Acumos Documentation

Release 1.0

Jul 08, 2021

Contents

Acumos AI is a platform and open source framework that makes it easy to build, share, and deploy AI apps. Acumos standardizes the infrastructure stack and components required to run an out-of-the-box general AI environment. This frees data scientists and model trainers to focus on their core competencies and accelerates innovation.

CHAPTER 1

Acumos Releases

1.1 Platform

1.1.1 Demeter Release, 10 June 2020

1.1.1.1 Demeter Release Notes

Demeter is the fourth release of the Acumos platform.

- Release Name: Demeter
- Release Version: 4.0
- Release Date: 10 June 2020
- Wiki: Demeter Release Notes

Release Highlights

- **Cloud Enablement:**
  - Containerized platform deployment incorporating cloud native functions, horizontal scaling, and implementation flexibility
- **On Boarding:**
  - CLI message response with the Acumos Docker model
  - Support for Pre-dockerized and Dockerized model URI with protobuf file to render models usable in Design studio.
- **Licensing:**
  - Activity tracking and reporting – License usage manager (LUM) maintains logs of model usage
  - Integration of License module with Portal UI.

- **Training:**
  - Bidirectional communication over the federation link between subscriber and supplier instances to support ML life cycle management and continuous learning
- ML Work Bench:
  - **Predictor Manager:**
    - The Predictor Manager manages the model deployment, visualization of deployment metadata and association to a project.
  - **Data source:**
    - The Data Source feature allows user to create and associate project data with a model to create, update, and delete data set used for training, validation and testing.
- **Portal:**
  - Integration of License module with Portal

Installation

Acumos provides a Zero to Acumos (Z2A) installation process for deploying to Ubuntu 20.04 development environments. The Z2A installation covers the case of starting with a VM or the case of starting from an existing Kubernetes installation. The Z2A was built as a modular design leveraging installation already existing for components that Acumos depends on (Nexus/MariaDB/etc.) The mindset is for any Acumos dependency you may use your own or use the defalt that is part of the Z2A installation.

To get begin Start_Here

Supported Browsers, Devices, and Resolutions

Detailed information can be found on the ../supported-browsers page.

How to Get Help

There are two options for getting help installing and using the Acumos platform:

- the Acumos Community mailing list
  - You must create an account to use the mailing list
  - Please use `[acumosaicommunity]Help:` plus your question in the subject line
- StackOverflow

Whether you post to the mailing list or to Stack Overflow, please be as descriptive as possible in the body so it's easier for a community member to help.

How to Report a Bug

You can report a bug by creating a Jira issue in the Acumos Jira. You must log in with your Linux Foundation ID. Guidelines for the content of a bug report are here.

1.1.1.2 Demeter Manifest

Operating System

The multi-node installation of Acumos was tested on Ubuntu 16.04 LTS.

The One Click installation has been run on Centos 7 and Ubuntu 16, 17, and 18.

Platform Components

The components that comprise the Acumos Platform are released as Docker images on Nexus.

Individual component release notes may be accessed from the *Component Releases* page.

Core Components

Project	Component	Artifact
Catalog, Data Model, and Data Management	Common Data Service (CDS) – server	acumos/common-dataservice
Catalog, Data Model, and Data Management	Federation	acumos/federation-gateway
Common Services	Microservice Generation	acumos/microservice-generation
Deployment	Azure Client	acumos/acumos-azure-client
Deployment	Kubernetes Client	acumos/kubernetes-client
Deployment	OpenStack Client	acumos/openstack-client
Design Studio	Composition Engine	acumos/ds-compositionengine
License-Manager	License-Manager	acumos/license-rtu-editor
License-Manager	License-Manager	acumos/license-profile-editor
License-Usage-Manager	License-Usage-Manager	acumos/lum-server
License-Usage-Manager	License-Usage-Manager	acumos/lum-db
Model Onboarding	Onboarding	acumos/onboarding-app
OA&M	Elasticsearch	acumos/acumos-elasticsearch
OA&M	Elk-client	acumos/elk-client
OA&M	Filebeat	acumos/acumos-filebeat
OA&M	Kibana	acumos/acumos-kibana
OA&M	Logstash	acumos/acumos-logstash
OA&M	Metricbeat	acumos/acumos-metricbeat
Portal	Portal Backend	acumos/acumos-portal-be
Portal	Portal Frontend	acumos/acumos-portal-fe
Security-Verification	Security-Verification	acumos/security-verification
Workbench	Dashboard-Webcomponent	acumos/dashboard-webcomponent
Workbench	Home-Webcomponent	acumos/home-webcomponent
Workbench	Notebook-Catalog-Webcomponent	acumos/notebook-catalog-webcomponent
Workbench	Notebook-Webcomponent	acumos/notebook-webcomponent
Workbench	Project-Webcomponent	acumos/project-webcomponent
Workbench	Project-Catalog-Webcomponent	acumos/project-catalog-webcomponent
Workbench	Pipeline-Catalog-Webcomponent	acumos/pipeline-catalog-webcomponent
Workbench	Pipeline-Webcomponent	acumos/pipeline-webcomponent
Workbench	Project-Service	acumos/project-service
Workbench	Notebook-Service	acumos/notebook-service
Workbench	Pipeline-Service	acumos/pipeline-service
Workbench	Model-Service	acumos/model-service

Continued on n

Table 1.1 – continued from previous page

Project	Component	Artifact
Workbench	Predictor-Service	acumos/predictor-service
Workbench	Datasource-service	acumos/datasource-service
Workbench	Datasource-webcomponent	acumos/datasource-webcomponent
Workbench	Datasource-catalog-webcomponent	acumos/datasource-catalog-webcomponent
Deployment	Deployment-client	acumos/deployment-client

Model Execution Components

Project	Component	Artifact	Version
Design Studio	SQL Data Broker	sqldatabroker	1.2.0
Design Studio	CSV Data Broker	csvdatabroker	1.4.0
Model Onboarding	Onboarding Base – R	onboarding-base-r	1.2.0
Design Studio	Runtime Orchestrator (Model Connector)	blueprint-orchestrator	2.0.13
Design Studio	Model Runner	h2o-genericjava-modelrunner	2.2.3
DataBroker	Data Broker	databroker-zipbroker	1.0.0
Design Studio	Proto Viewer (Probe)	acumos-proto-viewer	1.5.7

Third Party Software

Software	Version
MariaDB	10.2
Kong	0.11.0
Nexus Repository OSS	3.x
Docker-CE	18.06.1-ce for Ubuntu 16.04

Supporting Libraries Used by Platform Components

These supporting libraries are released as Java JAR files and referenced as libraries by various platform components.

Project	Component	JAR	Version
Acumos-Java-Client	Acumos-Java-Client	java_client	4.2.0
Catalog, Data Model, and Data Management	Common Data Service Client	cmn-data-svc-client	3.1.1
Design Studio	Generic Data Mapper Service	gdmservice	TDB
Design Studio	TOSCAGeneratorClient	TOSCAModelGenerator-Client	2.0.8
License-Manager	License-Manager	License-Manager-Client-Library	1.5.1
Acumos R Client	Acumos-r-client	acumos-r-client	0.3.0
Acumos C Client	Acumos-c-client	acumos-c-client	1.2
Acumos Python Client	Acumos-python-client	acumos-python-client	0.9.5
Python Model Runner	Python-model-runner	python-model-runner	0.2.4

Modeler Client Libraries

These libraries are used by modelers on their local workstations to prepare models for onboarding.

Project	Component	Version	Location
Model Onboarding	acumos-java-client	4.2.0	Nexus

1.1.2 Clio Release, 13 November 2019

1.1.2.1 Clio Release Notes

Clio is the third release of the Acumos platform.

- Release Name: Clio
- Release Version: 3.0.0
- Release Date: 13 November 2019
- Wiki: Clio Release Notes

Release Highlights

- **Model On Boarding / Common Services**
  - Onboarding & Microservice Generation of Spark/Java and C/C++ client
- **Design Studio /Machine Learning Workbench**
  - Enterprise Design Tools Integration to support plug-gable framework
  - Framework extended to support no-SQL database (Couch DB)
  - **Web component support for plug-gable framework**
    - Project Predictor Mapping, Model Asset Mapping & Collaboration
- **Federation**
  - ONAP model Integration with Acumos AI Marketplace
  - O-RAN Integration
- **License Management**
  - License Usage Manager (LUM) – manage license compliance for Acumos models
  - License Entitlement/RTU – support license agreements using standard Open Digital Rights Language
  - License Profile – ability to identify models as commercial
  - IP Asset Protection Rights - Model Activity Tracking & Reporting
- **Deployment**
  - Jenkins as a workflow engine as a stand alone or on-demand Kubernetes service

Installation

For Acumos Multi Node Installation .

Acumos provides a one-click installation script for deploying to Ubuntu 16.04 development environments. Both docker-compose and Kubernetes options are supported. Please see the One Click Deploy User Guide for details.

Supported Browsers, Devices, and Resolutions

Detailed information can be found on the ../supported-browsers page.

How to Get Help

There are two options for getting help installing and using the Acumos platform:

- the Acumos Community mailing list
  - You must create an account to use the mailing list
  - Please use `[acumosaicommunity]Help:` plus your question in the subject line
- StackOverflow

Whether you post to the mailing list or to Stack Overflow, please be as descriptive as possible in the body so it's easier for a community member to help.

How to Report a Bug

You can report a bug by creating a Jira issue in the Acumos Jira. You must log in with your Linux Foundation ID. Guidelines for the content of a bug report are here.

1.1.2.2 Clio Manifest

Operating System

The multi-node installation of Acumos was tested on Ubuntu 16.04 LTS.

The One Click installation has been run on Centos 7 and Ubuntu 16, 17, and 18.

Platform Components

The components that comprise the Acumos Platform are released as Docker images on Nexus.

Individual component release notes may be accessed from the *Component Releases* page.

Core Components

Project	Component	Artifact
Catalog, Data Model, and Data Management	Common Data Service (CDS) – server	acumos/common-dataservice
Catalog, Data Model, and Data Management	Federation	acumos/federation-gateway

Continued on ne

Table 1.2 – continued from previous page

Project	Component	Artifact
Common Services	Microservice Generation	acumos/microservice-generation
Deployment	Azure Client	acumos-azure-client
Deployment	Kubernetes Client	kubernetes-client
Deployment	OpenStack Client	openstack-client
Design Studio	Composition Engine	ds-compositionengine
License-Manager	License-Manager	acumos/license-rtu-editor
License-Manager	License-Manager	acumos/license-profile-editor
License-Usage-Manager	License-Usage-Manager	acumos/lum-server
License-Usage-Manager	License-Usage-Manager	acumos/lum-db
Model Onboarding	Onboarding	acumos/onboarding-app
OA&M	Elasticsearch	acumos-elasticsearch
OA&M	elk-client	elk-client
OA&M	Filebeat	acumos-filebeat
OA&M	Kibana	acumos-kibana
OA&M	Logstash	acumos-logstash
OA&M	Metricbeat	acumos-metricbeat
Portal	Portal Backend	acumos-portal-be
Portal	Portal Frontend	acumos-portal-fe
Security-Verification	Security-Verification	acumos/security-verification
Workbench	Dashboard-Webcomponent	acumos/dashboard-webcomponent
Workbench	Home-Webcomponent	acumos/home-webcomponent
Workbench	Notebook-Catalog-Webcomponent	acumos/notebook-catalog-webcomponent
Workbench	Notebook-Webcomponent	acumos/notebook-webcomponent
Workbench	Project-Webcomponent	acumos/project-webcomponent
Workbench	Project-Catalog-Webcomponent	acumos/project-catalog-webcomponent
Workbench	Pipeline-Catalog-Webcomponent	acumos/pipeline-catalog-webcomponent
Workbench	Pipeline-Webcomponent	acumos/pipeline-webcomponent
Workbench	Project-Service	project-service
Workbench	Notebook-Service	notebook-service
Workbench	Pipeline-Service	pipeline-service
Workbench	Model-Service	model-service
Workbench	Predictor-Service	predictor-service

Model Execution Components

Project	Component	Artifact	Version
Design Studio	SQL Data Broker	sqldatabroker	1.2.0
Design Studio	CSV Data Broker	csvdatabroker	1.4.0
Model Onboarding	Onboarding Base – R	onboarding-base-r	1.0.0
Design Studio	Runtime Orchestrator (Model Connector)	blueprint-orchestrator	2.0.13
Design Studio	Model Runner	h2o-genericjava-modelrunner	2.2.3
DataBroker	Data Broker	databroker-zipbroker	1.0.0
Design Studio	Proto Viewer (Probe)	acumos-proto-viewer	1.5.7

Third Party Software

Software	Version
MariaDB	10.2
Kong	0.11.0
Nexus Repository OSS	3.x
Docker-CE	18.06.1-ce for Ubuntu 16.04

Supporting Libraries Used by Platform Components

These supporting libraries are released as Java JAR files and referenced as libraries by various platform components.

Project	Component	JAR	Version
Acumos-Java-Client	Acumos-Java-Client	java_client	3.1.0
Catalog, Data Model, and Data Management	Common Data Service Client	cmn-data-svc-client	3.0.0
Design Studio	Generic Data Mapper Service	gdmservice	TDB
Design Studio	TOSCAGeneratorClient	TOSCAModelGenerator-Client	2.0.0
License-Manager	License-Manager	License-Manager-Client-Library	1.4.0

Modeler Client Libraries

These libraries are used by modelers on their local workstations to prepare models for onboarding.

Project	Component	Version	Location
Model Onboarding	acumos-java-client	3.1.0	Nexus

1.1.3 Boreas Release, 5 Jun 2019

1.1.3.1 Boreas Release Notes

Boreas is the second release of the Acumos platform.

- Release Name: Boreas
- Release Version: 2.0.0
- Release Date: 5 June 2019
- Wiki: Boreas Release Notes

Release Highlights

Support for onboarding of ONNX, PFA and Dockerized models.

Enhanced Acumos platform peering through a controlled process of partner catalog publication and subscription.

- Global catalog search capability
- Federation of Catalogs

Support for AI/ML model suppliers to provide a commercial software license with their models in the Acumos marketplace.

- Security scans of license metadata for models*[0]
- Support verification of licenses and Right-To-Use for commercial models†[0]
- Logging to enable model activity tracking and reporting

Support for ML Workbench to allow the creation and training of AI/ML models in Acumos platform.

- Support for Notebooks development environment (Jupyter).
- Support for Pipeline (NiFi‡[0]) tools are integrated with Acumos.

Enhanced support for deploying Acumos platform under Kubernetes

Enhanced user experience in portal.

- Publishing, unpublishing, deploying , onboarding, model building, and chaining, etc.

Enhanced logging standards

- Log formats aligned with ONAP.
- Support for Log management tools.

Installation

For Acumos Multi Node Installation .

Acumos provides a one-click installation script for deploying to Ubuntu 16.04 development environments. Both docker-compose and Kubernetes options are supported. Please see the One Click Deploy User Guide for details.

Supported Browsers, Devices, and Resolutions

Detailed information can be found on the ../supported-browsers page.

How to Get Help

There are two options for getting help installing and using the Acumos platform:

- the Acumos Community mailing list
  - You must create an account to use the mailing list
  - Please use `[acumosaicommunity]Help:` plus your question in the subject line
- StackOverflow

Whether you post to the mailing list or to Stack Overflow, please be as descriptive as possible in the body so it's easier for a community member to help.

[0] Disabled with Security Verification turned off.
[0] Disabled with Security Verification turned off.
[0] NiFi Pipeline tools are available as a Beta Feature only under K8S.

How to Report a Bug

You can report a bug by creating a Jira issue in the Acumos Jira. You must log in with your Linux Foundation ID. Guidelines for the content of a bug report are here.

1.1.3.2 Boreas Manifest

Operating System

The multi-node installation of Acumos was tested on Ubuntu 16.04 LTS.

The One Click installation has been run on Centos 7 and Ubuntu 16, 17, and 18.

Platform Components

The components that comprise the Acumos Platform are released as Docker images on Nexus.

Individual component release notes may be accessed from the *Component Releases* page.

Core Components

Project	Component	Artifact	Version
Catalog, Data Model, and Data Management	Common Data Service (CDS) – server	common-dataservice	2.2.4
Catalog, Data Model, and Data Management	Federation	federation-gateway	2.2.0
Common Services	Microservice Generation	microservice-generation	2.12.0
Deployment	Azure Client	acumos-azure-client	2.0.15
Deployment	Kubernetes Client	kubernetes-client	2.0.10
Deployment	OpenStack Client	openstack-client	2.0.12
Design Studio	Composition Engine	ds-compositionengine	2.1.0
Model Onboarding	Onboarding	onboarding-app	2.14.0
OA&M	Elasticsearch	acumos-elasticsearch	2.2.2
OA&M	elk-client	acumos-elk-client	0.0.2
OA&M	Filebeat	acumos-filebeat	2.2.2
OA&M	Kibana	acumos-kibana	2.2.2
OA&M	Logstash	acumos-logstash	2.2.2
OA&M	Metricbeat	acumos-metricbeat	2.2.2
Portal	Portal Backend	acumos-portal-be	2.2.16
Portal	Portal Frontend	acumos-portal-fe	2.2.16

Model Execution Components

Project	Component	Artifact	Version
DataBroker	Data Broker	databroker-zipbroker	1.0.0
Design Studio	CSV Data Broker	csvdatabroker	1.4.0
Design Studio	Model Runner	h2o-genericjava-modelrunner	2.2.3
Design Studio	Proto Viewer (Probe)	acumos-proto-viewer	1.5.7
Design Studio	Runtime Orchestrator (Model Connector)	blueprint-orchestrator	2.0.12
Design Studio	SQL Data Broker	sqldatabroker	1.2.0
Model Onboarding	Onboarding Base – R	onboarding-base-r	1.0.0

Third Party Software

Software	Version
MariaDB	10.2
Kong	0.11.0
Nexus Repository OSS	3.x
Docker-CE	18.06.1-ce for Ubuntu 16.04
Kubernetes	1.10

Supporting Libraries Used by Platform Components

These supporting libraries are released as Java JAR files and referenced as libraries by various platform components.

Project	Component	JAR	Version
Design Studio	Generic Data Mapper Service	gdmservice	TDB
Design Studio	TOSCAGeneratorClient	TOSCAModelGenerator-Client	2.0.0
Catalog, Data Model, and Data Management	Common Data Service Client	cmn-data-svc-client	2.2.2|2.2.2|2.2.2
Common Services	Nexus Client	acumos-nexus-client	2.2.1
Security-Verification	License-Manager	License-Manager-Client-Library	0.0.9
Acumos-Java-Client	Acumos-Java-Client	java_client	2.1.0

Modeler Client Libraries

These libraries are used by modelers on their local workstations to prepare models for onboarding.

Project	Component	Version	Location
Model Onboarding	acumos-java-client	2.2.0	Nexus
Model Onboarding	acumos-python-client	0.8.0	PyPI
Model Onboarding	acumos-r-client	0.2-8	RForge

Model Runners

Project	Component	Version	Location
Common Services	Python DCAE Model Runner	0.1.2	PyPI
Common Services	Python Model Runner	0.2.2	PyPI

1.1.4 Athena Maintenance Release, 12 December 2018

Note: There is a *required* database upgrade to populate Authorship data. Please see User and Author Data Upgrade for CDS 1.18.x for instructions.

1.1.4.1 Athena Maintenance Release Notes

Athena is the first release of the Acumos platform.

- Release Name: Athena Maintenance
- Release Version: 1.1.0
- Release Date: 12 December 2018

Supported Browsers, Devices, and Resolutions

Detailed information can be found on the ../supported-browsers page.

