.0-17                       platform
pform-auditlogs-service                         1               Thu Feb 22 18:59:55 2018        DEPLOYED        pform-auditlogs-service-0.6.0-27                        platform
pform-cluster-manager                           1               Thu Feb 22 19:12:12 2018        DEPLOYED        pform-cluster-manager-0.6.0-891                         platform
pform-data-pipeline                             1               Thu Feb 22 19:02:17 2018        DEPLOYED        pform-data-pipeline-0.6.0-362                           platform
pform-data-streaming-kafka-edmpdi-cp            1               Thu Feb 22 18:43:12 2018        DEPLOYED        pform-data-streaming-kafka-edmpdi-cp-0.6.0-319          platform
pform-data-streaming-kafka-edmpdi-cp-service    1               Thu Feb 22 18:39:00 2018        DEPLOYED        pform-data-streaming-kafka-edmpdi-cp-service-0.6.0-319  platform
pform-data-streaming-kafka-zookeeper            1               Thu Feb 22 18:37:26 2018        DEPLOYED        pform-data-streaming-kafka-zookeeper-0.6.0-319          platform
pform-data-streaming-rabbitmq                   1               Thu Feb 22 19:02:43 2018        DEPLOYED        pform-data-streaming-rabbitmq-0.6.0-319                 platform
pform-email-service                             1               Thu Feb 22 19:13:04 2018        DEPLOYED        pform-email-service-0.1.0-18                            platform
```

Fig. 9.4: Figure: Examine Service version number deployed in pod

9.1.4 Looking for Dependency Health

If services that rely upon Postgres (Airflow, App Manager, KMS, Cluster Manager, IDM, Job Management) are failing it might be caused by a problem with the Postgres deployment on the persistent VM. Check to see if the databases are present by ssh'ing into the persistent VM and using psql to list the databases hosted on the server. The output should look similar to the following:

Figure: Examine Service Dependency for failures

9.1.5 Loom Tenant Creation Issues

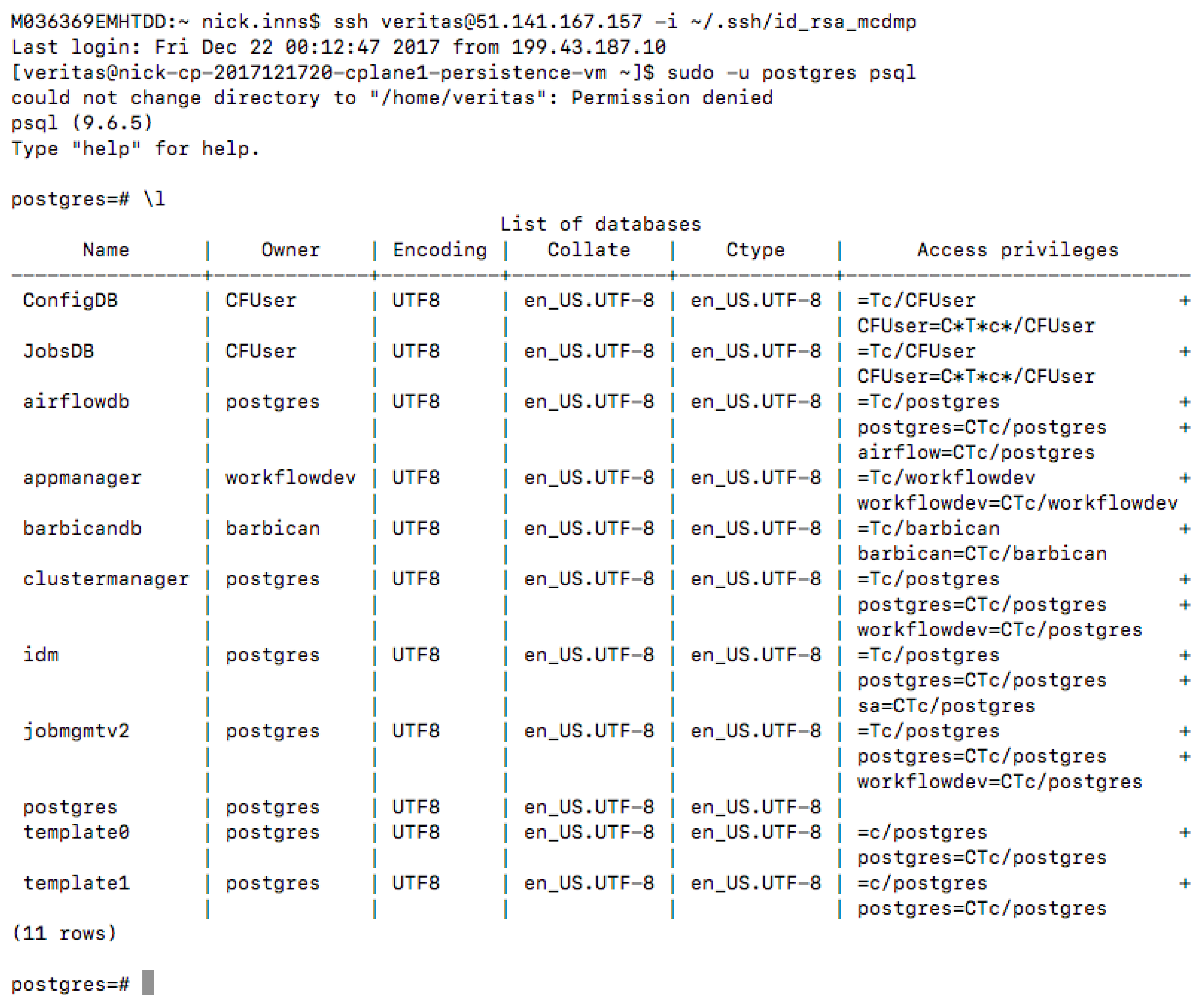
If you find that tenants are not being created properly (i.e. they never reach a state of 'Active' in the UI), the first thing to check is whether the create_tenant dag has been triggered. Use port forwarding to map a local port (e.g. 8080) to port 8080 on the Airflow pod and use a web browser to view the Airflow UI using http://localhost:8080. For e.g., use the command:

```
kubectl port-forward pform-airflow-service-2073380254-bgff7 8080:8080 -n platform
```

In the example below we can see that although Airflow indicates a data plane DAG run has taken place twice (one failed, one succeeded) no executions are listed against add_tenant. So we know that the add tenant request never reached Airflow:

Figure: Examine Add Tenant DAG Execution Status

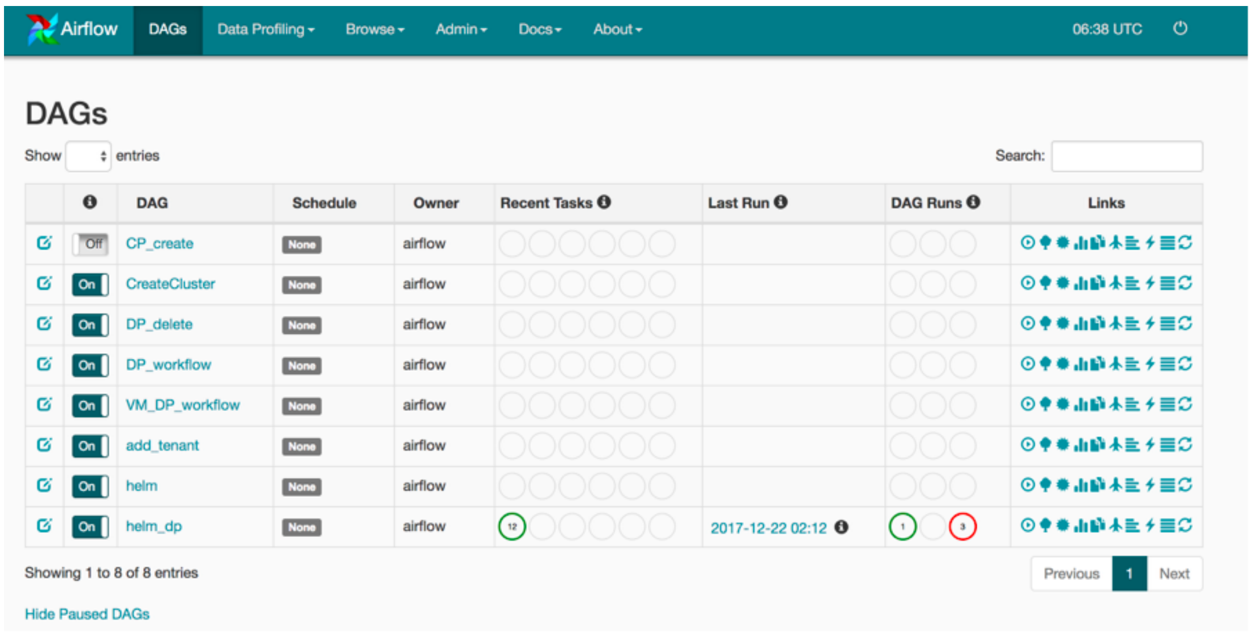
If a DAG run for create_tenant has taken place then the log files for this may indicate a problem. In this situation we know that a DAG run has not taken place. So the next step is to check the logs for cluster manager. Use the command:



```
M036369EMHTDD:~ nick.inns$ ssh veritas@51.141.167.157 -i ~/.ssh/id_rsa_mcdmp
Last login: Fri Dec 22 00:12:47 2017 from 199.43.187.10
[veritas@nick-cp-2017121720-cplane1-persistence-vm ~]$ sudo -u postgres psql
could not change directory to "/home/veritas": Permission denied
psql (9.6.5)
Type "help" for help.

postgres=# \l
                                         List of databases
      Name      |   Owner     | Encoding |  Collate   |   Ctype    |      Access privileges
----------------+-------------+----------+------------+------------+-----------------------------
 ConfigDB       | CFUser      | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =Tc/CFUser                  +
                |             |          |            |            | CFUser=C*T*c*/CFUser
 JobsDB         | CFUser      | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =Tc/CFUser                  +
                |             |          |            |            | CFUser=C*T*c*/CFUser
 airflowdb      | postgres    | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =Tc/postgres                +
                |             |          |            |            | postgres=CTc/postgres       +
                |             |          |            |            | airflow=CTc/postgres
 appmanager     | workflowdev | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =Tc/workflowdev             +
                |             |          |            |            | workflowdev=CTc/workflowdev
 barbicandb     | barbican    | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =Tc/barbican                +
                |             |          |            |            | barbican=CTc/barbican
 clustermanager | postgres    | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =Tc/postgres                +
                |             |          |            |            | postgres=CTc/postgres       +
                |             |          |            |            | workflowdev=CTc/postgres
 idm            | postgres    | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =Tc/postgres                +
                |             |          |            |            | postgres=CTc/postgres       +
                |             |          |            |            | sa=CTc/postgres
 jobmgmtv2      | postgres    | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =Tc/postgres                +
                |             |          |            |            | postgres=CTc/postgres       +
                |             |          |            |            | workflowdev=CTc/postgres
 postgres       | postgres    | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 template0      | postgres    | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres                 +
                |             |          |            |            | postgres=CTc/postgres
 template1      | postgres    | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres                 +
                |             |          |            |            | postgres=CTc/postgres
(11 rows)