Issues Addressed

Jira AthenaMaintenance-Fixed

Issue Type	Issue key	Component/s	Summary
Bug	ACUMOS-2109	common-dataservice	Need update sql script to populate first-author metadata for Athena in DB
Bug	ACUMOS-2102	portal-marketplace	IST2: Different name is displaying on the model tile on marketplace and manage my model screen for multiple user
Bug	ACUMOS-2074	portal-marketplace	Portal marketplace tile has unnecessary constant text
Story	ACUMOS-2073	portal-marketplace	Portal require author and default to user when publishing to any catalog
Bug	ACUMOS-2056	portal-marketplace	Portal displays incorrect person detail on tile, shows Admin instead of author
Bug	ACUMOS-2008	portal-marketplace	On-Boarding Model contains links to docs.acumos.org/en/latest instead of docs.acumos.org/en/athena
Bug	ACUMOS-1988	portal-marketplace	ADC-Staging - Logged in user not matching name on black bar
Story	ACUMOS-1953	portal-marketplace	Portal don't show first-time user Tag/Theme selection dialog
Bug	ACUMOS-1933	portal-marketplace	IST: Newly Added tag is not displaying on the model tiles (marketplace , manage my model) when user published the model
Bug	ACUMOS-1916	on-boarding	<IST2> <Onboarding> API token authentication not working for Java model when onboarded through CLI
Story	ACUMOS-1818	portal-marketplace	Portal improve power of Marketplace left-side seach-by-keyword field
Bug	ACUMOS-1653	portal-marketplace	IST2: Deploy to Local : Download packages and help is not working on the popup

Known Issues and Limitations

Jira AthenaMaintenance-KnownIssues

Issue Type	Issue key	Component/s	Summary
Bug	ACUMOS-1932	portal-marketplace	IST: Solution name is not displaying in the notification when user published the model to company marketplace
Bug	ACUMOS-1928	on-boarding	<IST> <Onboarding> API token Authentication is not working for R model which is onboarded through CLI
Bug	ACUMOS-1924	portal-marketplace	Edit Peer dialog always sets self status to false
Bug	ACUMOS-1912	portal-marketplace	IST2: Comment Count is getting zero from tiles when user change the view on marketplace screen
Story	ACUMOS-1904	portal-marketplace	IST2: Publish request entry is displaying for a deleted model.
Bug	ACUMOS-1903	portal-marketplace	IST2: When onboarding of a model fail user is not getting both logs by the link provided on the notification bell icon
Bug	ACUMOS-1889	portal-marketplace	IST2: Web Onboarding: Quit(X) is not working during and after uploading of files
Bug	ACUMOS-1885	portal-marketplace	IST2 - Status is not moving for states when model is published
Bug	ACUMOS-1883	common-dataservice	CDS add method to get user unread notification count
Bug	ACUMOS-1882	portal-marketplace	Portal manage-my-models page shows status Not Started altho deploy to cloud process is completed
Bug	ACUMOS-1803	portal-marketplace	IST2: View Comment box(tool tip) getting cut down for blank text on publish request screen
Bug	ACUMOS-1775	portal-marketplace	Portal publish-approve screen does not allow viewing comments after approve/decline
Bug	ACUMOS-1626	portal-marketplace	IST: Author Name is not displaying when user added the success story
Bug	ACUMOS-1531	portal-marketplace	IST2: Manage My Model: Document: Same Document is not getting selected if user cancel first time
Bug	ACUMOS-516	platform-oam	<IST> <OA&M > Logs are not displayed on IST Logcollector when accessed through application

Security Notes

Integrated security and license scanning of models is not available.

Installation

Acumos provides a one-click installation script for deploying to Ubuntu 16.04 development environments. Both docker-compose and Kubernetes options are supported. Please see the One Click Deploy User Guide for details.

Documentation

The Acumos Athena release provides multiple points of documentation:

- A high level *Platform Architecture Guide* of how components relate to each other
- A collection of documentation provided by *each component*

- The Acumos wiki remains a good source of information on meeting plans and notes from committees, project teams and community events

Licenses

Acumos source code is licensed under the Apache Version 2 License. Acumos documentation is licensed under the Creative Commons Attribution 4.0 International License.

How to Get Help

There are two options for getting help installing and using the Acumos platform:

- the Acumos Community mailing list
  - You must create an account to use the mailing list
  - Please use `[acumosaicommunity]Help:` plus your question in the subject line
- StackOverflow

Whether you post to the mailing list or to Stack Overflow, please be as descriptive as possible in the body so it's easier for a community member to help.

How to Report a Bug

You can report a bug by creating a Jira issue in the Acumos Jira. You must log in with your Linux Foundation ID. Guidelines for the content of a bug report are here.

1.1.4.2 Athena Maintenance Manifest

Operating System

The multi-node installation of Acumos was tested on Ubuntu 16.04 LTS.

The One Click installation has been run on Centos 7 and Ubuntu 16, 17, and 18.

Platform Components

The components that comprise the Acumos Platform are released as Docker images on Nexus.

Individual component release notes may be accessed from the *Component Releases* page.

Core Components

Project	Component	Artifact	Version
Catalog, Data Model, and Data Management	Common Data Service (CDS) – server	common-dataservice	1.18.4
Catalog, Data Model, and Data Management	Federation	federation-gateway	1.18.7
Common Services	Microservice Generation	microservice-generation	1.8.2
Deployment	Azure Client	acumos-azure-client	1.2.22
Deployment	Kubernetes Client	kubernetes-client	1.1.0
Deployment	OpenStack Client	openstack-client	1.1.22
Design Studio	Composition Engine	ds-compositionengine	1.40.2
Model Onboarding	Onboarding	onboarding-app	1.39.0
OA&M	Elasticsearch	acumos-elasticsearch	1.18.2
OA&M	Filebeat	acumos-filebeat	1.18.2
OA&M	Kibana	acumos-kibana	1.18.2
OA&M	Logstash	acumos-logstash	1.18.2
OA&M	Metricbeat	acumos-metricbeat	1.18.2
Portal	Hippo CMS	acumos-cms-docker	1.3.5
Portal	Portal Backend	acumos-portal-be	1.16.6
Portal	Portal Frontend	acumos-portal-fe	1.16.6

Model Execution Components

Project	Component	Artifact	Version
DataBroker	Data Broker	databroker-zipbroker	1.0.0
Design Studio	CSV Data Broker	csvdatabroker	1.4.0
Design Studio	Model Runner	h2o-genericjava-modelrunner	2.2.3
Design Studio	Proto Viewer (Probe)	acumos-proto-viewer	1.5.7
Design Studio	Runtime Orchestrator (Model Connector)	blueprint-orchestrator	2.0.11
Design Studio	SQL Data Broker	sqldatabroker	1.2.0
Model Onboarding	Onboarding Base – R	onboarding-base-r	1.0.0

Third Party Software

Software	Version
MariaDB	10.2
Kong	0.11.0
Nexus Repository OSS	3.x
Docker-CE	18.06.1-ce for Ubuntu 16.04
Kubernetes	1.10

Supporting Libraries Used by Platform Components

These supporting libraries are released as Java JAR files and referenced as libraries by various platform components.

Modeler Client Libraries

These libraries are used by modelers on their local workstations to prepare models for onboarding.

Project	Component	Version	Location
Model Onboarding	acumos-java-client	1.11.1	Nexus
Model Onboarding	acumos-python-client	0.7.0	PyPI
Model Onboarding	acumos-r-client	0.2-7	RForge

Model Runners

Project	Component	Version	Location
Common Services	Python DCAE Model Runner	0.1.2	PyPI
Common Services	Python Model Runner	0.2.1	PyPI

1.1.5 Athena Release, 7 Nov 2018

1.1.5.1 Athena Release Notes

Athena is the first release of the Acumos platform.

- Release Name: Athena
- Release Version: 1.0.0
- Release Date: 7 November 2018

Release Highlights

Portal and Marketplace

- Marketplace personalization - ability to choose model tags (IoT, Mobile, Wireless, etc) so those models will appear first in the Marketplace
- Model authorship
- New user email verification
- Publisher role added so models can be approved before being published to the Public Marketplace
- Ability to download Kubernetes artifacts

Design Studio

- Enhanced CSV Data Broker
- SQL Data Broker
- Split and Join capability - parameter-based and array-based split/collation schemes
- Ability to create Directed Acyclic Graph (DAG) composite solutions
- Enhanced Model connector - support for orchestrating DAG solutions
- Enhanced Probe endpoints
- Validate composite solution and generate deployment blueprint

Federation

- Site configuration

Deployment of Models

- Models may be deployed to a local environment as well as to a Cloud environment
- Support added to deploy models to a Kubernetes environment
  - Deploy models on their own servers/VMs under a private Kubernetes environment
  - Deploy models on hardware - workstations or lab servers
  - Avoid complex prerequisites and costs associated with deploying on VMs/Docker

Platform Operation, Administration, and Management

- Deployment of the platform to a Kubernetes environment
- One-click, single node deployment to Kubernetes as well as Docker
- Kibana dashboard

Supported Browsers, Devices, and Resolutions

Detailed information can be found on the ../supported-browsers page.

Known Issues and Limitations

Onboarding

- Java Client: command-line on-boarding does not support API token but does support JWT
- R Client: command-line on-boarding does not support API token but does support JWT

Design Studio

- Design Studio Data Broker, Splitter, and Collator functionality requires that specific toolkit models be on-boarded; see the ../../AcumosUser/portal-admin/addendum/onboard-ds-toolkits section in the Portal and Marketplace Admin Guide for details.

Portal Marketplace UI

- Manage Themes - selecting themes - the instruction in the modal dialog states "Choose your tags. . . " but if you select more than one tag, the error message "You cannot select more than one tag" is displayed; only single tag selection is supported at this time
- ON-BOARDING MODEL page contains incorrect URLs: **To know more about on-boarding, please have a look at** : https://docs.acumos.org/en/latest/AcumosUser/portal-user/portal/index.html should be https://docs.acumos.org/en/athena/AcumosUser/portal-user/portal/index.html
- Web On-boarding: Quit(X) is not working during and after uploading of files for web on-boarding
- Deploy to Local: Help link displayed in the pop-up window does not work
- Notification: Solution name is not displayed in the notification after a user published the model to the Company Marketplace
- Publishing a Model to Company or Public Marketplace
  - A newly added tag is not displayed on the model tiles on the Marketplace and Manage My Model pages when a user publishes a model; workaround: to add a new tag – **after** the model has been published, you need to go back to Publish to Company or Publish to Public and type in the tag name and then click someplace else on the screen for the tag to be added to the model (tag is still not added to drop-down list)
  - Status is not moving for states when a model is published to Company
- Publish Request
  - Filter is applied to entire list but existing page breaks are maintained even if filter results are less than selected number of records/page; workaround: select to show more requests/page than number of requests in the list

Security Notes

Integrated security and license scanning of models is not available.

Installation

Acumos provides a one-click installation script for deploying to Ubuntu 16.04 development environments. Both docker-compose and Kubernetes options are supported. Please see the One Click Deploy User Guide for details.

Documentation

The Acumos Athena release provides multiple points of documentation:

- A high level *Platform Architecture Guide* of how components relate to each other
- A collection of documentation provided by *each component*

- The Acumos wiki remains a good source of information on meeting plans and notes from committees, project teams and community events

Licenses

Acumos source code is licensed under the Apache Version 2 License. Acumos documentation is licensed under the Creative Commons Attribution 4.0 International License.

How to Get Help

There are two options for getting help installing and using the Acumos platform:

- the Acumos Community mailing list
  - You must create an account to use the mailing list
  - Please use `[acumosaicommunity]Help:` plus your question in the subject line
- StackOverflow

Whether you post to the mailing list or to Stack Overflow, please be as descriptive as possible in the body so it's easier for a community member to help.

How to Report a Bug

You can report a bug by creating a Jira issue in the Acumos Jira. You must log in with your Linux Foundation ID. Guidelines for the content of a bug report are here.

1.1.5.2 Athena Manifest

Operating System

The multi-node installation of Acumos was tested on Ubuntu 16.04 LTS.

The One Click installation has been run on Centos 7 and Ubuntu 16, 17, and 18.

Platform Components

The components that comprise the Acumos Platform are released as Docker images on Nexus.

Individual component release notes may be accessed from the *Component Releases* page.

Core Components

Project	Component	Artifact	Version
Catalog, Data Model, and Data Management	Common Data Service (CDS) – server	common-dataservice	1.18.4
Catalog, Data Model, and Data Management	Federation	federation-gateway	1.18.7
Common Services	Microservice Generation	microservice-generation	1.8.2
Deployment	Azure Client	acumos-azure-client	1.2.22
Deployment	Kubernetes Client	kubernetes-client	1.1.0
Deployment	OpenStack Client	openstack-client	1.1.22
Design Studio	Composition Engine	ds-compositionengine	1.40.2
Model Onboarding	Onboarding	onboarding-app	1.39.0
OA&M	Elasticsearch	acumos-elasticsearch	1.18.2
OA&M	Filebeat	acumos-filebeat	1.18.2
OA&M	Kibana	acumos-kibana	1.18.2
OA&M	Logstash	acumos-logstash	1.18.2
OA&M	Metricbeat	acumos-metricbeat	1.18.2
Portal	Hippo CMS	acumos-cms-docker	1.3.5
Portal	Portal Backend	acumos-portal-be	1.16.3
Portal	Portal Frontend	acumos-portal-fe	1.16.3

Model Execution Components

Project	Component	Artifact	Version
DataBroker	Data Broker	databroker-zipbroker	0.0.1
Design Studio	CSV Data Broker	csvdatabroker	1.4.0
Design Studio	Model Runner	h2o-genericjava-modelrunner	2.2.3
Design Studio	Proto Viewer (Probe)	acumos-proto-viewer	1.5.7
Design Studio	Runtime Orchestrator (Model Connector)	blueprint-orchestrator	2.0.11
Design Studio	SQL Data Broker	sqldatabroker	1.2.0
Model Onboarding	Onboarding Base – R	onboarding-base-r	1.0.0

Third Party Software

Software	Version
MariaDB	10.2
Kong	0.11.0
Nexus Repository OSS	3.x
Docker-CE	18.06.1-ce for Ubuntu 16.04
Kubernetes	1.10

Supporting Libraries Used by Platform Components

These supporting libraries are released as Java JAR files and referenced as libraries by various platform components.

Project	Component	JAR	Version
Design Studio	Generic Data Mapper Service	gdmservice	1.2.0
Design Studio	TOSCAGeneratorClient	TOSCAModelGenerator-Client	1.33.1
Catalog, Data Model, and Data Management	Common Data Service Client	cmn-data-svc-client	1.18.2|1.18.3|1.18.4
Common Services	Nexus Client	acumos-nexus-client	2.2.1

Modeler Client Libraries

These libraries are used by modelers on their local workstations to prepare models for onboarding.

Project	Component	Version	Location
Model Onboarding	acumos-java-client	1.11.0	Nexus
Model Onboarding	acumos-python-client	0.7.0	PyPI
Model Onboarding	acumos-r-client	0.2-7	RForge

Model Runners

Project	Component	Version	Location
Common Services	Python DCAE Model Runner	0.1.2	PyPI
Common Services	Python Model Runner	0.2.1	PyPI

1.2 Component and Weekly

1.2.1 Release Notes

1.2.1.1 Component Releases

Each component maintains its own release notes.

Core Components

Catalog, Data Model, and Data Management

- Common Data Service
- Federation Gateway
- Model Schema

Common Services

- H2O Java Model Runner
- Microservice Generation
- Nexus Client
- Python DCAE Model Runner
- Python Model Runner

Design Studio

The Design Studio component repository includes the Composition Engine, TOSCA Model Generator Client, Generic Data Mapper Service, CSV Data Broker, and SQL Data Broker. Additional components are in separate repositories.

- Design Studio
- Proto Viewer ("Probe")
- Runtime Orchestrator ("Model Connector")

Deployment

- Deployment Client
- Kubernetes Client
- Azure Client
- OpenStack Client

Model Onboarding

- Java Client
- Onboading
- Python Client
- R Client

Portal and Marketplace

- Acumos Hippo CMS
- Portal

Supporting Components

Operations, Administration, and Management (OA&M)

- Platform OA&M

System Integration

- System Integration

Example Models

- Face Privacy Filter
- Image Classification
- Image Mood Classifier
- VM Predictor

1.2.1.2 Weekly Builds

Release notes for weekly builds are on the wiki here.

Weekly builds may be unstable and are not recommended for deployment to a production environment.

CHAPTER 2

Portal and Marketplace User Guides

- Portal and Marketplace User Guide
- Portal and Marketplace Publisher Guide
- Portal and Marketplace Admin Guide
- Portal and Marketplace License Admin Guide
- Design Studio User Guide

CHAPTER 3

Model On-Boarding Guides

- Java (Generic, H2o.ai, Spark): Java Model On-Boarding Guide
- Python: Python Model On-Boarding Guide (recommended version for Clio release is 0.8.0)
- R: R Model On-Boarding Guide
- ONNX: ONNX Model On-Boarding Guide
- Pre-dockerized models and model URI: Pre-dockerized Models and models URI On-boarding Guide
- C++: C++ Model On-Boarding guide

CHAPTER 4

Operational User Guides

4.1 START HERE

For those unfamiliar with Acumos and by extension *z2a*, this is a quick intro. If you are here, you may know what Acumos is but you probably don't know:

- what is *z2a*?
- where do I start with *z2a*?

4.1.1 What is *z2a*?

Zero-to-Acumos (*z2a*) is a modular collection of Linux shell scripts that have been assembled to perform a simple set of tasks: install and (where possible) configure Acumos on a Kubernetes (k8s) cluster.

z2a is composed of two (2) distinct process flows; Flow-1 and Flow-2. In each flow scenario, installation of additional Acumos plugins is optional as a follow-on procedure.

4.1.2 What is *z2a* Flow-1?

z2a Flow-1 (default) performs an Acumos installation including:

- end-user environment creation;
- VM Operating System preparation;
- *z2a* dependency installation;
- Kubernetes cluster creation; and,
- deployment of Acumos noncore and core components on a single VM.

z2a Flow-1 is the original *z2a* process flow targeting development/test environments where a Kubernetes cluster is built and Acumos is installed from scratch on a single VM.

NOTE: *z2a* (Flow-1) should not be used as a production environment deployment tool. *z2a* (Flow-1) has been primarily designed for development and/or test environment installations on pre-built VMs. A key component of *z2a* (Flow-1), *kind* - Kubernetes in Docker - is not recommended for production installs or production workloads.

4.1.3 What is *z2a* Flow-2?

z2a Flow-2 performs an Acumos installation including:

- end-user environment creation;
- *z2a* dependency installation;
- deployment of Acumos noncore and core components on an existing Kubernetes cluster.

z2a Flow-2 is a new *z2a* process flow targeting pre-built Kubernetes cluster environments. (i.e. BYOC - Bring Your Own Cluster)

NOTE: *z2a* (Flow-2) can be used as a production environment deployment tool when appropriate preparations are made. *z2a* (Flow-2) has been primarily designed for installation on a pre-built k8s cluster.

NOTE: Provisioning of a k8s cluster is beyond the scope of *z2a*.

4.1.4 Where do I start with *z2a*?

If you just want to start installing Acumos, refer to the *TL;DR* document which provides an abbreviated installation guide for Acumos and Acumos plugins. Please refer to the following documents for additional information:

NOTE: Some of the documents listed below are currently being updated.

CONFIGURATION - Acumos configuration information document
https://docs.acumos.org/en/latest/submodules/system-integration/docs/z2a/configuration.html

HOWTO - Acumos task document
https://docs.acumos.org/en/latest/submodules/system-integration/docs/z2a/how-to.html

INSTALLATION-GUIDE - Acumos installation document
https://docs.acumos.org/en/latest/submodules/system-integration/docs/z2a/installation-guide.html

README-PLUGINS-SETUP - Acumos Plugin Setup guidance
https://docs.acumos.org/en/latest/submodules/system-integration/docs/z2a/readme-proxy.html

README-PROXY - proxy configuration guidance
https://docs.acumos.org/en/latest/submodules/system-integration/docs/z2a/readme-proxy.html

README-VALUES - additional values configuration guidance
https://docs.acumos.org/en/latest/submodules/system-integration/docs/z2a/readme-values.html

START-HERE - brief Acumos introduction document (this document)
https://docs.acumos.org/en/latest/submodules/system-integration/docs/z2a/start-here.html

TL;DR - abbreviated installation guide

https://docs.acumos.org/en/latest/submodules/system-integration/docs/z2a/tl-dr.html

Created 2020/07/16

Last Modified 2020/10/21

4.2 Platform Operations, Administration, and Management (OA&M) User Guide

Operations, Administration and Management/Maintenance are the processes, activities, tools, and standards involved with operating, administering, managing and maintaining any system.

4.2.1 Acumos Elastic Stack for Log Analytics

One of the functions of (OA&M) for the Acumos platform is to collect and correlate log files from the other platform components in order to support debugging, metrics, alarms, etc. for development and operations purposes. These metrics can reveal issues and potential risks so administrators can take corrective action. To this end, the OA&M component has defined a logging standard to be used by all of those components in order to support correlation. OA&M uses the Elasticsearch, Logstack, Kibana stack and Filebeat to collect and centralize logs that are generated via the microservices. This guide that describes how to use the Acumos Elastic Stack (formerly known as the ELK Stack).