postgres=#
```



DAGs

Show entries   Search:

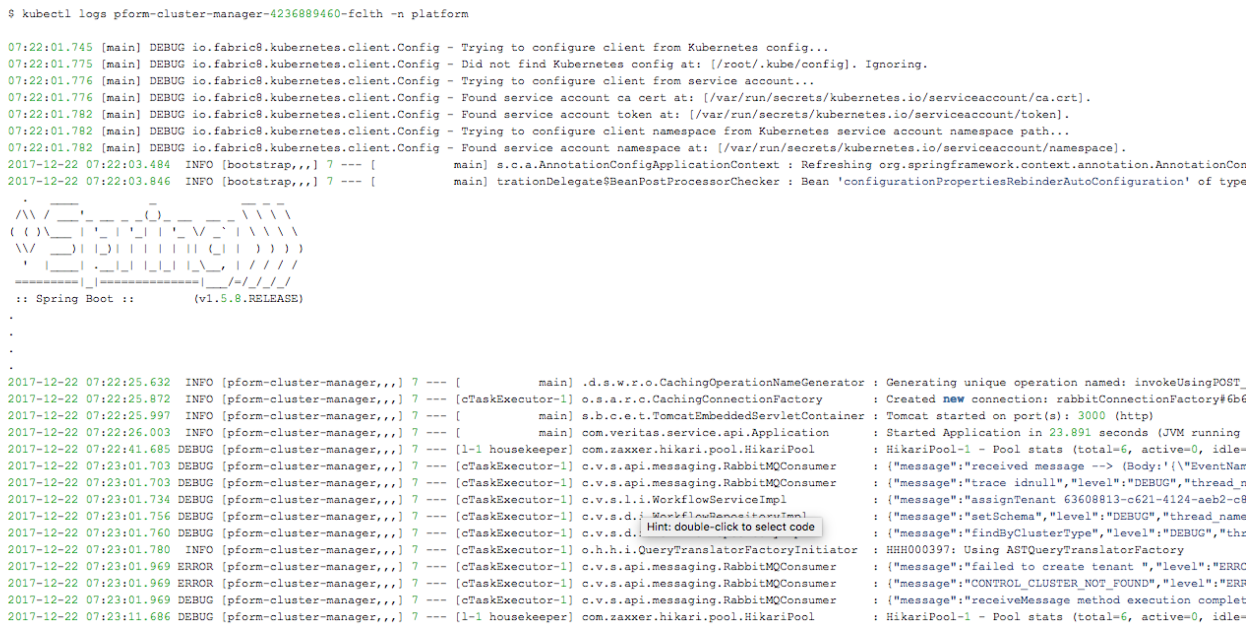
		DAG	Schedule	Owner	Recent Tasks	Last Run	DAG Runs	Links
	Off	CP_create	None	airflow				
	On	CreateCluster	None	airflow				
	On	DP_delete	None	airflow				
	On	DP_workflow	None	airflow				
	On	VM_DP_workflow	None	airflow				
	On	add_tenant	None	airflow				
	On	helm	None	airflow				
	On	helm_dp	None	airflow	12	2017-12-22 02:12	1 3	

Showing 1 to 8 of 8 entries   Previous 1 Next

Hide Paused DAGs

```
kubectl logs <cluster manager pod name> -n platform
```

In the figure below, notice the "CONTROL_CLUSTER_NOT_FOUND" error. That needs further debugging on what went wrong with the cluster.