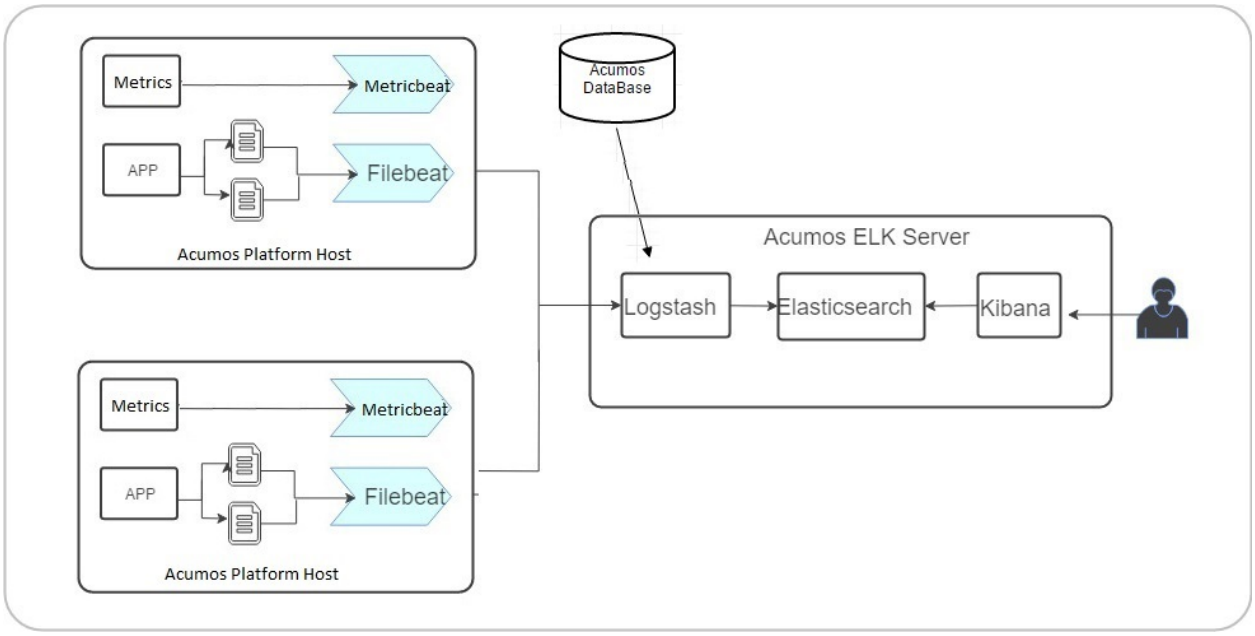
4.2.1.1 Target Users

Acumos Platform super admins

4.2.1.2 Assumptions

All the modules are following the Acumos Logging Guidelines. As per mentioned in Acumos Log Standards Wiki

4.2.1.3 Elastic Stack Architecture



4.2.1.4 Elastic Stack Component Goal

Acumos ELK stack setup has five main components:

- **Elasticsearch**: Elasticsearch is a distributed open source search engine based on Apache Lucene. Acumos Elasticsearch stores all the logs and metrics of Acumos platform host.
- **Logstash**: Logstash is a data pipeline that helps collect, parse, and analyze a large variety of incoming logs generated across Acumos Platform.
- **Kibana**: Web interface for searching and visualizing logs.
- **Filebeat**: Filebeat serves as a log shipping agent, Installed on Acumos platform servers it sends logs to Logstash.
- **Metricbeat**: Installed on Acumos platform servers. it periodically collects the metrics from the Acumos platform host operating system which includes running components information and ships them to elasticsearch. These metrics are used for monitoring.

4.2.1.5 Elastic Stack Component Versions

- elasticsearch 5.5.1
- kibana:5.5.1
- logstash:5.5.1
- filebeat:6.0.1
- metricbeat:6.2.4

4.2.1.6 Elastic Stack Setup

Elastic Stack installation is automated with Docker Compose. Installation is done on a server separate from where Acumos has been installed.

Note We will install components namely Elasticsearch, Logstash and Kibana on a single server, which we will refer to as Acumos ELK stack log collector server. Beat agents namely Filebeat and Metricbeat are installed on Acumos platform host servers.

4.2.1.7 Prerequisites

Docker and Docker Compose installed

4.2.1.8 Steps for first time, clean install

1. Clone the platform-oam repository

```
$ git clone https://gerrit.acumos.org/r/platform-oam
```

2. Create docker volume namely acumos-esdata and acumos-logs if no volumes created earlier.If acumos-esdata and acumos-logs volume already exist on host machine then skip this step.

```
$ docker volume create acumos-esdata
$ docker volume create acumos-logs
```

3. The acumos-elk-env.sh file is the environment file for ELK stack. Update variables ELASTICSEARCH_IMAGE , LOGSTASH_IMAGE , KIBANA_IMAGE with the latest release image.

```
$ cd elk-stack

$ vi acumos-elk-env.sh
```

4. The docker-compose.yml file as well as component directories are located in the elk-stack directory. Edit docker-compose.yml and make changes to these environment variables (ACUMOS_ELK_JDBC_CONNECTION_STRING, ACUMOS_ELK_JDBC_USERNAME, ACUMOS_ELK_JDBC_PASSWORD) to connect to database instance. Edit elasticsearch.yml and make changes to these environment variables ACUMOS_ELK_ELASTICSEARCH_HOST.

```
$ cd elk-stack

$ vi docker-compose.yml
```

5. Starts and attaches to containers for Elasticsearch, Logstash, Kibana

```
$ ./docker-compose-elk.sh up -d
```

6. To stop the running containers without removing them

```
$ ./docker-compose-elk.sh stop
```

4.2.1.9 Steps to upgrade

1. A new version of the base code will need to be pulled from the garret repo. Before that step make a backup of your platform directory.

```
$ git clone https://gerrit.acumos.org/r/platform-oam
```

2. Verify that the volumes previously created are present. If not create the volumes (same as step 2 in clean install):

```
$ docker volume create acumos-esdata
$ docker volume create acumos-logs
```

3. Copy and replace "acumos-elk-env.sh" from your backup (which can be found out in the location as /elk-stack/acumos-elk-env.sh). That will have all the previous environment variables.

Else update the environment variable using below:

```
$ cd elk-stack

$ vi acumos-elk-env.sh
```

4. Copy and replace "docker-compose.yml" (which can be found out in the location as /elk-stack/docker-compose.yml). Which will have all the previous changes.

Else update the environment variable using below:

```
$ cd elk-stack

$ vi docker-compose.yml
```

5. Starts and attaches to containers for Elasticsearch, Logstash, Kibana

```
$ ./docker-compose-elk.sh up -d
```

6. To stop the running containers without removing them

```
$ ./docker-compose-elk.sh stop
```

4.2.1.10 Filebeat setup steps:

Filebeat should be installed as an agent on the servers on which Acumos is running. Add the configuration below to the docker-compose where the Acumos is installed.

```
filebeat:
    container_name: filebeat
    image: <filebeat-image-name>
    volumes:
      - <volume-name>:/filebeat-logs
    environment:
      - LOGSTASH_HOST=<elk-stack-host-hostname>
      - LOGSTASH_PORT=5000
```

4.2.1.11 Metricbeat setup steps:

Metricbeat should be installed as an agent on the servers on which Acumos is running. Add the configuration below to the docker-compose where the Acumos is installed.

```
metricbeat:
    image: <metricbeat-image-name>
    network_mode: host
```

```
    volumes:
    #Mount the docker, filesystem to enable Metricbeat to monitor the host rather␣
↪than the Metricbeat container.
      - /proc:/hostfs/proc:ro
      - /sys/fs/cgroup:/hostfs/sys/fs/cgroup:ro
      - /:/hostfs:ro
      - /var/run:/var/run:rw
      - /var/run/docker.sock:/var/run/docker.sock
    command: metricbeat -e -strict.perms=false -system.hostfs=/hostfs
    environment:
      - SHIPPER_NAME=DOCKY
      - ELASTICSEARCH_HOST=<elk-stack-host-hostname>
      - ELASTICSEARCH_PORT=9200
      - PROCS=.*
      - PERIOD=10s
      - SHIPPER_NAME=super-app
```

4.2.1.12 Adding a New Log

Filebeat docker is a customized image that depends on filebeat.yml, a configuration layer. For adding new log under prospectors of filebeat.yml, need to add log location path as it is in <volume-name>.

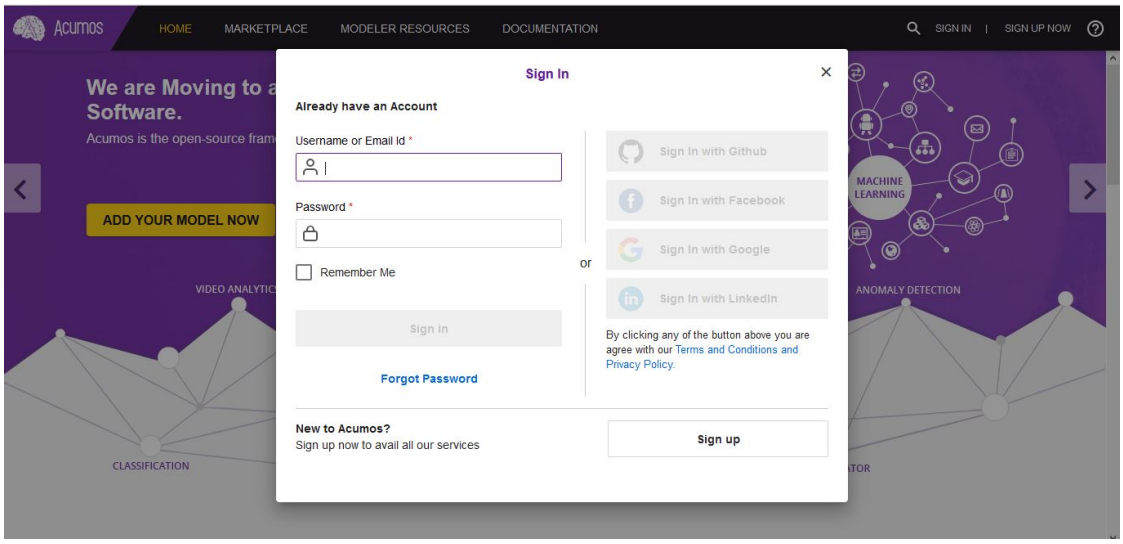
```
filebeat.prospectors:
  - input_type: log
    paths:
      - /filebeat-logs/portal-be/*.log
```

4.2.1.13 Elastic Stack UI Tour

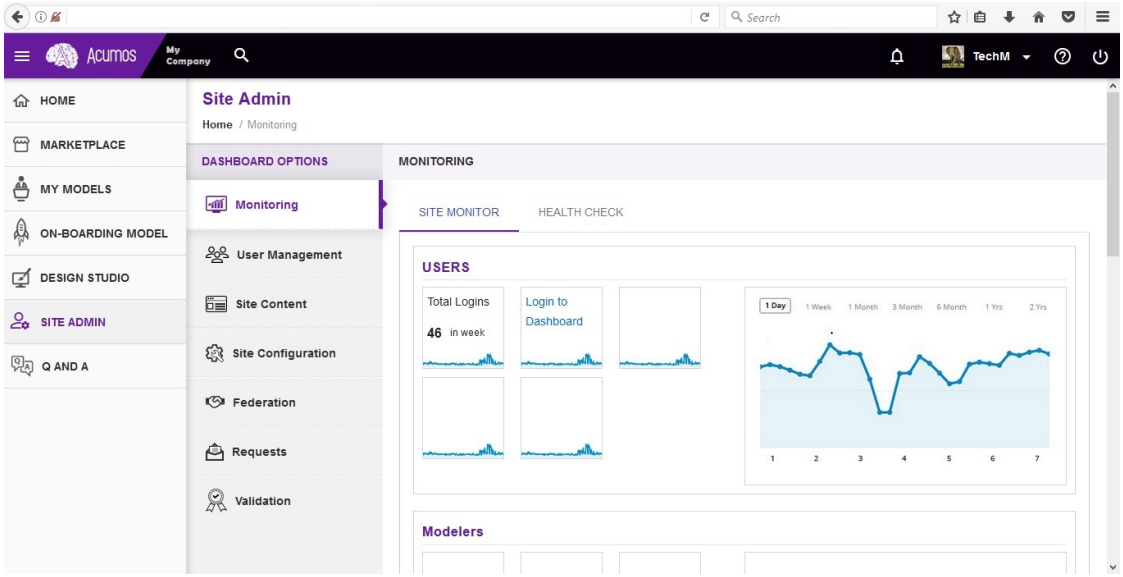
According to the Kibana website, Kibana is an open source analytics and visualization platform designed to work with Elasticsearch. You use Kibana to search, view, and interact with data stored in Elasticsearch indices. You can easily perform advanced data analysis and visualize your data in a variety of charts, tables, and maps. Kibana makes it easy to understand large volumes of data. Its simple, browser-based interface enables you to quickly create queries in real time.

For more details visit Kibana User Guide.

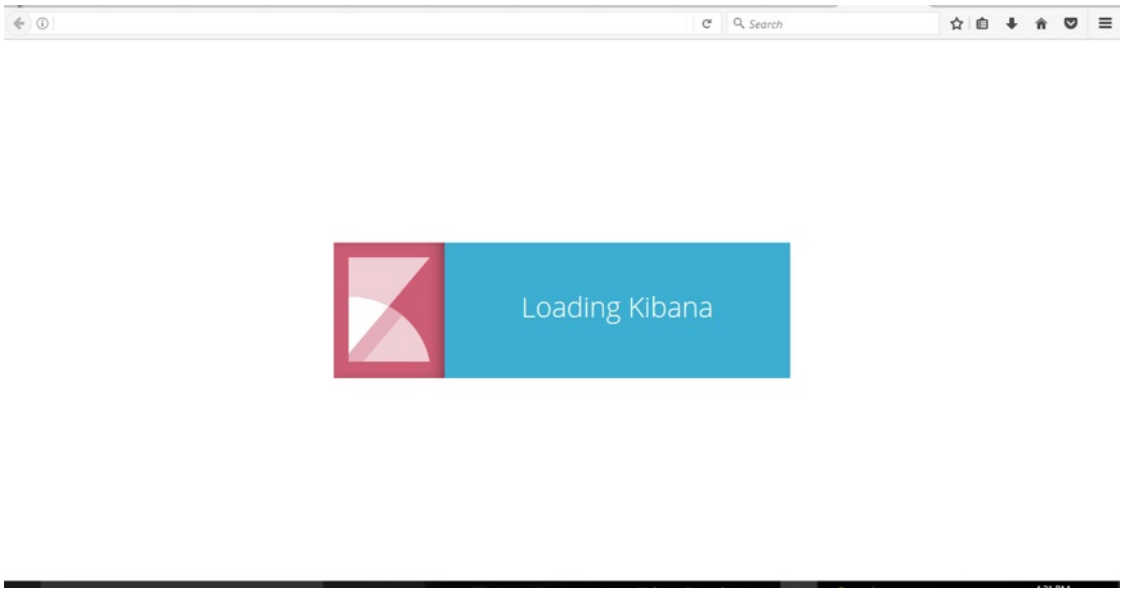
Site admins have access to Elastic Stack's Kibana Dashboard. Login to the dashboard:



Go to SITE ADMIN -> Monitoring and click on **Login to Dashboard** in the USERS section



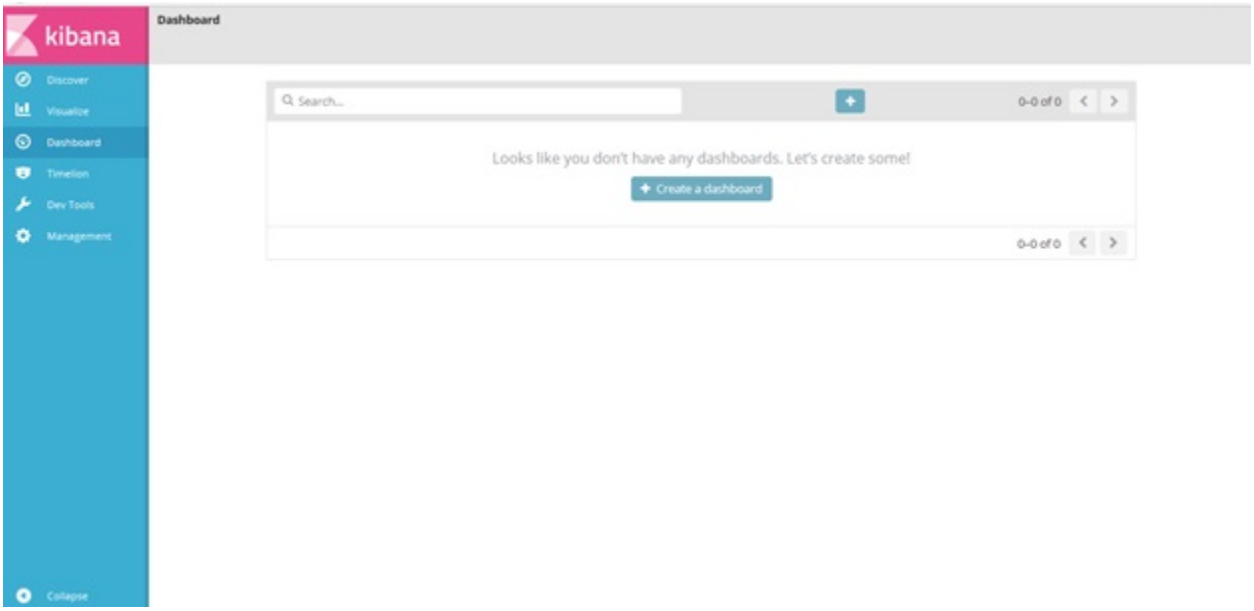
Redirects to Loading Kibana visualization platform



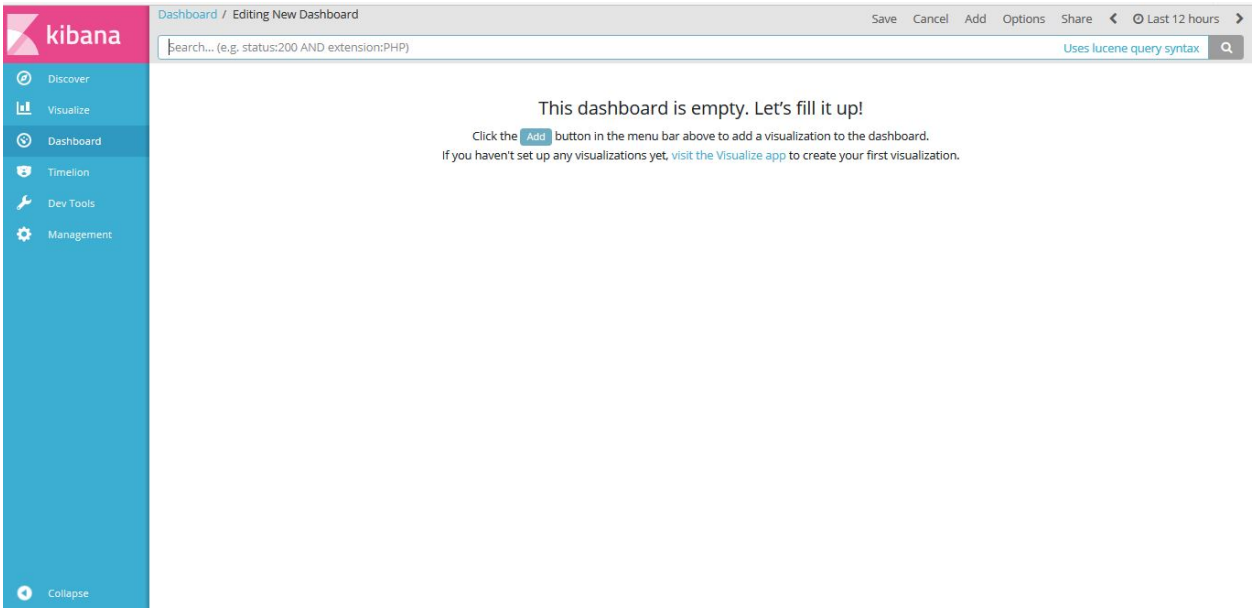
4.2.1.14 Acumos Kibana Dashboard Creation

The Kibana dashboard is used to view all the saved Visualizations.

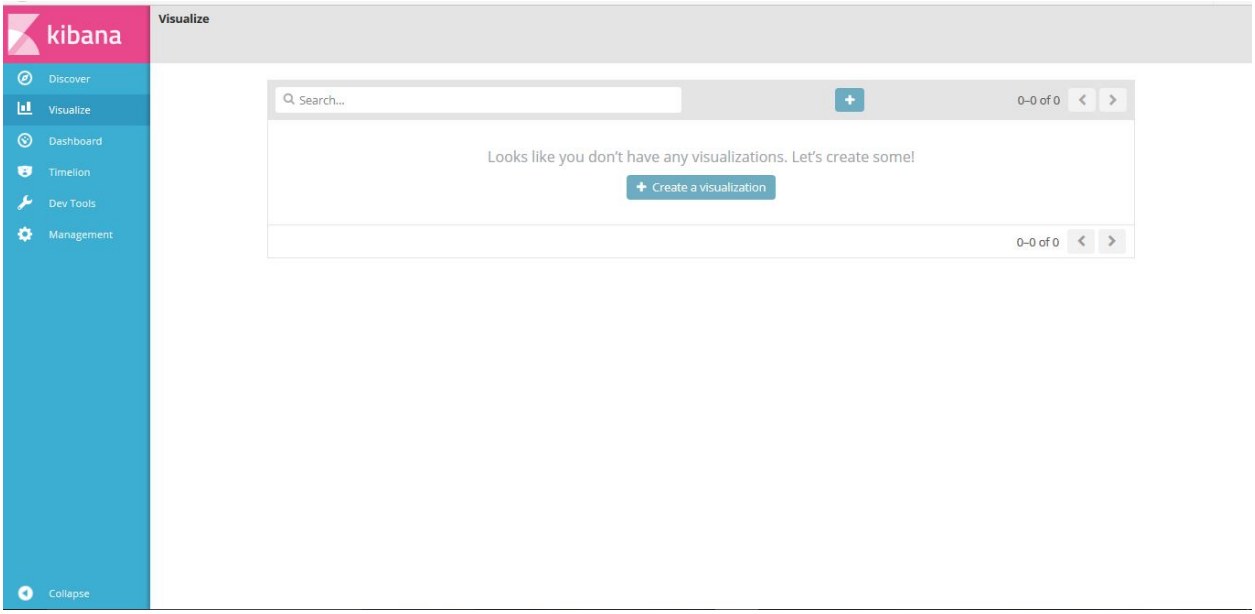
To create dashboard click on Create a dashboard or On plus sign show in the search bar.



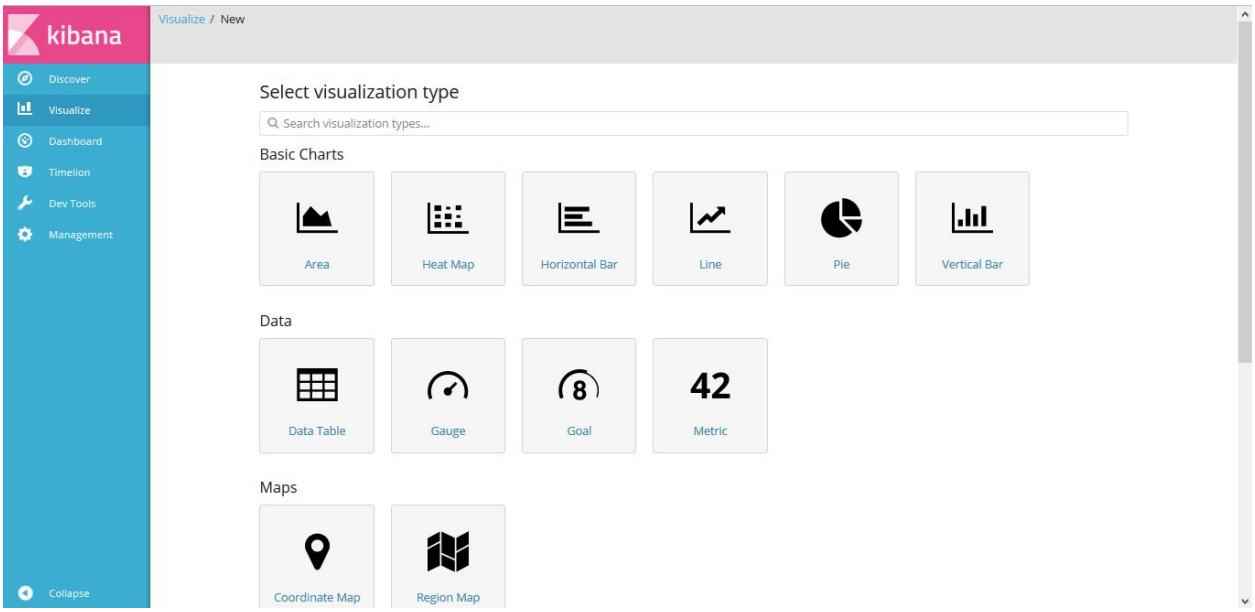
click on Visit the Visualize app



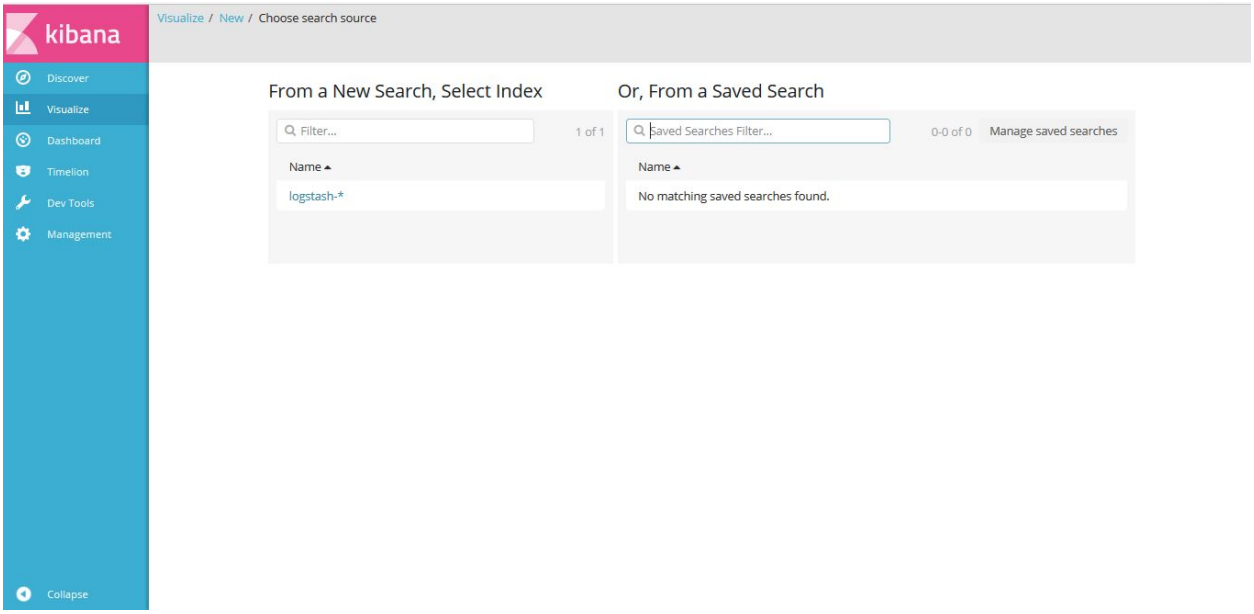
click on "Create a visualization" or "+"(i.e Plus sign) show in the search bar.



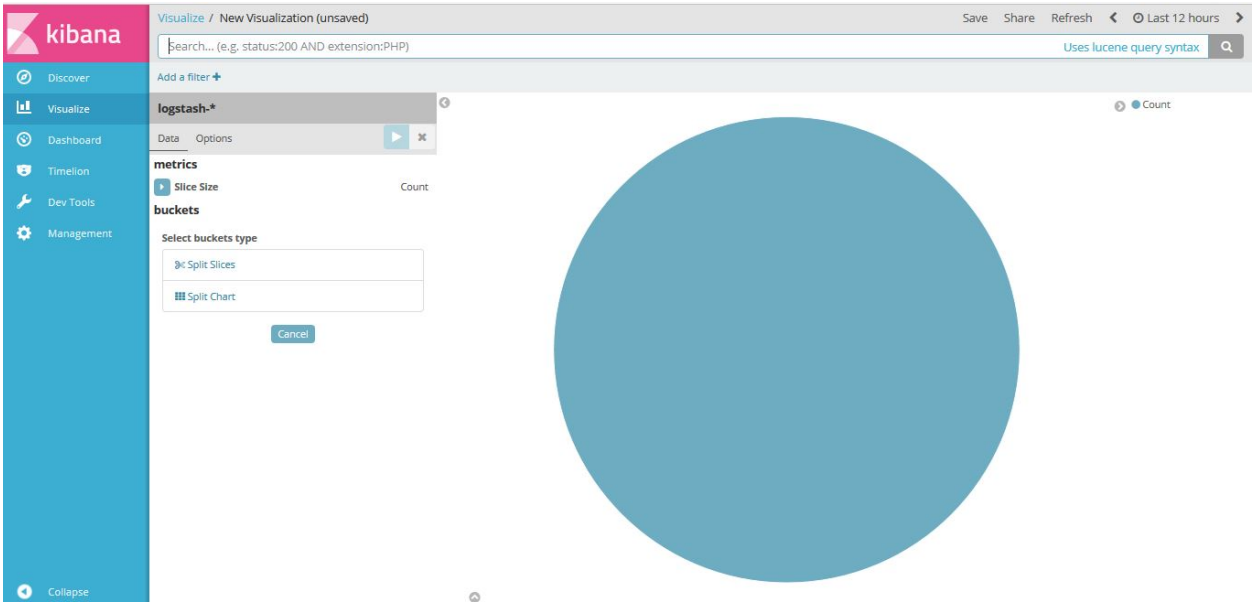
Select visualization type. For example click on "Pie".



Choose search source as `logstash-*`

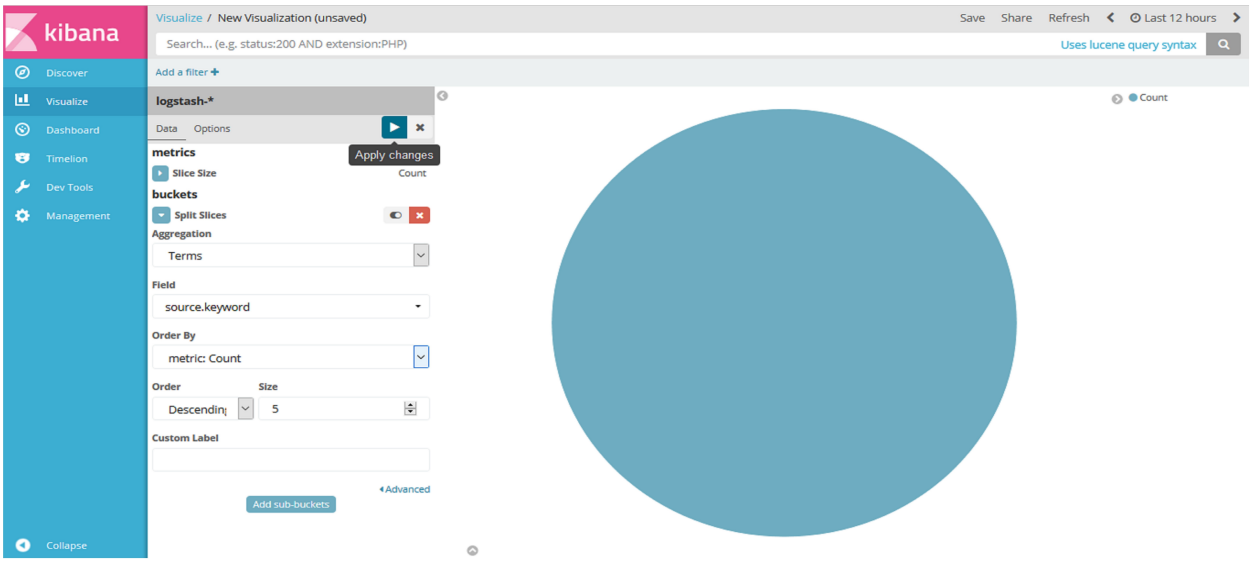


Click on Split Slices

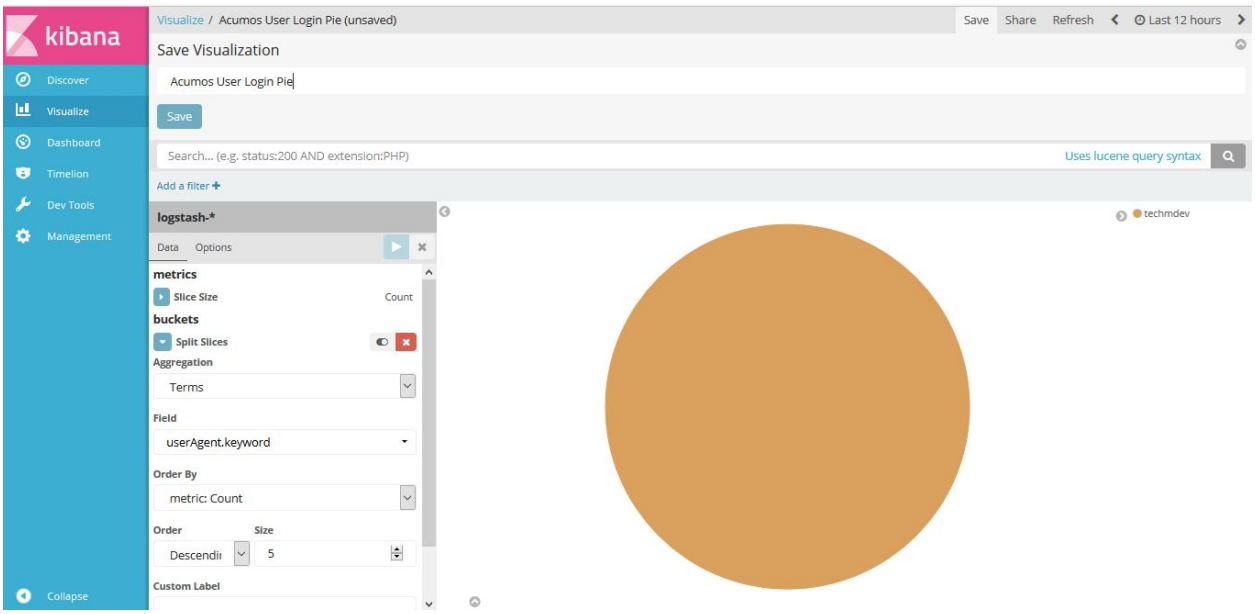


Select Aggregation as “Terms” and Field as “userAgent.keyword”, Click on “Apply changes”

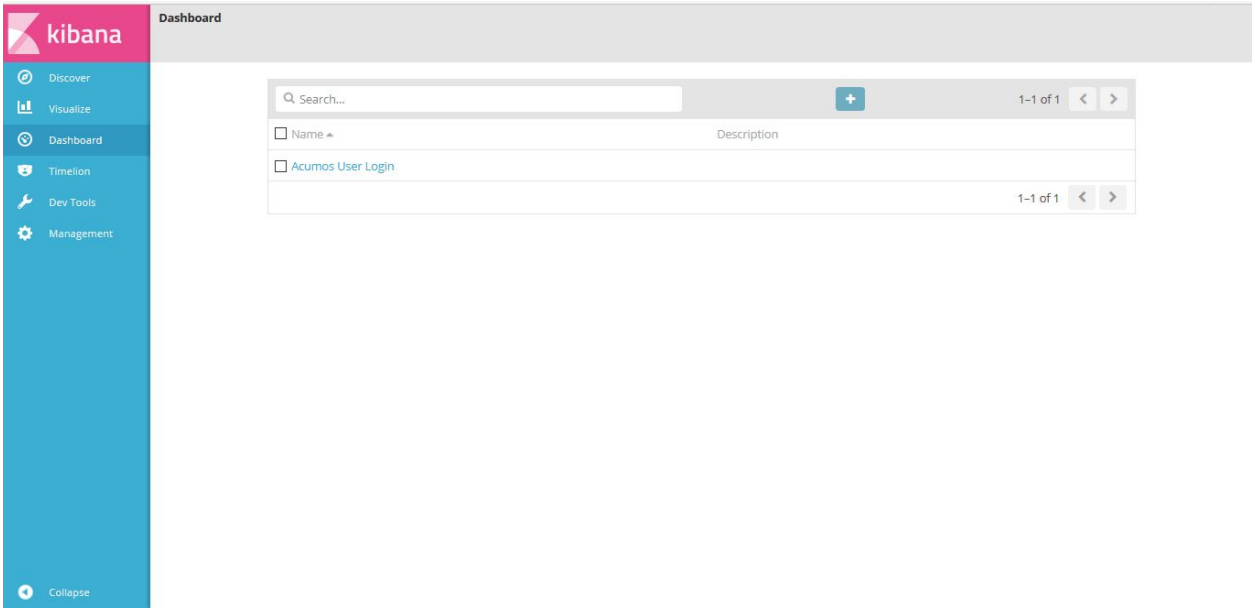
Note: Elasticsearch aggregations are to extract and process your data.



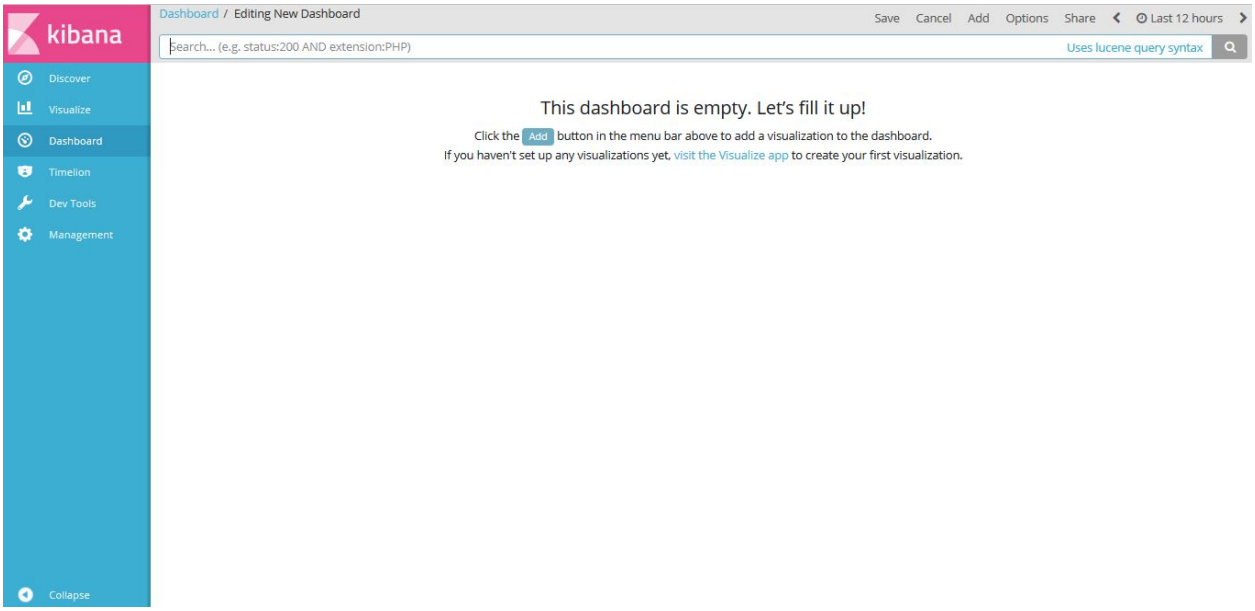
To save this chart click on “Save”, Enter a name appropriate name. For example “Acumos User Login”.



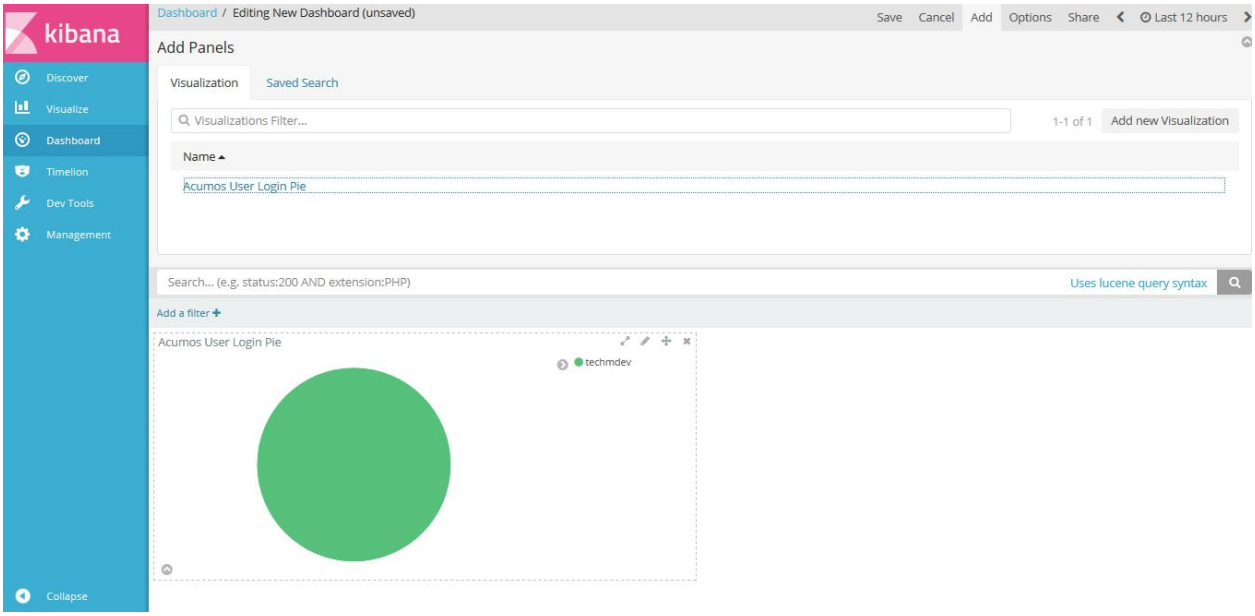
Click on “Dashboard”, On the below screen visualization namely “Acumos User Login” is appearing. For select this visualization click on “+” (i.e. plus sign) show in the search bar.