```
$ kubectl logs pform-cluster-manager-4236889460-fclth -n platform

07:22:01.745 [main] DEBUG io.fabric8.kubernetes.client.Config - Trying to configure client from Kubernetes config...
07:22:01.775 [main] DEBUG io.fabric8.kubernetes.client.Config - Did not find Kubernetes config at: [/root/.kube/config]. Ignoring.
07:22:01.776 [main] DEBUG io.fabric8.kubernetes.client.Config - Trying to configure client from service account...
07:22:01.776 [main] DEBUG io.fabric8.kubernetes.client.Config - Found service account ca cert at: [/var/run/secrets/kubernetes.io/serviceaccount/ca.crt].
07:22:01.782 [main] DEBUG io.fabric8.kubernetes.client.Config - Found service account token at: [/var/run/secrets/kubernetes.io/serviceaccount/token].
07:22:01.782 [main] DEBUG io.fabric8.kubernetes.client.Config - Trying to configure client namespace from Kubernetes service account namespace path...
07:22:01.782 [main] DEBUG io.fabric8.kubernetes.client.Config - Found service account namespace at: [/var/run/secrets/kubernetes.io/serviceaccount/namespace].
2017-12-22 07:22:03.484  INFO [bootstrap,,,] 7 --- [           main] s.c.a.AnnotationConfigApplicationContext : Refreshing org.springframework.context.annotation.AnnotationCon
2017-12-22 07:22:03.846  INFO [bootstrap,,,] 7 --- [           main] trationDelegate$BeanPostProcessorChecker : Bean 'configurationPropertiesRebinderAutoConfiguration' of type
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v1.5.8.RELEASE)
.
.
.
.
2017-12-22 07:22:25.632  INFO [pform-cluster-manager,,,] 7 --- [           main] .d.s.w.r.o.CachingOperationNameGenerator : Generating unique operation named: invokeUsingPOST_
2017-12-22 07:22:25.872  INFO [pform-cluster-manager,,,] 7 --- [cTaskExecutor-1] o.s.a.r.c.CachingConnectionFactory       : Created new connection: rabbitConnectionFactory#6b6
2017-12-22 07:22:25.997  INFO [pform-cluster-manager,,,] 7 --- [           main] s.b.c.e.t.TomcatEmbeddedServletContainer : Tomcat started on port(s): 3000 (http)
2017-12-22 07:22:26.003  INFO [pform-cluster-manager,,,] 7 --- [           main] com.veritas.service.api.Application      : Started Application in 23.891 seconds (JVM running
2017-12-22 07:22:41.685 DEBUG [pform-cluster-manager,,,] 7 --- [l-1 housekeeper] com.zaxxer.hikari.pool.HikariPool        : HikariPool-1 - Pool stats (total=6, active=0, idle=
2017-12-22 07:23:01.703 DEBUG [pform-cluster-manager,,,] 7 --- [cTaskExecutor-1] c.v.s.api.messaging.RabbitMQConsumer     : {"message":"received message --> (Body:'{\"EventNam
2017-12-22 07:23:01.703 DEBUG [pform-cluster-manager,,,] 7 --- [cTaskExecutor-1] c.v.s.api.messaging.RabbitMQConsumer     : {"message":"trace idnull","level":"DEBUG","thread_n
2017-12-22 07:23:01.734 DEBUG [pform-cluster-manager,,,] 7 --- [cTaskExecutor-1] c.v.s.l.i.WorkflowServiceImpl            : {"message":"assignTenant 63608813-c621-4124-aeb2-c8
2017-12-22 07:23:01.756 DEBUG [pform-cluster-manager,,,] 7 --- [cTaskExecutor-1] c.v.s.d.i.WorkflowRepositoryImpl         : {"message":"setSchema","level":"DEBUG","thread_name
2017-12-22 07:23:01.760 DEBUG [pform-cluster-manager,,,] 7 --- [cTaskExecutor-1] c.v.s.d.                                 : {"message":"findByClusterType","level":"DEBUG","thr
2017-12-22 07:23:01.780  INFO [pform-cluster-manager,,,] 7 --- [cTaskExecutor-1] o.h.h.i.QueryTranslatorFactoryInitiator  : HHH000397: Using ASTQueryTranslatorFactory
2017-12-22 07:23:01.969 ERROR [pform-cluster-manager,,,] 7 --- [cTaskExecutor-1] c.v.s.api.messaging.RabbitMQConsumer     : {"message":"failed to create tenant ","level":"ERRC
2017-12-22 07:23:01.969 ERROR [pform-cluster-manager,,,] 7 --- [cTaskExecutor-1] c.v.s.api.messaging.RabbitMQConsumer     : {"message":"CONTROL_CLUSTER_NOT_FOUND","level":"ERR
2017-12-22 07:23:01.969 DEBUG [pform-cluster-manager,,,] 7 --- [cTaskExecutor-1] c.v.s.api.messaging.RabbitMQConsumer     : {"message":"receiveMessage method execution complet
2017-12-22 07:23:11.686 DEBUG [pform-cluster-manager,,,] 7 --- [l-1 housekeeper] com.zaxxer.hikari.pool.HikariPool        : HikariPool-1 - Pool stats (total=6, active=0, idle=
```

Figure: Tenant could not be created as there was cluster error

9.1.6 Content source added but cannot view data in the Loom Application Dashboard

If you have added a content source and are not seeing data in the visibility app you might want to check to see if jobs have been created in the database. To do this log in to the persistent VM via ssh (see above) and access Postgres setting the context to the JobsDB. In the select command below, substitute the tenantId with the tenantId of the tenant you are trouble-shooting:

```
$ psql -d JobsDB -U CFUser
JobsDB=> select * from "JobsQueue" where "TenantId"='3337b7fa-342f-4522-b62b-
↪e4c36b1856cf';
```

You should see jobs of type DISCOVERY, SCAN and UPLOAD created for your tenant with statuses of CREATED, RUNNING or SUCCEEDED. There maybe multiple entries for each depending on the content source - e.g. for Box, a set of jobs will be created for each account that has access to the data.

9.1.7 Kafka does not produce or consume any data, how to fix it?

On one of the recent builds we have seen an issue that after initial bootstrap Kafka may not be producing and consuming any data. This seems to be a timing issue during bootstrap. We do not have any specific logs or errors being thrown from Kafka server. Workaround for this issue would be running following 3 commands and wait until pods come up properly:

```
$ kubectl -n platform delete pod pform-data-streaming-kafka-stfl-0
$ kubectl -n platform delete pod pform-data-streaming-kafka-stfl-1-0
$ kubectl -n platform delete pod pform-data-streaming-kafka-stfl-2-0
```

This will delete the pods only - not the deployment or service configuration.

Check that the pods start properly by using the get pods command (they will go through the normal cycle of ContainerCreating to Running). After the pods have started log into Kafka on two different terminal sessions and then produce and consume data by using the following commands (remember to substitute the pod name with the one from your environment):

```
$ kubectl -n platform exec  -it pform-data-streaming-kafka-stfl-0 sh

Producer:

sh-4.2# kafka-console-producer.sh --broker-list pform-data-streaming-kafka-
↪service:9093 --topic INFRASTRUCTURE_DISCOVERY
<Type Anything on the console and wait for 30 seconds. Than press ctrl+c >

Consumer:

sh-4.2# kafka-console-consumer.sh --bootstrap-server pform-data-streaming-kafka-
↪service:9093 --topic INFRASTRUCTURE_DISCOVERY --from-beginning
```

You should see the characters you typed into the producer session appearing in the consumer session. If you don't then this would suggest there is still a configuration issue with Kafka.

9.2 Loom Service Patching

1. Quick patching Service for troubleshooting

   Most of the Loom Services Docker images provide run command which start the service as part of container initialization process. The service starts with pid 1, and it becomes difficult to restart the service without bringing down the container (i.e., service pod).

   During troubleshooting you may require to make some quick changes (log message, variable change or fix) and test it before committing the changes. The typical workflow in this case involves rebuilding the Docker image for the service and redeploying it again in Kubernetes. In an ideal deployment, as part of the run command of Docker image, we should install a package (dpkg) in the container. This makes it easy to restart the service without bringing down the container.

   Following method can be used so save some time during troubleshooting of Loom Services and applying service patches. In short, what you need to do is use "Sleep infinity" as a Docker RUN command, which would start the container with root process (pid 1) in sleep. This makes it easy to replace any jar /config information in this container.

   - From your service directory edit the Dockerfile and change the run command to:

     ```
     sleep infinity
     ```

     For e.g.,

         CMD ["sleep","infinity"]

   - Get the list of pods using the command:

```
helm ls
```

- Delete the service you wish to troubleshoot using the command:

```
helm delete --purge <service-name>
```

- Build the Docker image of the service using the edited Dockerfile (step 1) with different tag using the command:

```
   docker build -t <service-name>:<tag>

For e.g.,

   docker build -t apps-classification-orchestrator:test
```

- Login into the repository you want to push the image to using the command:

```
    docker login < repository> -u <username> -p <password>

For e.g.,

    docker login mcdmpdemoregistry.azurecr.io -u test -p test
```

- Tag the image with repository name using the command:

```
    docker tag <service-name>:<tag> < repository>/<service-name>:<tag>

For e.g.,

    docker tag apps-classification-orchestrator:test  mcdmpdemoregistry.
↪azurecr.io/apps-classification-orchestrator:test
```

- Push the image to repository using the command:

```
    docker push < repository >/<service-name>:<tag>

For e.g.,

    docker push  mcdmpdemoregistry.azurecr.io/apps-classification-
↪orchestrator:test
```

- From deployment/helm folder run the following commands to deploy the service:
  - Verify correct repository and tag are applied in deployment.yaml using the command:

```
  helm install --debug --dry-run --set image.repository=<repository>/ --
↪set image.tag=<tag> <service-name>/

For e.g.,

  helm install --debug --dry-run --set image.repository=mcdmpdemoregistry.
↪azurecr.io/ --set image.tag=test apps-classification-orchestrator/
```

  - Install the service using command:

```
  helm install --namespace=<namespace> --name=<service-name> --set image.
↪repository=<repository>/ --set image.tag=<tag> <service-name>
```

```
For e.g.,

  helm install --namespace=platform --name=apps-classification-
↪orchestrator --set image.repository=mcdmpdemoregistry.azurecr.io/ --set␣
↪image.tag=test apps-classification-orchestrator/
```

- Pod is up and running with infinite sleep. You can exec into pod and execute the jar, script, etc., within it. Also new jars, files, scripts, etc., can be copied into the pod.

2. Patch for terminal Dimensions not supported inside Pods

   Many a times, when accessing a Pod say Kafka or Loom Connector Service, the pod doesn't use the entire terminal dimension. The text gets wrapped around or overlapped or only few lines are shown while editing a file inside pod using vi editor.

   The following script can be used to get rid of the problem for that instance.

   - In your machine, create a file, say file with the name exec_script.sh
   - Update the file with the following commands:

```
if [ "$1" = "" ]; then
  echo "Usage: bash exec_script.sh <pod_name>"
  exit 1
fi

COLUMNS=`tput cols`
LINES=`tput lines`
TERM=xterm
kubectl --namespace=platform exec -i -t $1 env COLUMNS=$COLUMNS LINES=$LINES␣
↪TERM=$TERM bash
```

   - To get inside the pod instead of using:

```
kubectl --namespace=platform -it <pod_name> bash
```

     use the following command:

```
bash exec_script.sh <pod_name>
```

9.3 Troubleshooting Loom networking issues

TBD

9.4 Control plane Issues

TBD

9.5 Data plane Issues

TBD

9.6 Message flow between data plane/control plane

TBD

9.7 Tenant deployment issues

TBD

9.8 Security Issues

TBD

9.9 Issues related to Loom logs, Access logs

TBD

9.10 Moving to different availability set

TBD

9.11 Issues related to sluggish Loom Application response

TBD

9.12 Issues related to Loom Metrics

TBD

9.13 Issues related to System Load

TBD

CHAPTER

TEN

DEPLOYMENT FAQ

This is a FAQ document template which will cover generic Q/A related to Loom deployment. Note this will cover all deployment issues which are not specific to a particular deployment model.

10.1 Question A: Related to any type of Loom deployment

TBD