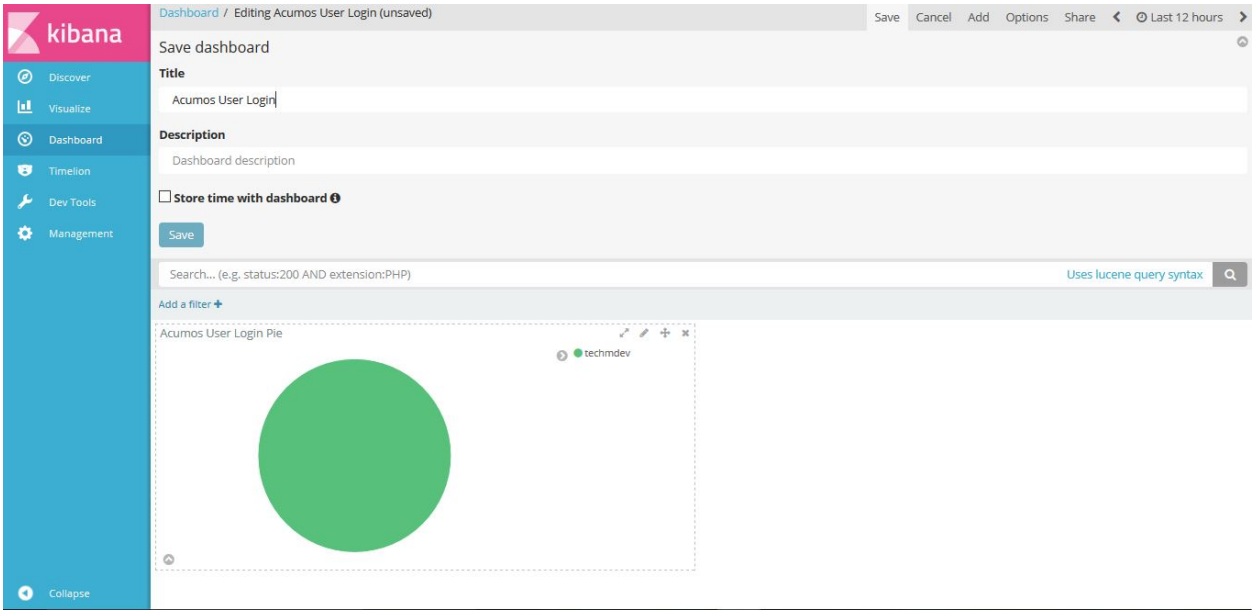
Click on “Add” button, to add the visualization.



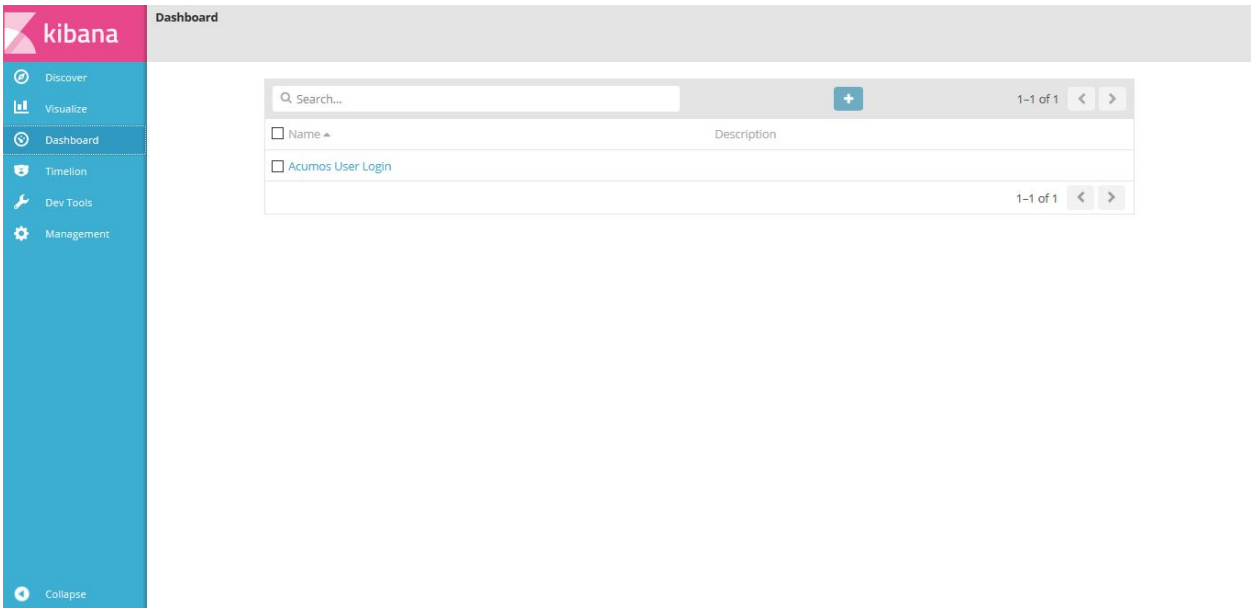
Select the visualization for example here we have visualization namely “Acumos User Login”.



Click on “Save” button. Enter a name appropriate name. For example “Acumos User Login”.

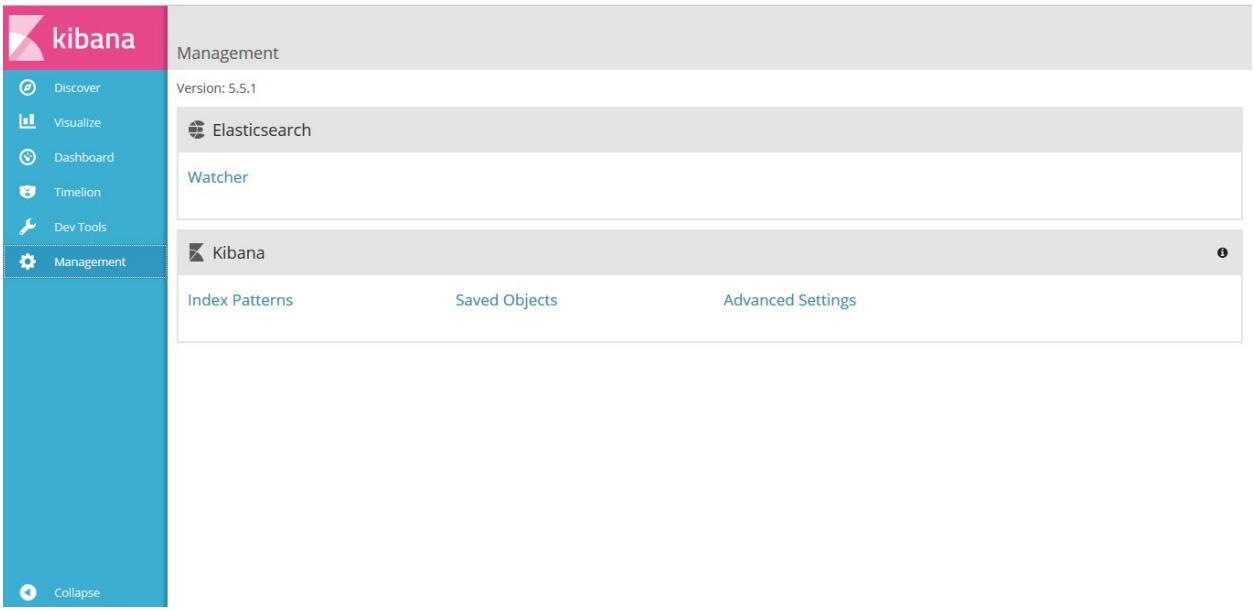


Click on “Dashboard”, On the below screen created dashboard can be viewed namely “Acumos User Login”.

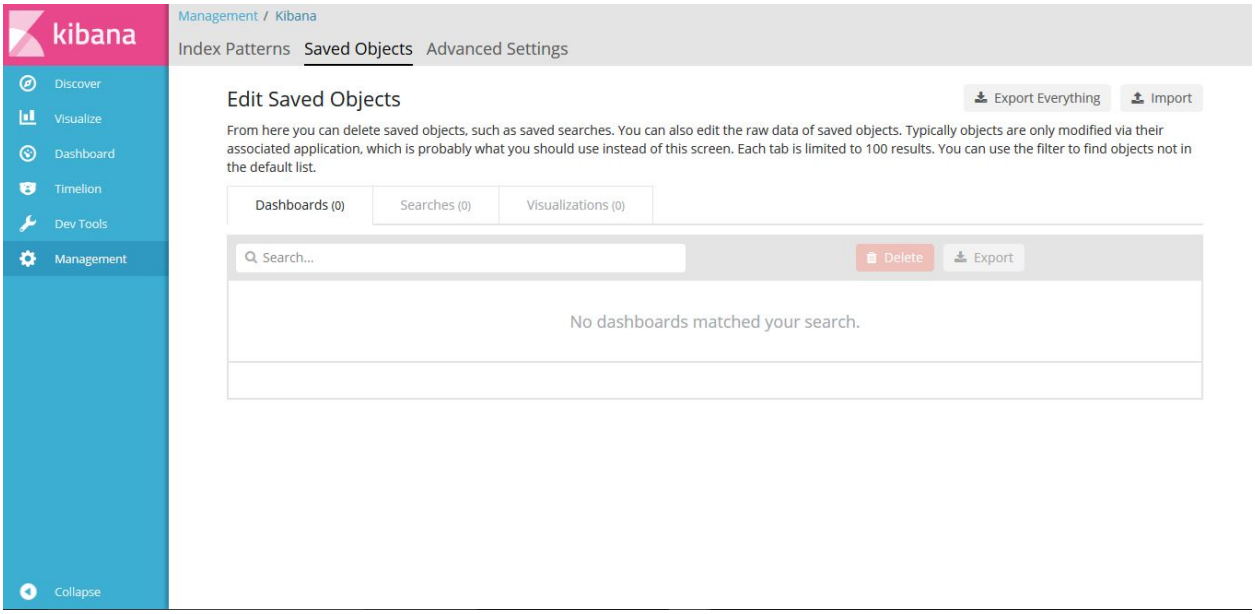


4.2.1.15 Acumos Kibana Dashboard Save

Click on “Management”, On the below screen click on save object.



Click on “Export Everything” to export the dashboard and “Import” to import the saved dashboard.



Note: export/import document should be in JSON format.

An example JSON file that can be used to import a Dashboard is available in the platform-oam repo, elk-stack directory.

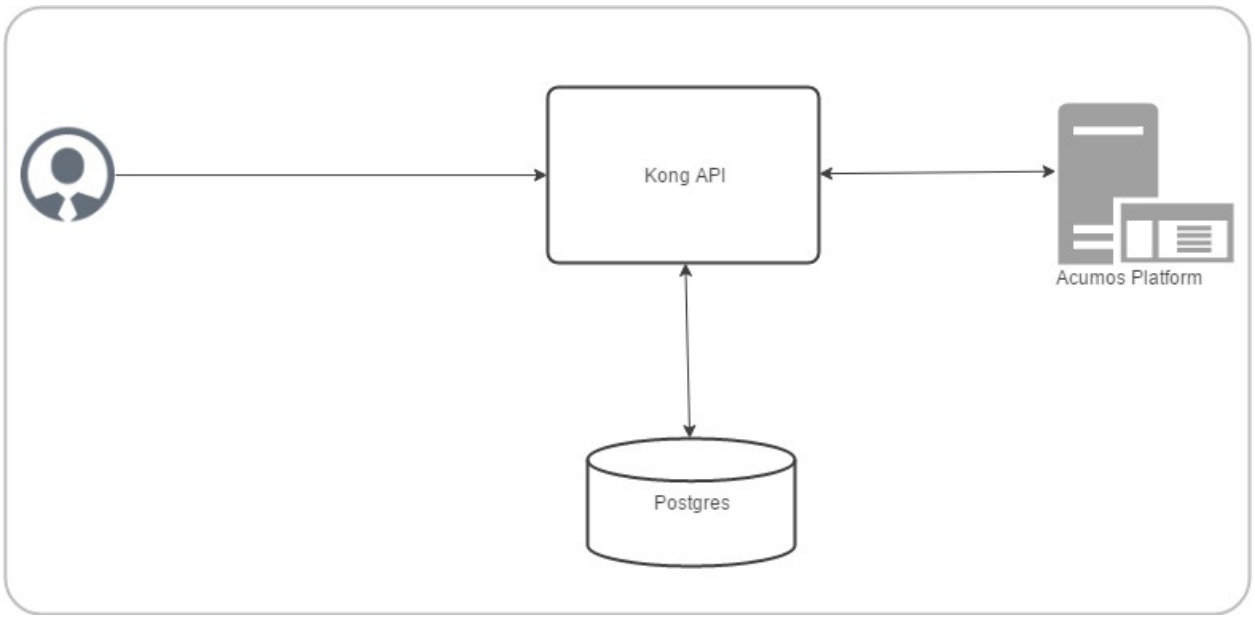
4.3 System Integration User Guide

4.3.1 Acumos API Management with Kong

According to the Kong website, Kong is a scalable, open source API Layer/Gateway/Middleware. The Acumos Platform uses Kong as a reverse proxy server. SSL certificates are installed on the Kong server so each containerized app doesn't have to install its own certs. Kong is highly configurable. Browse the Kong documentation for a detailed description and user guides.

Kong API helps in reducing the rewriting of the same piece of code again and again for SSL certificates configuration in order to make the API secure. Now we don't need to do any coding/configuration work in API anymore.

Backend Architecture



Note: All the configuration data sent through the Admin API is stored in Kong's data store. Kong is capable of supporting both Postgres and Cassandra as storage backend. We have chosen Postgres.

4.3.1.1 Kong API component versions

- postgres:9.4
- kong:0.11.0

4.3.1.2 Acumos Kong API setup

Kong API completely containerized solution is automated with docker compose. It installed with its own docker-compose file.

In dockers-compose definition, there are three services:

- kong-database
- kong-migration
- kong

Kong uses an external datastore to store its configuration such as registered APIs, Consumers and Plugins. The entire configuration data is stored in Kong's data store. The kong-migration service is used to create the objects in the kong-database. This bootstrap functionality is not provided by kong service, so kong-migration service run once inside the container.

By default Kong listens on the following ports:

:8000 on which Kong listens for incoming HTTP traffic from your clients, and forwards it to your upstream services.

:8443 on which Kong listens for incoming HTTPS traffic. This port has a similar behavior as the :8000 port, except that it expects HTTPS traffic only. This port can be disabled via the configuration file.

:8001 on which the Admin API used to configure Kong listens.

:8444 on which the Admin API listens for HTTPS traffic.

Acumos Kong is running on port

:7000 on which Acumos Kong listens for incoming HTTP traffic from your clients, and forwards it to your upstream services.

:443 on which Acumos Kong listens for incoming HTTPS traffic. This port has a similar behavior as the :7000 port, except that it expects HTTPS traffic only. This port can be disabled via the configuration file.

:7001 on which the Admin API used to configure Acumos Kong listens.

:7004 on which the Admin API listens for HTTPS traffic.

Note: Acumos Kong API docker-compose.yml and shell script can be run before or after the main docker-compose. Ensure before access the service URL via acumos Kong API all the services which we are going to access should be up and running.

4.3.1.3 Prerequisites

Docker and Docker Compose installed

4.3.1.4 Steps

1. Clone the system-integration repository

```
$ git clone https://gerrit.acumos.org/r/system-integration
```

2. Builds, (re)creates, starts, and attaches to containers for kong, postgres.

```
$ ./docker-compose-kong.sh up -d
```

3. To stop the running containers without removing them

```
$ ./docker-compose-kong.sh stop
```

4.3.1.5 Steps to create self signed in certificate

1. Create the private server key

```
openssl genrsa -des3 -out server.key 2048
```

2. Now we create a certificate signing request

```
openssl req -new -key server.key -out server.csr -sha256
```

3. Remove the passphrase

```
cp server.key server.key.org
```

```
openssl rsa -in server.key.org -out server.key
```

4. Signing the SSL certificate

```
openssl x509 -req -in server.csr -signkey server.key -out server.crt -sha256
```

4.3.1.6 Acumos API configuration

Please update the configuration settings in "secure-acumos-api.sh" script to match your environment:

1. Copy your host certificate and key under acumos-kong-api "certs" directory
2. Change the values of placeholders below before running the script

```
export ACUMOS_KONG_CERTIFICATE_PATH=./certs

export ACUMOS_CRT=localhost.csr

export ACUMOS_KEY=localhost.key

export ACUMOS_HOST_NAME=<your hostname>

export ACUMOS_HOME_PAGE_PORT=8085

export ACUMOS_CCDS_PORT=8003

export ACUMOS_ONBOARDING_PORT=8090
```

Run the "secure-acumos-api.sh" script, Please ensure that Acumos Kong API container is up.

```
./secure-acumos-api.sh
```

4.3.1.7 Expose new service:

Use the Admin API port 7001 to configure Kong. Acumos standard sample to expose the service is present in shell script:

```
./secure-acumos-api.sh
```

For more details visit Kong Admin API documentation,

4.3.1.8 Deployment of Acumos platform under Azure-K8s

```
Introduction
```

This user guide describes how to deploy Acumos platform using Kubernetes an open-source container-orchestration system for automating deployment, scaling and management of containerized applications under public cloud Azure.

```
What's included in the acumosk8s public cloud Azure
```

In system-integration repo folder acumosk8s-public-cloud/azure:

- deployments/all_start_stop.sh: the main script that kicks off the deployment, to setup pods Acumos, elk, docker, kong, nexus, proxy and mariadb under a kubernetes environment.
- acumos-kubectl.env: environment setup file that is customized as new environment parameters get generated (e.g. passwords). Used by various scripts in this toolset, to set shell environment variables that they need.
- deployments/: kubernetes deployment templates for all system components.
- services/all_start_stop.sh: the script that gets all the services started, to setup service for Acumos, elk, docker, kong, nexus, proxy, mariadb and federation under a kubernetes environment.
- services/: kubernetes service templates for all system components.
- configmap/: kubernetes configmap templates for ELK stack.
- volumeclaim/all_start_stop.sh: the script that creates persistent volume claim for mariadb, nexus ,output, web onboarding, federation certificates and acumos logs.

```
Release Scope
```

Current Release (Athena)

The Athena release includes these capabilities that have been implemented/tested:

- Multi-Node deployment of the Acumos platform under kubernetes.
- deployment with a new Acumos database or redeployment with a current database and components compatible with that database version.
- Component services under kubernetes as named below (deployed as one pod-based service a.k.a acumos):
  - core components of the Acumos platform
    - Portal Marketplace: acumos
    - Hippo CMS: acumos
    - Solution Onboarding: acumos
    - Design Studio Composition Engine: acumos
    - Federation Gateway: federation-service
    - Azure Client: acumos
    - Common Data Service: acumos
    - Filebeat: acumos
    - Elasticsearch: elasticsearch
    - Logstash: logstash-service
    - Kibana: kibana-service
  - external/dependency components
    - docker engine/API: acumos-docker-service under kubernetes.
    - MariaDB: mariadb running as acumos-mysql service under kubernetes.
    - Kong proxy: running as acumos-kong-proxy, acumos-postgres service under kubernetes.
    - Nexus: running as acumos-nexus-service under kubernetes.

* Proxy: running as acumos-proxy under kubernetes.

4.3.1.9 Future Releases

Future releases may include these new features:

- Scaling up, monitoring health tool.

4.3.1.10 Prerequisites

Setup of Kubernetes cluster in Azure and kubectl, the Kubernetes command-line client ,Tiller to install using helm charts.

4.3.1.11 Step-by-Step Guide

1. Clone the system-integration repository.

```
$ git clone https://gerrit.acumos.org/r/system-integration
```

2. Change directory to acumosk8s-public-cloud/azure

```
$ cd  acumosk8s-public-cloud/azure
```

3. Edit acumos-kubectl.env file to make changes related to latest assembly , database connection , credentials ,etc.

```
$ vi acumos-kubectl.env
```

4. Use kubectl create command on kubernetes client machine to create a namespace.

```
$ kubectl create namespace <namespace name>
Example: kubectl create namespace acumos-ns01
```

5. Change directory to acumosk8s-public-cloud/azure/volumeclaim to create persistent volume claim (pvc).

```
$ cd  acumosk8s-public-cloud/azure/volumeclaim
```

6. Edit acumos-volumeclaim.sh file and update variable ENV_FILE for absolute path of acumos-kubectl.env file.

```
$ vi acumos-volumeclaim.sh
```

7. Run all-start-stop.sh script under volumeclaim directory. This will create pvc for certs, nexus, output, acumos logs, webonboarding and mariadb.

```
$ ./all-start-stop.sh create
```

8. This step needs to be executed only if all the pvc created earlier needs to be deleted.This will delete all the pvc created under the given namespace.

```
$ ./all-start-stop.sh delete
```

9. If each volumeclaim need to be created individually then skip step 7 and use below command.

```
$ ./acumos-volumeclaim.sh <name of volumeclaim .yaml file> create
Example: ./acumos-volumeclaim.sh acumos-volumeclaim.yaml create
```

10. Create a secret file for acumos that contains base64 encoding to pull docker image from nexus repo.

```
$ log "Create k8s secret for docker image pulling from nexus repo"
  b64=$(cat ~/.docker/config.json | base64 -w 0)
  cat <<EOF >acumos-secret.yaml
  apiVersion: v1
  kind: Secret
  metadata:
    name: acumos-secret
    namespace: acumos-ns01
  data:
    .dockerconfigjson: $b64
  type: kubernetes.io/dockerconfigjson
  EOF
```

11. Create configmap for ELK stack.

```
$ cd  acumosk8s-public-cloud/azure/configmap
$ ./acumos-configmap.sh <name of config.yaml file> create
Example: ./acumos-configmap.sh es-config.yaml create
      ./acumos-configmap.sh logstash-config.yaml create
```

12. Change directory to acumosk8s-public-cloud/azure/deployments

```
$ cd  acumosk8s-public-cloud/azure/deployments
```

13. Edit acumos-deployment.sh file and update variable ENV_FILE for absolute path of acumos-kubectl.env file.

```
$ vi acumos-deployment.sh
```

14. Run all-start-stop.sh script under deployments directory. This will create kubernetes deployment for mariadb ,kong, elk, acumos (containing all components), nexus, docker and proxy.

```
$ ./all-start-stop.sh create
```

15. This step needs to be executed only if all the deployment.yaml created earlier needs to be deleted.This will delete kubernetes deployment for mariadb ,kong, elk, acumos (containing all components), nexus, docker and proxy created under the given namespace.

```
$ ./all-start-stop.sh delete
```

16. If each deployment need to be created individually then skip step 14 and use below command.

```
$ ./acumos-deployment.sh <name of deployment.yaml file> create
Example: ./acumos-deployment.sh acumos-deployment.yaml create
```

17. Change directory to acumosk8s-public-cloud/azure/services

```
$ cd  acumosk8s-public-cloud/azure/services
```

18. Edit acumos-service.sh file and update variable ENV_FILE for absolute path of acumos-kubectl.env file.

```
$ vi acumos-service.sh
```

19. Run all-start-stop.sh script under services directory. This will create kubernetes service for mariadb ,kong, elk, acumos (containing all components), nexus, docker ,federation and proxy. After services are up and

running we need to map external endpoints generated for kibana-service , federation-service and acumos-nexus-service to FQDN in azure e.g. IP 40.117.115.236 generated for kibana is mapped to acumosk8s-log.eastus.cloudapp.azure.com

```
$ ./all-start-stop.sh create
```

20. This step needs to be executed only if all the services.yaml created earlier needs to be deleted.This will delete kubernetes services for mariadb ,kong, elk, acumos (containing all components), nexus, docker , federation and proxy created under the given namespace.

```
$ ./all-start-stop.sh delete
```

21. If each service need to be created individually then skip step 19 and use below command.

```
$ ./acumos-service.sh <name of service.yaml file> create
Example: ./acumos-service.sh acumos-service.yaml create
```

22. Create a certs directory in kubernetes client machine and generate files acumos-k8s.cert , acumos-k8s.key , acumos-k8s.pkcs12 and acumosTrustStore.jks

23. Create certificate and run ./create-certs.sh , this shell file includes below line

```
openssl req -x509 -newkey rsa:4096 -keyout acumos-k8s.key -out acumos-k8s.cert -days␣
↪365
```

24. Install certificates and run ./install-certificates.sh that includes below line. acumosk8s.eastus.cloudapp.azure.com is the FQDN and 8001 is port no that is exposed.

```
curl -i -X POST http://acumosk8s.eastus.cloudapp.azure.com:8001/certificates \
-F "cert=acumos-k8s.cert" \
-F "key=acumos-k8s.key" \
-F "snis=acumosk8s.eastus.cloudapp.azure.com,localhost"
```

25. Add to certificates run ./add-to-cacert.sh , this shell file includes below line.

```

```

/usr/lib/jvm/java-8-oracle/bin/keytool -import -noprompt -keystore acumosTrustStore.jks -storepass changeit -alias acumos-k8s -file acumos-k8s.pem

26. Generate pkcs12.sh file run ./generate-pkcs12.sh , this file includes below code.

```
#! /bin/bash
CERT_DIR=/path-to-directory/acumos-k8s/certs
CERT_FILE=acumos-k8s.cert
CERT_KEY=acumos-k8s.key
PKCS12_FILE=acumos-k8s.pkcs12
openssl pkcs12 -export -nokeys -in ${CERT_DIR}/${CERT_FILE} -out ${CERT_DIR}/${PKCS12_
↪FILE}
```

27. Give read and execute access to .pkcs12 and .jks file by making use of below command

```
chmod 755 acumosTrustStore.jks
chmod 755 acumos-k8s.pkcs12
```

28. Copy acumosTrustStore.jks and acumos-k8s.pkcs12 to volume mounted for federation gateway container. Make use of below commands. In our case /path-to-directory/acumos-k8s/certs/acumos-k8s.pkcs12 is the path where file is located under K8 , acumos-ns01 is the namespace created and acumos-1353575208-c235g is the pod name that

contains all the containers including federation-gateway. /app/certs is the mount directory for federation-gateway container

```
kubectl cp /path-to-directory/acumos-k8s/certs/acumos-k8s.pkcs12 acumos-ns01/acumos-
↪1353575208-c235g:/app/certs/ -c federation-gateway

kubectl cp /path-to-directory/acumos-k8s/certs/acumosTrustStore.jks acumos-ns01/
↪acumos-1353575208-c235g:/app/certs/ -c federation-gateway
```

29. After copying .pkcs12 and .jks file restart the federation-gateway pod
30. Run secure-acumos-api-internal.sh file on K8. You need to change few configuration listed below based on your environment in this file

```
export ACUMOS_KONG_API_HOST_NAME=acumosk8s.eastus.cloudapp.azure.com
export ACUMOS_KONG_API_HOST_SNIS=acumosk8s.eastus.cloudapp.azure.com
export ACUMOS_KONG_API_PORT=8001
export ACUMOS_KONG_CERTIFICATE_PATH=/path-to-directory/acumos-k8s/certs
export ACUMOS_CRT=acumos-k8s.cert
export ACUMOS_KEY=acumos-k8s.key
export ACUMOS_HOST_NAME=acumos.acumos-ns01
export ACUMOS_NEXUS_HOST_NAME=acumos-nexus-service.acumos-ns01
export ACUMOS_HOME_PAGE_PORT=8085
export ACUMOS_ONBOARDING_PORT=8090
export ACUMOS_CMS_PORT=9080
export ACUMOS_NEXUS_PORT=8001
```

31. Follow below steps to set up CMS.
    - Login to the Hippo CMS console as "admin/admin", at http://<hostname>:<ACUMOS_CMS_PORT>/cms/console, where ACUMOS_CMS_PORT is per acumos-kubectl.env; for the default, the address is acumosk8s.eastus.cloudapp.azure.com:9080/cms/console
    - On the left, click the + at `hst:hst` and then also at `hst:hosts`. Click the + at the `dev-env` entry, and the same for the nodes as they appear: `com, azure, cloudapp, eastus`
    - Right-click on the "acumos-dev1-vm01-core" entry and select "Move node".
    - In the `Move Node` dialog, select the `dev-env` node, enter "<hostname>" at `To`, and click``OK``. Default hostname is acumosk8s
    - When the dialog closes, you should see your node renamed and moved under `dev-env`. You may also want to save your changes by pressing the `Write changes to repository` button in the upper right.
    - With the "<hostname>" node selected, click `Add Property` from the toolbar.
    - In the `Add a new Property` dialog, place your cursor in the `Name` field and then select `hst:schemeagnostic`. click `OK`.
    - Make sure the hostname is selected on the left. Then select the check box under the new attribute. This attribute is essential, as internal to the Acumos platform the Hippo CMS service is accessed via HTTP, but externally, user web browsers access the Acumos portal via HTTPS. Also click the `Write changes to repository` button on the upper right.
    - Delete the superfluous node. Right-click the `com` node, select `Delete node`.
    - Select the `Save immediately` check box and click `OK`
32. Follow below step to set up MariaDB

Run below command to connect to acumos-mysql container.

```
kubectl -n acumos-ns01 exec -it <acumos-mysql-pod name> /bin/sh
```

Connect to Mariadb.

```
mysql -u root -p <password>
```

Execute below scripts to create acumos and acumos cms database. e.g we have used CDS but it need to be same mentioned in env file.

```
drop database if exists CDS;
create database CDS;
create user 'CDS_USER'@'localhost' identified by 'CDS_PASS';
grant all on CDS.* to 'CDS_USER'@'localhost';
create user 'CCDS_USER'@'%' identified by 'CDS_PASS';
grant all on CDS.* to 'CDS_USER'@'%';
```

```
drop database if exists acumos_CMS;
create database acumos_CMS;
create user 'CMS_USER'@'localhost' identified by 'CMS_PASS';
grant all on acumos_CMS.* to 'CMS_USER'@'localhost';
create user 'CMS_USER'@'%' identified by 'CMS_PASS';
grant all on acumos_CMS.* to 'CMS_USER'@'%';
```

Execute the DDL and DML scripts for any database version that needs to be configured.

4.3.1.12 Set up using Helm Charts

1. Clone the system-integration repository.

```
$ git clone https://gerrit.acumos.org/r/system-integration
```

2. Change directory to acumosk8s-public-cloud/azure/HELM

```
$ cd  acumosk8s-public-cloud/azure/HELM
```

3. Create a secret file for acumos that contains base64 encoding to pull docker image from nexus repo.

```
$ log "Create k8s secret for docker image pulling from nexus repo"
  b64=$(cat ~/.docker/config.json | base64 -w 0)
  cat <<EOF >acumos-secret.yaml
  apiVersion: v1
  kind: Secret
  metadata:
    name: acumos-secret
    namespace: <namespace name>
  data:
    .dockerconfigjson: $b64
  type: kubernetes.io/dockerconfigjson
  EOF
```

4. Use below helm install command on kubernetes client machine to install helm chart for non core components like nexus, mariadb ,etc and elk stack.

```
$ helm install k8-noncore-chart
$ helm install k8-elk-chart
```

5. Follow below step to set up MariaDB

Run below command to connect to acumos-mysql container.

```
kubectl -n <namespace_name> exec -it <acumos-mysql-pod name> /bin/sh
```

Connect to Mariadb.

```
mysql -u root -p <password>
```

Execute below scripts to create acumos database. e.g we have used CDS but it need to be same mentioned in env file.

```
drop database if exists CDS;
create database CDS;
create user 'CDS_USER'@'localhost' identified by 'CDS_PASS';
grant all on CDS.* to 'CDS_USER'@'localhost';
create user 'CDS_USER'@'%' identified by 'CDS_PASS';
grant all on CDS.* to 'CDS_USER'@'%';
```

Execute the DDL and DML scripts for any database version that needs to be configured.This is available in common data service gerrit repo.

6. Edit values.yaml file inside k8-acumos-chart to make changes related to latest assembly , database connection , credentials ,onboarding-cli service,etc.

```
$ cd k8-acumos-chart
$ vi values.yaml
```

7. Use below helm install command on kubernetes client machine to install helm chart for acumos core components like portal- fe , portal-be, onboarding,etc.

```
$ helm install k8-acumos-chart
```

8. To view and delete the helm charts installed.

```
$ helm list
$ helm delete <chart name>
```

9. Generate certificates using above mentioned steps. Copy acumosTrustStore.jks and acumos-k8s.pkcs12 to volume mounted for federation gateway container. Make use of below commands. In our case /path-to-directory/acumos-k8s/certs/acumos-k8s.pkcs12 is the path where file is located under K8 , acumos-ns01 is the namespace created and acumos-1353575208-c235g is the pod name that contains all the containers including federation-gateway. /app/certs is the mount directory for federation-gateway container

```
kubectl cp /path-to-directory/acumos-k8s/certs/acumos-k8s.pkcs12 acumos-ns01/acumos-
↪1353575208-c235g:/app/certs/ -c federation-gateway

kubectl cp /path-to-directory/acumos-k8s/certs/acumosTrustStore.jks acumos-ns01/
↪acumos-1353575208-c235g:/app/certs/ -c federation-gateway
```

10. After copying .pkcs12 and .jks file restart the federation-gateway pod.

11. To redeploy core components based on weekly assembly use chart k8-acumos-redeploy-chart.

```
$ helm install k8-acumos-redeploy-chart
```

12. Run secure-acumos-api-internal.sh file on K8. You need to change few configuration listed below based on your environment in this file

```
export ACUMOS_KONG_API_HOST_NAME=acumosk8s.FQDN

export ACUMOS_KONG_API_HOST_SNIS=acumosk8s.FQDN

export ACUMOS_KONG_API_PORT=8001

export ACUMOS_KONG_CERTIFICATE_PATH=/path-to-directory/certificates-is-stored

export ACUMOS_CRT=acumos-k8s.cert

export ACUMOS_KEY=acumos-k8s.key

export ACUMOS_HOST_NAME=<acumos service name>.<namespace>

export ACUMOS_NEXUS_HOST_NAME=acumos-nexus-service.<namespace>

export ACUMOS_HOME_PAGE_PORT=8085

export ACUMOS_ONBOARDING_PORT=8090

export ACUMOS_NEXUS_PORT=8001
```

4.3.1.13 Monitoring resource utilization in kubernetes using Prometheus and Grafana

1. Create a folder called prometheus. Here we will create all our monitoring resources.Create a file called prometheus/namespace.yml with the content.

```
kind: Namespace
apiVersion: v1
metadata:
  name: prometheus
```

2. Apply & Test the namespace exists.

```
$ kubectl get namespaces
```

3. Deploy Prometheus into the prometheus namespace.

```
$ helm install stable/prometheus --namespace prometheus --name prometheus
```

4. We can confirm by checking that the pods are running.

```
$ kubectl get pods -n prometheus
```

5. Deploy Grafana into the prometheus namespace.

```
$ helm install stable/grafana --namespace prometheus --name grafana
```

6. Grafana is deployed with a password. Run below command to get the initial password.The username is admin.

```
$ kubectl get secret --namespace prometheus grafana -o jsonpath="{.data.admin-
↪password}"
 | base64 --decode ; echo
```

7. Port Forward the Grafana dashboard to see whats happening

```
$ export POD_NAME=$(kubectl get pods --namespace prometheus -l "app=grafana,
↪release=grafana" -o
  jsonpath="{.items[0].metadata.name}")
$ kubectl --namespace prometheus port-forward $POD_NAME 3000
```

8. Go to http://localhost:3000 in your browser. You should see the Grafana login screen.If step 7 gives connectivity issue then we can change type as LoadBalancer in Grafana service file that will create an external endpoint and URL will be accessible.
9. Set the SMTP settings in Grafana configmap to send email alerts notification.

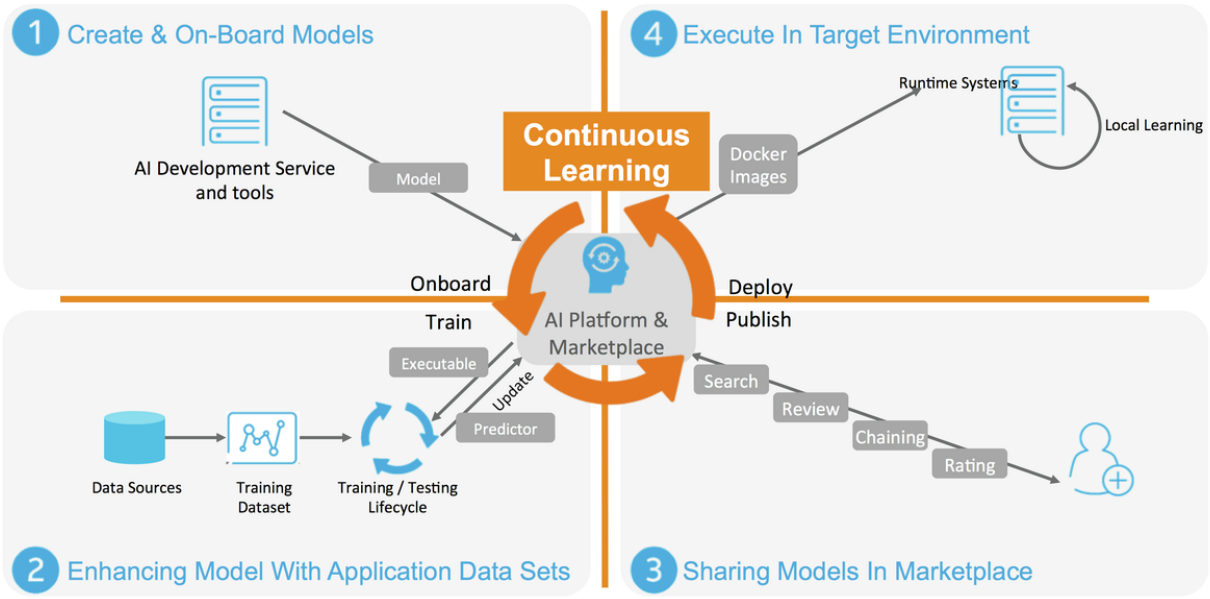
CHAPTER 5

Contributors to the Acumos Platform Code

5.1 Platform Architecture

5.1.1 Architecture Guide

5.1.1.1 Introduction



Acumos is a platform which enhances the development, training and deployment of AI models. Its purpose is to scale up the introduction of AI-based software across a wide range of industrial and commercial problems in order to reach a critical mass of applications. In this way, Acumos will drive toward a data-centric process for producing software based upon machine learning as the central paradigm. The platform seeks to empower data scientists to

publish more adaptive AI models and shield them from the task of custom development of fully integrated solutions. Ideally, software developers will use Acumos to change the process of software development from a code-writing and editing exercise into a classroom-like code training process in which models will be trained and graded on their ability to successfully analyze datasets that they are fed. Then, the best model can be selected for the job and integrated into a complete application.

Acumos is not tied to any specific modeling language or toolkit and not limited to any one run-time infrastructure or cloud service. Acumos creates an open source mechanism for packaging, sharing, licensing and deploying AI models in the form of portable, containerized microservices and publishes them in a shared, secure catalog. Using Acumos, data scientists can build abstract AI models, using their favorite or most appropriates tools, which can be adapted to a variety of data formats, using data adaptation libraries, and formed into applications using a simplified chaining process. These models are intended to be used by IT professionals, who can integrate the models into practical applications, without a data science background or training in the various AI toolkits employed by the data scientists.

Acumos is intended to enable the use of a wide range of tools and technologies in the development of machine learning models including support for both open sourced and proprietary toolkits. Models can be easily onboarded and wrapped as containerized microservices which are interoperable with many other components.

Acumos provides a toolkit-independent App Store called a Marketplace for data-powered decision making and artificial intelligence software models. It provides a means to securely share AI microservices along with information on how they perform, such as ratings, popularity statistics and user-provided reviews to apply crowd sourcing to software development. The platform provides integration between model developers and applications in order to automate the process of user feedback, exception handling and software updates.

Acumos Design Studio can be used to chain together multiple models, along with data translation tools, filters and output adapters into a full end-to-end solution which can then be deployed into any run-time environment. The Acumos catalog contains information on the licensing and execution requirements of both reusable AI models and fully integrated solutions and this can be easily searched to make model selection a simple process.

Acumos' Data Broker provides capabilities for acquiring data from external sources, then using the data to train or tune models and retaining the data in order to provide retraining of future models.

The source code of the Acumos platform, itself, is available under an OSI-approved open source license so that any aspect can be readily adapted to new development toolkits, private data source and datastreams and custom run-time environments.

5.1.1.2 Scope

This document provides an architectural overview of the Acumos platform, as of the Athena release. All aspects of the Acumos platform are represented in this overview, including:

- the Acumos portal, a web-based framework and content management system supporting Acumos platform operator and user interaction with the platform
- various core components of the Acumos platform that are deployed as integrated services with the Acumos portal, and provide specific functions in support of the user experience, such as
  - a model onboarding service
  - a model design studio service
  - various model deployment clients and supporting components, as of this release supporting deployment under Azure, OpenStack, and kubernetes
  - an inter-platform model federation service
  - various common services, such as a common data service and microservice generation service
- various model developer support clients, used in model onboarding
- various non-Acumos components that provide necessary dependencies as services to the platform, such as

  - runtime environment and control based upon Docker-CE and/or Kubernetes
  - a database backend service based upon MariaDB
  - a default artifact repository for Maven and docker artifacts, based upon Nexus
  - a default ingress controller / reverse proxy for the plaform, based upon Kong
  - various components that provde a platform logging and analytics service
    * a platform component log aggregation service based upon Filebeat
    * a platform host and container analytics service based upon Metricbeat
    * logging/analytics storage, search, and visualization based upon the ELK stack (ElasticSearch, Logstash, Kibana)
- deployment and operations support tools for the platform

5.1.1.3 Requirements

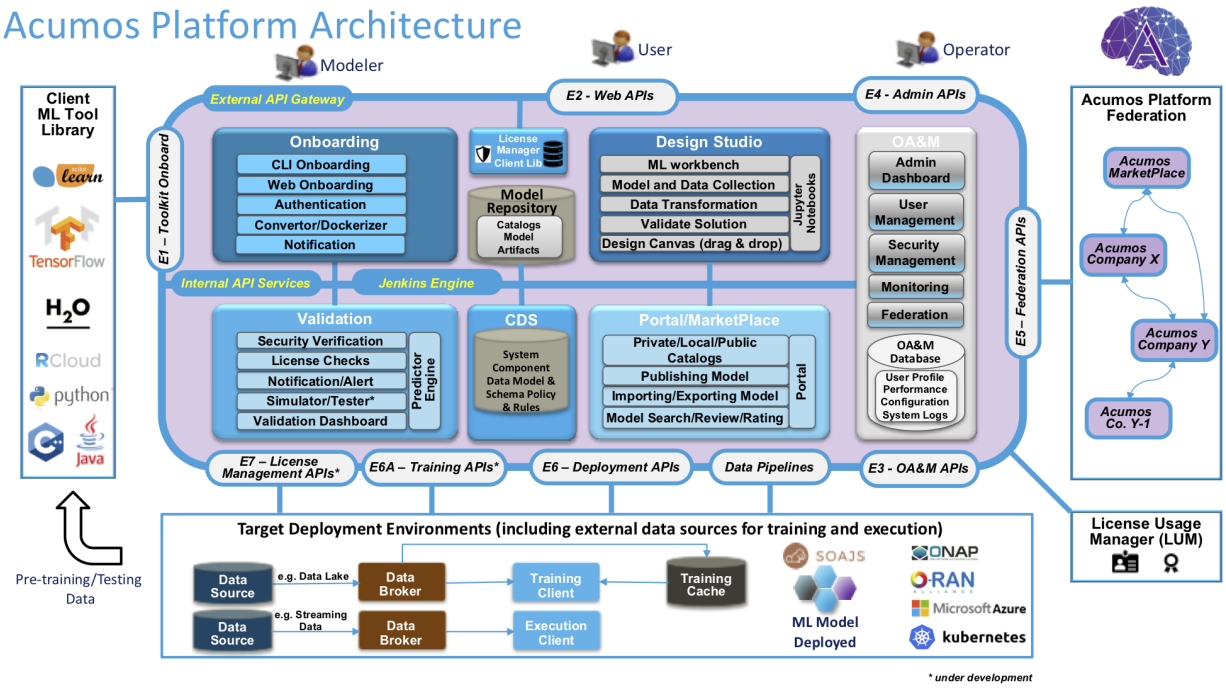
As described on the Acumos.org website, Acumos AI is a platform and open source framework that makes it easy to build, share, and deploy AI apps, and operate the Acumos portals that enable those capabilities. Acumos standardizes the infrastructure stack and components required to run an out-of-the-box general AI environment. This frees data scientists and model trainers to focus on their core competencies and accelerates innovation.

The Acumos platform enables the following high-level set of capabilities in support of the goals above, which are fulfilled through the various components and interfaces of the Acumos platform architecture:

- Build machine-learning models and solutions
  - Use client libraries to generate model package for onboarding by CLI or Web
  - Generate model microservice images with embedded model runners based upon an Ubuntu docker base image
  - Design and generate composite solutions as a directed graph of multiple model microservices, with additional supporting components
- Share models and solutions
  - Onboard models by CLI and Web
  - Share with your team, and publish to company and public marketplaces
  - Federate multiple Acumos portals for model/solution distribution
- Deploy models and solutions
  - Download for local deployment under docker and kubernetes
  - Deploy to public and private clouds (Azure, OpenStack)
  - Interact with models, and observe solution-internal dataflow
- Operate Acumos platforms
  - Deploy the platform under docker or kubernetes, as a single-node (all-in-one) or multi-node platform
  - Secure the platform
  - Administer the platform via the portal UI
  - logging and analytics collectiom, storage, and visualization

5.1.1.4 Architecture

Architecture Overview



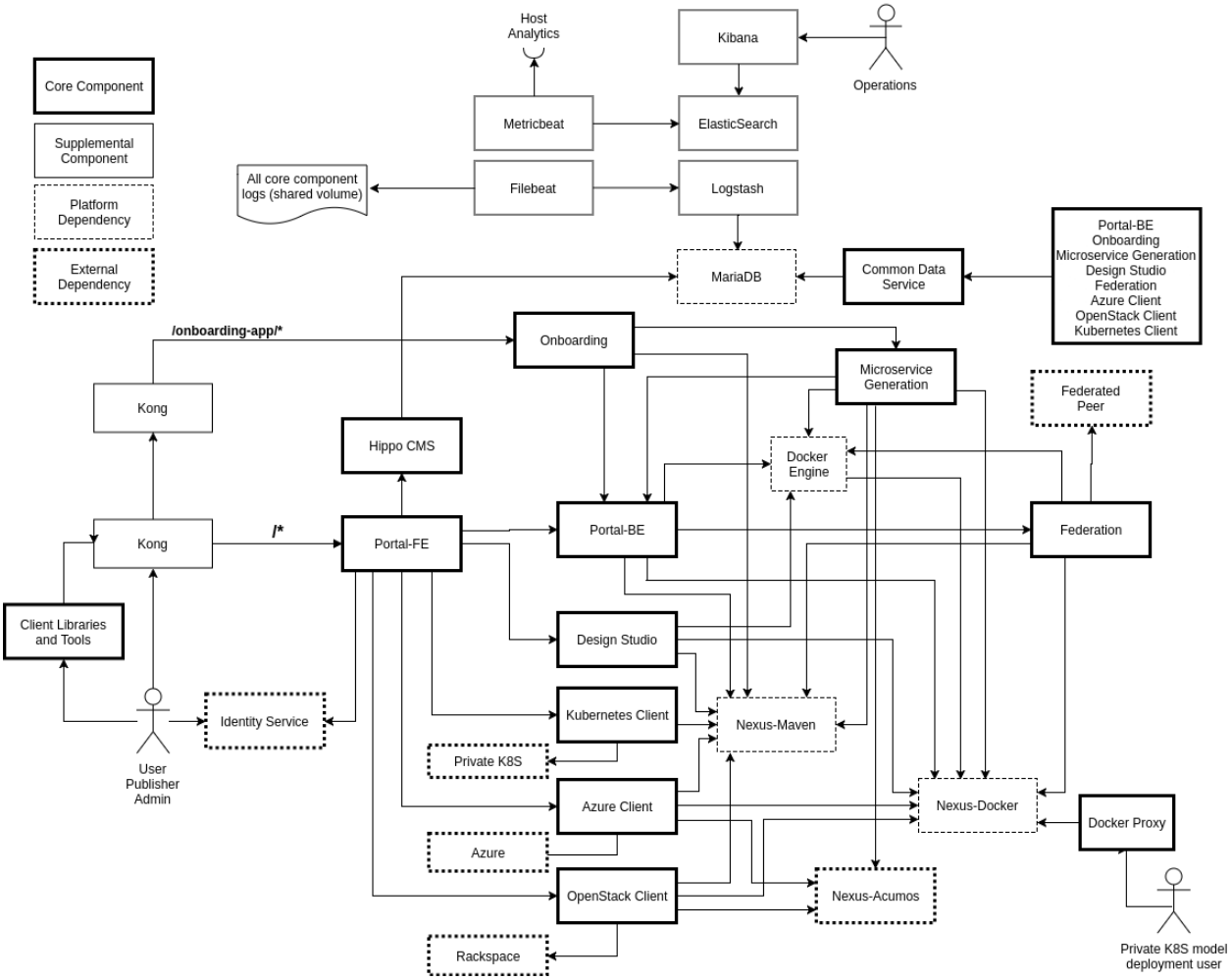
Component Interactions

The following diagram shows the major dependencies among components of the Acumos architecture, and with external actors. The arrow represent dependency, e.g. upon APIs, user interfaces, etc. The arrows are directed at the provider of the dependency. Some dependencies are so common that they aren't shown directly, for diagram clarity. These include:

- collection of logs from all components
- dependency upon the Common Data Service (shown as a single block of components)

The types of components/actors in the diagram are categorized as:

- Core Component: components that are developed/packaged by the Acumos project, and provide/enable core functions of the Acumos platform as a service
- Supplemental Component: components that are integrated from upstream projects, in some cases packaged as Acumos images, and provide supplemental/optional support functions for the platform. These functions may be provided by other components, or omitted from the platform deployment.
- Platform Dependency: upstream components that are required, to support key platform functions such as relational database and docker image creation. The examples shown (Nexus and Docker) may be replaced with other components that are API-compatible, and may be pre-existing, or shared with other applications.
- External Dependency: external systems/services that are required for the related Acumos function to be fully usable



Interfaces and APIs

External Interfaces and APIs

E1 - Toolkit Onboarding

The various clients used to onboard models call the APIs in the Onboarding service. See the Onboading App documentation for details.

E2 - Web APIs

The portal Web API (E2) are the interface for the users to upload their models to the platform. It provides means to share AI microservices along with information on how they perform. See the following for more information:

- Portal Web API

E3 - OA&M APIs

The OA&M subsystem defines data formats supported by the various logging and analytics support components described under *Operations, Admin, and Maintenance (OAM)*. These are primarily focused on log formats that Acumos components will follow when saving log files that are collected by the logging subsystem.

E4 - Admin APIs

The Admin API (E4) provides the interfaces to configure the site global parameters. See the following for more information:

- Portal Marketplace

E5 - Federation APIs

The federation (public) E5 interface is a REST-based API specification. Any system that decides to federate needs to implement this interface, which assumes a pull-based mechanism. As such, only the server side is defined by E5. The server allows clients to poll to discover solutions, and to retrieve solution metadata, solution artifacts and user-provided documents. See the following for more information:

- Federation Gateway

E6 - Deployment APIs

The Deployment subsystem primarily consumes APIs of external systems such as cloud service environments, including Azure, OpenStack, and private kubernetes clouds. The developer guides for the "Deployers" that coordinate model deployment in those specific environments address the specific APIs consumed by those Deployers. See the following for more information:

- Acumos Azure Client
- Openstack Client
- Kubernetes Client

Microservice Generation

The DCAE model API is intended to be used with models dedicated for ONAP. It builds a DCAE/ONAP microservice and required artifacts. See the Microservice Generation documentation for details.

Internal Interfaces and APIs

Common Data Service

The Common Data Service provides a storage and query micro service for use by system components, backed by a relational database. The API provides Create, Retrive, Update and Delete (CRUD) operations for system data including users, solutions, revisions, artifacts and more. The microservice endpoints and objects are documented extensively using code annotations that are published via Swagger in a running server, ensuring that the documentation is exactly synchronized with the implementation. View this API documentation in a running CDS instance at a URL like the following, but consult the server's configuration for the exact port number (e.g., "8000") and context path (e.g., "ccds") to use:

```
http://localhost:8000/ccds/swagger-ui.html
```

See the following for more information:

- Common Data Service

Hippo CMS

Portal Backend

Federation Gateway

The federation (local) E5 interface is a REST-based API specification, just like the public interface. This interface provides secure communication services to other components of the same Acumos instance, primarily used by the Portal. The services include querying remote peers for their content and fetching that content as needed. See the following for more information:

- Federation Gateway

Microservice Generation

Azure Client

The Azure Client exposes two APIs that are used by the Portal-Markeplace to initiate model deployment in the Azure cloud service environment:

- POST /azure/compositeSolutionAzureDeployment
- POST /azure/singleImageAzureDeployment

The Azure Client API URL is configured for the Portal-Markeplace in the Portal-FE component template (docker or kubernetes).

See Azure Client API for details.

OpenStack Client

The OpenStack Client exposes two APIs that are used by the Portal-Markeplace to initiate model deployment in an OpenStack service environment hosted by Rackspace:

- POST /openstack/compositeSolutionOpenstackDeployment
- POST /openstack/singleImageOpenstackDeployment

The OpenStack Client API URL is configured for the Portal-Markeplace in the Portal-FE component template (docker or kubernetes).

See OpenStack Client API for details.

Kubernetes Client

The Kubernetes Client expose one API that is used by the Portal-Markeplace to provide the user with a downloadable deployment package for a model to be deployed in a private kubernetes service environment:

- GET /getSolutionZip/{solutionId}/{revisionId}

The Kubernetes Client API URL is configured for the Portal-Markeplace in the Portal-FE component template (docker or kubernetes).

See Kubernetes Client API for details.

ELK Stack

The ELK Stack is used to provide the *E3 - OA&M APIs* via which components publish standard-format log files for aggregation and presentation at operations dashboards.

Nexus

The Nexus component exposes two APIs enabling Acumos platform components to store and access artifacts in various repository types, including:

- Maven (for generic artifacts)
- docker (as a docker registry), using the Docker Registry HTTP API V2

The Maven repository service is accessed via an API exposed thru the *Nexus Client* Java library. The docker repository service is accessed via the Docker Registry HTTP API V2. Both services are configured for clients through URLs and credentials defined in the component template (docker or kubernetes).

Docker

The docker-engine is the primary service provided by *Docker-CE*, as used in Acumos. The docker-engine is accessed by the Docker Engine API.

The docker-engine API URL is configured for Acumos components in the template (docker or kubernetes) for the referencing component.

Kong

Kong provides a reverse proxy service for Acumos platform functions exposed to users, such as the Portal-Marketplace UI and APIs, and the Onboarding service APIs. The kong proxy service is configured via the Kong Admin API.

Core Components

The following sections describe the scope, role, and interaction of the core Acumos platform components and component libraries. The sections are organized per the Acumos project teams that lead development on the components.

Portal and User Experience

Portal Frontend

The Portal Frontend is designed to make it easy to discover, explore, and use AI models. It is completely built on angularJs and HTML. It uses portal backend APIs to fetch the data and display.

Portal Backend

Provides REST endpoints and Swagger documentation. Portal backend is built on Spring Boot which exposes the endpoints to manage the models.

For more information: Portal Backend Documentation

Acumos Hippo CMS

Acumos Hippo CMS is a content management system which is used to store the images of the text descriptions for the Acumos instance.

For more information: Acumos Hippo CMS Documentation

Model Onboarding

Onboarding App

The Onboarding app provides an ingestion interface for different types of models to enter the Acumos platform. The solution for accommodating a myriad of different model types is to provide a custom wrapping library for each runtime. The client libraries encapsulate the complexity surrounding the serialization and deserialization of models.

The Onboarding App interacts with the following Acumos platform components and supporting services:

- the Portal, which calls the Onboarding app during web-based model onboarding
- the Nexus Client, which stores and retrieves model artifacts from the Nexus maven repo
- the Common Data Service Client, which stores model attributes
- the Microservice Generation, which creates the dockerized microservice

For more information: Onboading Documentation.

Java Client

The Acumos Java Client is a Java client library used to on-board H2o.ai and Generic Java models. This library creates artifacts required by Acumos, packages them with the model in a bundle, and pushes the model bundle to the onboarding server.

The Java Client interacts with the Onboading app.

For more information: Java Client Documentation.

Python Client

The Acumos Java Client is a Python client library used to on-board Python models and more specifically Scikit learn, TensorFlow and TensorFlow/Keras models. It creates artifacts required by Acumos, packages them with the model in a bundle, and pushes the model bundle to the onboarding app.

The Python Client interacts with the Onboading app.

For more information: Python Client Documentation.

R Client

The R client is a R package that contains all the necessary functions to create a R model for Acumos. It creates artifacts required by Acumos, packages them with the model in a bundle, and pushes the model bundle to the onboarding app.

The R Client interacts with the Onboading app.

For more information: R Client Documentation.

Design Studio

The Design Studio component repository includes following components:

- Composition Engine
- TOSCA Model Generator Client
- Generic Data Mapper Service
- Data Broker (CSV and SQL)

For more information: Design Studio Documentation

Additional components are in separate repositories.

Design Studio Composition Engine

The Acumos Portal UI has a Design Studio that invokes the Composition Engine API to:

1. Create machine learning applications (composite solutions) out of the basic building blocks – the individual Machine Learning (ML) models contributed by the user community
2. Validate the composite solutions
3. Generate the blueprint of the composite solution for deployment on the target cloud

The Design Studio Composition Engine is Spring Boot backend component which exposes REST APIs required to carry out CRUD operations on composite solutions.

TOSCA Model Generator Client

The TOSCA Model Generator Client is a library used by the Onboarding component to generate artifacts (TGIF.json, Protobuf.json) that are required by the Design Studio UI to perform operations on ML models, such as drag-drop, display input output ports, display meta data, etc.

Generic Data Mapper Service

The Generic Data Mapper Service enables users to connect two ML models 'A' and 'B' where the number of output fields of model 'A' and input fields of model 'B' are the same. The user is able to connect the field of model 'A' to required field of model 'B'. The Data Mapper performs data type transformations between Protobuf data types.

Data Broker

At a high level, a Data Broker retrieves and converts the data into protobuf format. The Data Brokers retrieve data from the different types of sources like database, file systems (UNIX, HDFS Data Brokers, etc.), Router Data Broker, and zip archives.

The Design Studio provides the following Databrokers:

1. CSV DataBroker: used if source data resides in text file as a comma (,) separated fields.
2. SQL DataBroker: used if source data is SQL Data base. Currently MYSQL database is supported.

Runtime Orchestrator

The Runtime Orchestrator (also called Blueprint Orchestrator or Model Connector) is used to orchestrate communication between the different models in a Composite AI/ML solution.

For more information: Runtime Orchestrator Documentation.

Proto Viewer

This component allows visualization of messages transferred in protobuf format. This is a passive component that shows the messages explicitly delivered to it; it does not listen ("sniff") all network traffic searching for protobuf data. Displaying the contents of a protobuf message requires the corresponding protocol buffer definition (.proto) file, which are fetched from a network server, usually a Nexus registry.

For more information: Proto Viewer Documentation.

Deployment

The deployment components enable users to launch models and solutions (composite models with additional supporting components) in various runtime environments, which are generally specific to the deployment component "client". These clients are invoked by user actions in the Portal, e.g. selecting a deployment target for a model in the various UI views where deployment is an option.

Azure Client

The Azure Client assists the user in deploying models into the Azure cloud service, as described in the Deploy Acumos Model to Azure User Guide. The Azure Client uses Azure APIs to perform actions such as creating a VM where the model will be deployed. The process depends upon a variety of prerequisite configuration steps by the user, as described in the user guide linked above.

Once a VM has been created, the Azure Client executes commands on the VM to download and deploy the various model components. See the Acumos Azure Client Developers Guide for more info.

The Azure Client interacts with the following Acumos platform components and supporting services:

- the Portal, for which the Azure Client coordinates model deployment upon request by the user
- the Nexus Client, which retrieves model artifacts from the Nexus maven repo
- the Common Data Service Client, which retrieves model attributes stored in the CDS
- the Runtime Orchestrator, which the Azure Client configures with the information needed to route protobuf messages through a set of composite model microservices

- the Data Broker, which the Azure Client configures with the information needed to ingest model data into the model
- the Proto Viewer, which the Azure Client configures with the information needed to display model messages on the Proto Viewer web interface
- the Filebeat service, which collects the log files created by the Azure Client, using a shared volume
- supporting services
  - the docker-engine, which retrieves docker images from the Acumos platform Nexus docker repo
  - the Acumos project Nexus docker repo, for access to deployment components such as the Runtime Orchestrator, Data Broker, and Proto Viewer

Openstack Client

The Openstack Client assists the user in deploying models into an Openstack based public cloud hosted by Rackspace, as described in the Openstack Client Users Guide. The Openstack Client uses OpenStack APIs to perform actions such as creating a VM where the model will be deployed. The process depends upon a variety of prerequisite configuration steps by the user, as described in the user guide linked above.

Once a VM has been created, the Openstack Client executes commands on the VM to download and deploy the various model components. See the Openstack Client Developers Guide for more info.

The Openstack Client interacts with the following Acumos platform components and supporting services:

- the Portal, for which the OpenStack Client coordinates model deployment upon request by the user
- the Nexus Client, which retrieves model artifacts from the Nexus maven repo
- the Common Data Service Client, which retrieves model attributes stored in the CDS
- the Runtime Orchestrator, which the Openstack Client configures with the information needed to route protobuf messages through a set of composite model microservices
- the Data Broker, which the Openstack Client configures with the information needed to ingest model data into the model
- the Proto Viewer, which the Openstack Client configures with the information needed to display model messages on the Proto Viewer web interface
- the Filebeat service, which collects the log files created by the Openstack Client, using a shared volume
- supporting services
  - the docker-engine, which retrieves docker images from the Acumos platform Nexus docker repo
  - the Acumos project Nexus docker repo, for access to deployment components such as the Runtime Orchestrator, Data Broker, and Proto Viewer

Kubernetes Client

The Kubernetes Client and associated tools assists the user in deploying models into a private kubernetes cloud, as described in Acumos Solution Deployment in Private Kubernetes Cluster.

For a model that the user wants to deploy (via the "deploy to local" option), the Kubernetes Client generates a deployable solution package, which as described in the guide above, is downloaded by the user. After unpacking the solution package (zip file), the user then takes further actions on the host where the models will be deployed, using a set of support tools included in the downloaded solution package:

- optionally installing a private kubernetes cluster (if not already existing)
- deploying the model using an automated script, and the set of model artifacts included in the solution package

The Kubernetes Client interacts with the following Acumos platform components:

- the Portal, for which the Kubernetes Client coordinates model deployment upon request by the user
- the Nexus Client, which retrieves model artifacts from the Nexus maven repo
- the Common Data Service Client, which retrieves model attributes stored in the CDS
- the Filebeat service, which collects the log files created by the Kubernetes Client, using a shared volume

The Kubernetes Client deployment support tool "deploy.sh" interacts with the following Acumos platform components and supporting services:

- the Runtime Orchestrator, which deploy.sh configures with the information needed to route protobuf messages through a set of composite model microservices
- the Data Broker, which deploy.sh configures with the information needed to ingest model data into the model
- the Proto Viewer, which deploy.sh configures with the information needed to display model messages on the Proto Viewer web interface
- supporting services
  - the docker-engine, which retrieves docker images from the Acumos platform Nexus docker repo
  - the kubernetes master (via the kubectl client), to configure, manage, and monitor the model components
  - the Acumos project Nexus docker repo, for access to deployment components such as the Runtime Orchestrator, Data Broker, and Proto Viewer

Docker Proxy

As described in Acumos Solution Deployment in Private Kubernetes Cluster, the Docker Proxy provides an authentication proxy for the Acumos platform docker repo. Apart from the use for model deployment into kubernetes, the Docker Proxy addresses a key need of Acumos platform users, and opportunities to enhance the other deployment clients related to:

- the ability to retrieve model microservice docker images from the Acumos platform using the normal process of "docker login" followed by "docker pull"

Using the normal docker protocol for image download will enhance the simplicity, speed, efficiency, and reliability of:

- user download of a model for local deployment, e.g. for local testing
- deployment processes using the Azure and OpenStack clients, to be considered as a feature enhancement in the Boreas release

The Docker Proxy interacts with the following Acumos platform components and supporting services:

- the Kubernetes Client deployment support tool "deploy.sh", for which the Docker Proxy provides docker login and image pull services
- supporting services
  - The Nexus docker repo, from which the Docker Proxy pulls model microservice images

Catalog, Data Model and Data Management

This project includes the Common Data Service, the Federation Gateway, and the Model Schema subprojects.

Common Data Service

The Acumos Common Data Service provides a storage and query layer between Acumos system components and a relational database. The server component is a Java Spring-Boot application that provides REST service to callers and uses Hibernate to manage the persistent store. The client component is a Java library that provides business objects (models) and methods to simplify the use of the REST service.

For more info: ../../submodules/common-dataservice/docs/index

Federation Gateway

The Federation Gateway component provides a mechanism to exchange models between two Acumos instances via a secure network channel. The Gateway is implemented as a server that listens for requests on a REST API. It also has a client feature that communicates with remote instances.

For more info: ../../submodules/federation/docs/index

Model Schema

The Model Schema is the JSON schema used to define and validate the Acumos model metadata generated by client libraries such as the Acumos python client library.

For more info: ../../submodules/model-schema/docs/index

Common Services

Microservice Generation

The Microservice Generation component is in charge of dockerize the model, create the microservice and store artifacts in Nexus.

For more information Microservice Generation.

Nexus Client

Generic Model Runner

Python DCAE Model Runner

Supplemental Components

The following sections describe the scope, role, and interaction of components that supplement the Acumos platform as deployed components and tools. These components and tools are developed and/or packaged by the Acumos project to provide supplemental support for the platform.

Operations, Admin, and Maintenance (OAM)

The Platform-OAM project maintains the repos providing:

- Acumos platform deployment support tools

- Logging and Analytics components based upon the ELK Stack, of which Acumos uses the open source versions

System Integration

The System Integration repo contains Acumos platform deployment support tools e.g.

- Docker-compose templates for manual platform installation under docker-ce
- Kubernetes templates for platform deployment in Azure-kubernetes
- Oneclick / All-In-One (AIO) platform deployment under docker-ce or kubernetes
  - See One Click Deploy User Guide

Filebeat

Filebeat is a support component for the ELK stack. Filebeat monitors persistent volumes in which Acumos components save various log files, and aggregates those files for delivery to the Logstash service.

Metricbeat

Metricbeat is a support component for the ELK stack. Metricbeat monitors host and process resources and delivers the to the Logstash service.

ELK Stack

The ELK Stack provides the core services that archive, access, and present analytics and logs for operations support dashboards. It includes:

- Logstash: a server-side data processing pipeline that ingests data from multiple sources, transforms it, and then sends it to ElasticSearch for storage
- ElasticSearch: a data storage, search, and analytics engine
- Kibana: a visualization frontend for ElasticSearch based data

See *Platform Operations, Administration, and Management (OA&M) User Guide* for more info.

External Components

The following sections describe the scope, role, and interaction of externally-developed components that are deployed (some, optionally) as part of the Acumos platform or as container runtime environmments in which the Acumos platform is deployed.

MariaDB

MariaDB is a relational database system. Acumos platform components that directly use MariaDB for database services include:

- Common Data Service, for storage of platform data in the CDS database
- Portal-Marketplace, for storage of Hippos CMS data

- ELK stack, for access to platform user analytics

Acumos platform components access the MariaDB service via a URL and credentials defined in the component template (docker or kubernetes).

Nexus

Nexus (Nexus 3) is used as an artifact repository, for

- artifacts related to simple and composite models
- model microservice docker images

Acumos platform components that directly use Nexus for repository services include:

- Design Studio
- Onboarding
- Azure Client
- Microservice Generation
- Portal-Marketplace
- Federation

Kong

The Kong Community Edition is an optional component used as needed as a reverse proxy for web and API requests to the platform. The primary web and API services exposed through the kong proxy are

- the Onboarding service APIs (URL paths based upon /onboarding-app)
- the Portal-Marketplace web frontend and APIs (all other URL paths)

Docker-CE

Docker Community Edition is used as a key component in the platform for the purposes:

- accessing docker repositories, including the Acumos platform docker repository
- building docker images
- launching containers on request of the kubernetes master node

The docker-engine is the main feature of Docker-CE used in Acumos, and is deployed:

- for Docker-CE based platform deployments, on one of the platform hosts (e.g. VMs or other machines)
- for kubernetes based platform deployments, as a containerized service using the Docker-in-Docker (docker-dind) variant of the official docker images

Kubernetes

Kubernetes provides a container management environment in which the Acumos platform (as a collection of docker image components) and models can be deployed. Kubernetes cluster installation tools are provided by the kubernetes-client repo, and can be used for establishing a private kubernetres cluster where the Acumos platform and models can be deployed. The Acumos AIO toolkit can deploy the Acumos platform in a private kubernetes cluster. For kubernetes

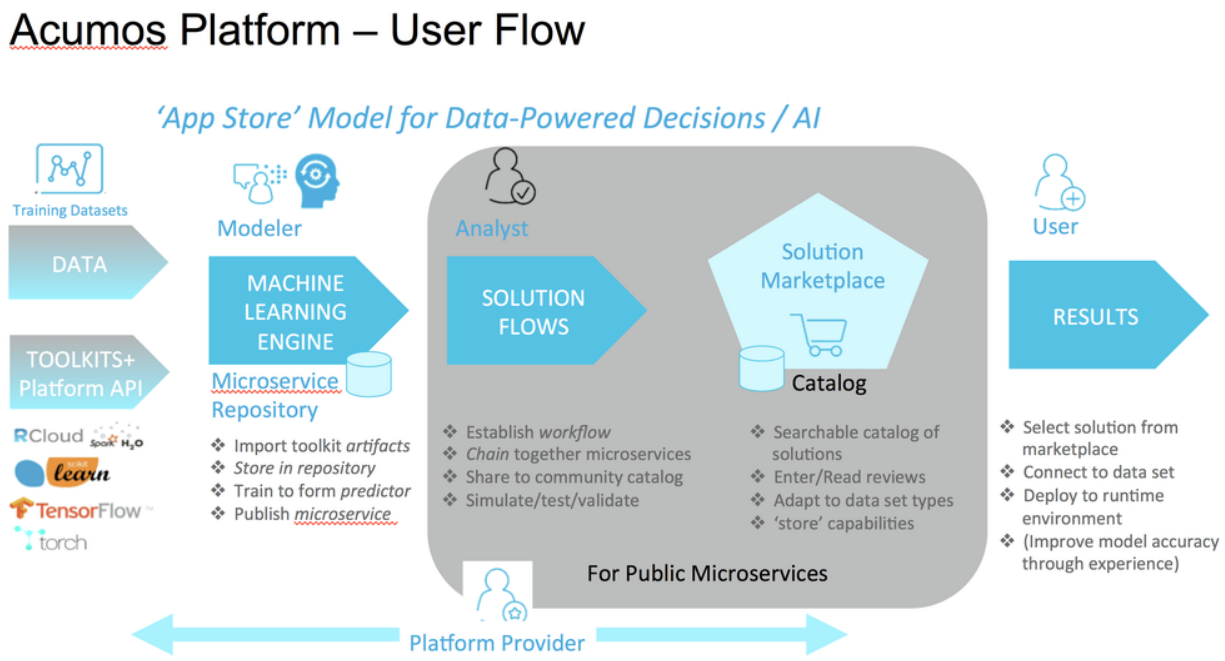
clusters hosted by public cloud providers e.g. Azure, Acumos provides kubernetes templates for the Acumos platform components in the system-integration repo.

5.1.1.5 Platform Flow

User Journeys

Following are some illustrative "user journey" diagrams for common Acumos workflows.

Acumos Platform User Flow



Acumos User Signup Flow

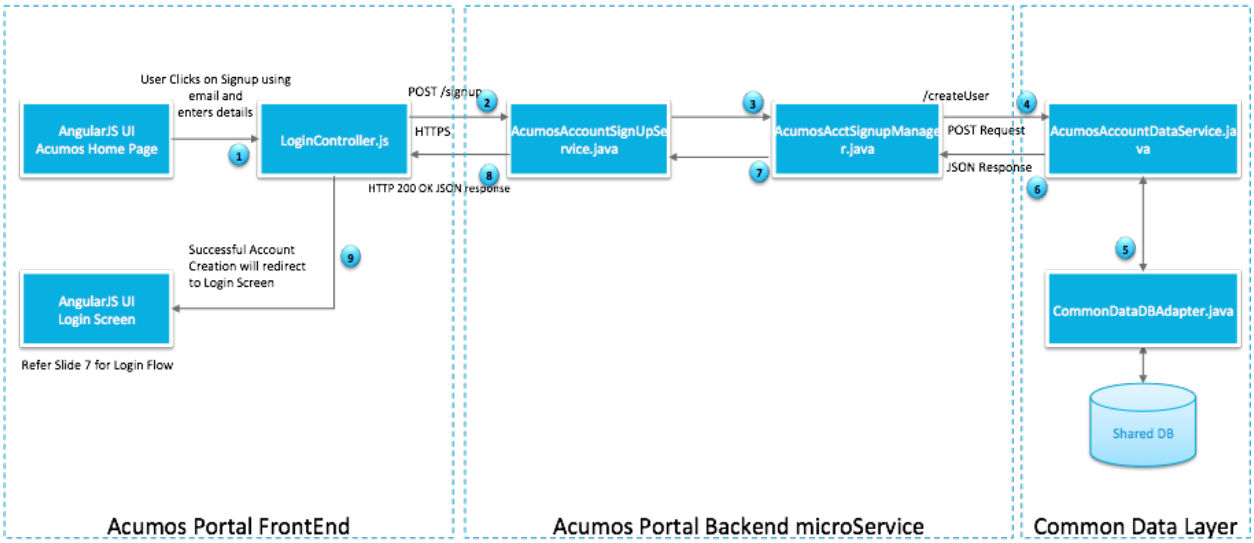
Acumos User Login Flow

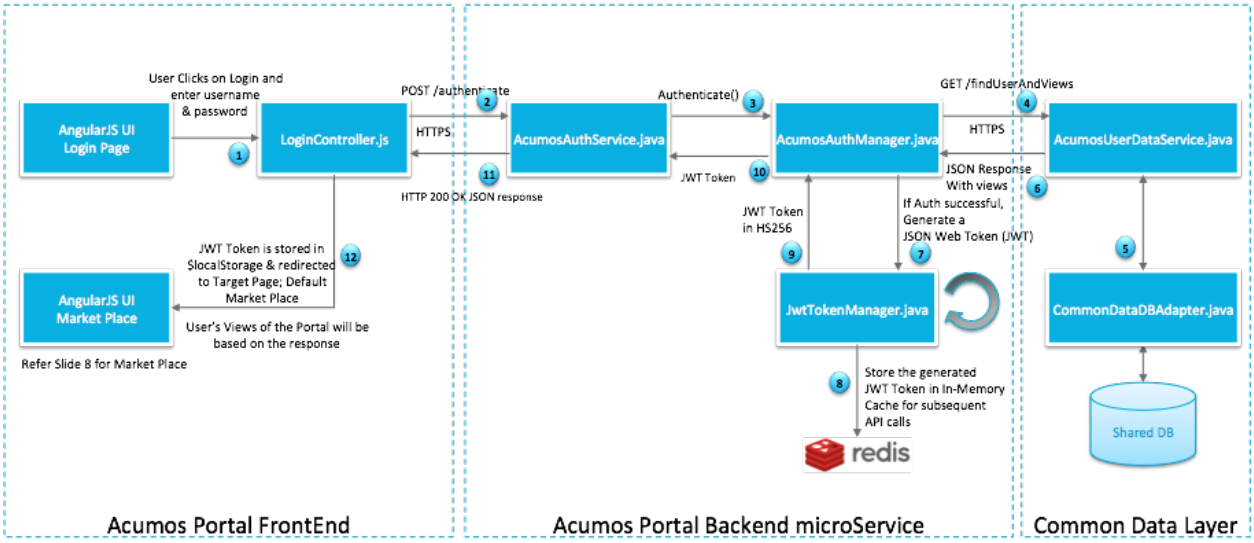
Component Interaction

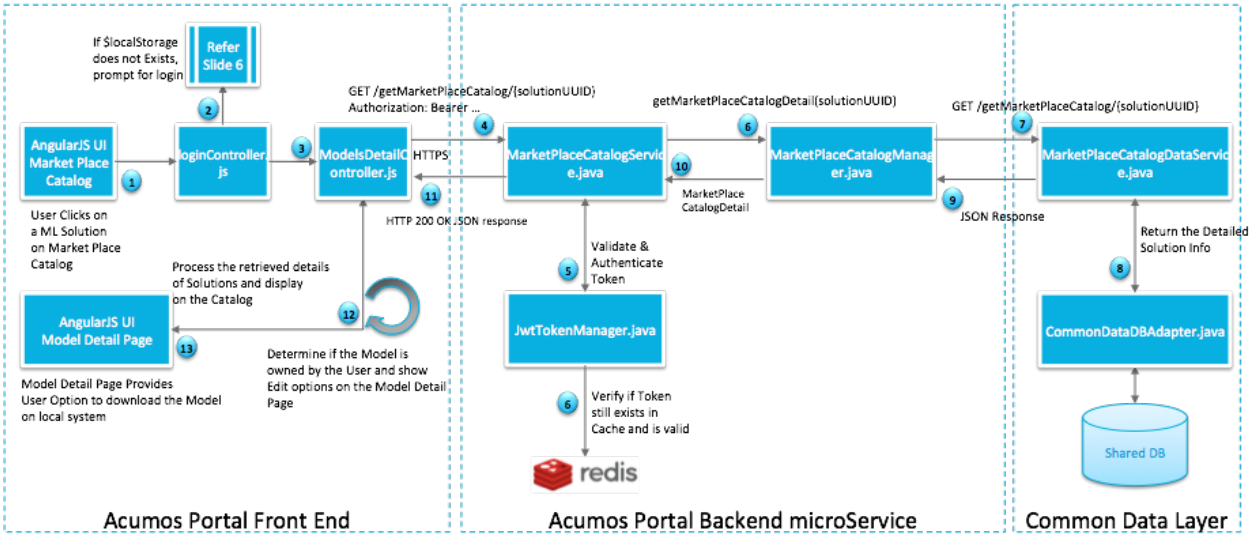
Following are some illustrative diagrams for common Acumos component interactions.

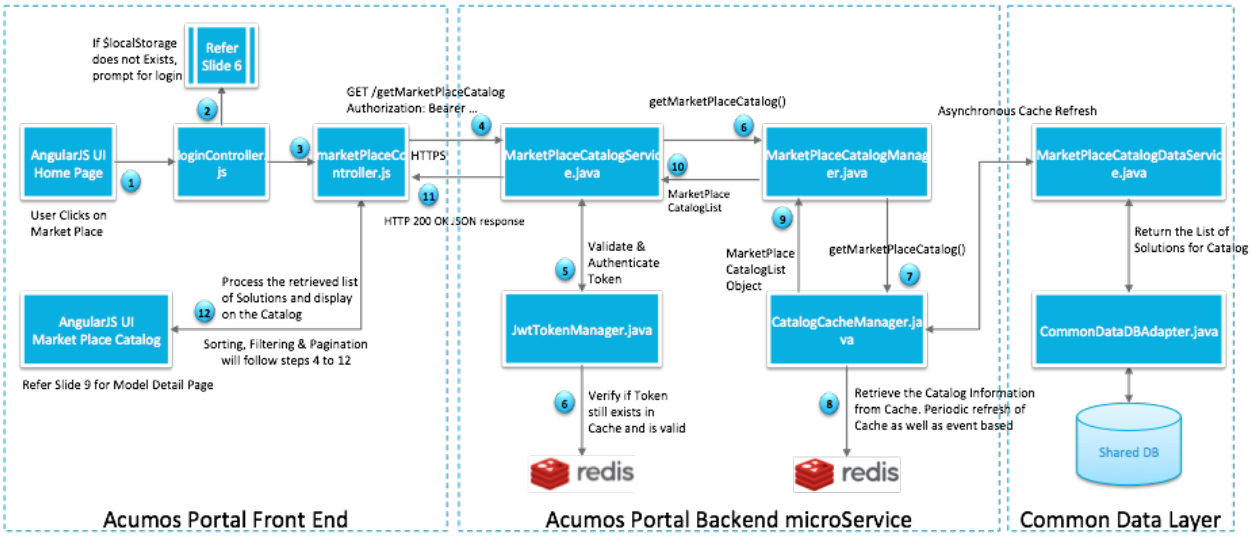
Acumos Model Detail Flow

Acumos Catalog Flow


User Clicks on Signup using email and enters details
AngularJS UI Acumos Home Page
LoginController.js
POST /signup
HTTPS
AcumosAccountSignUpService.java
AcumosAcctSignupManager.java
/createUser
POST Request
JSON Response
AcumosAccountDataService.java
HTTP 200 OK JSON response
Successful Account Creation will redirect to Login Screen
AngularJS UI Login Screen
Refer Slide 7 for Login Flow
CommonDataDBAdapter.java
Shared DB
Acumos Portal FrontEnd
Acumos Portal Backend microService
Common Data Layer


User Clicks on Login and enter username & password
AngularJS UI Login Page
LoginController.js
POST /authenticate
HTTPS
AcumosAuthService.java
Authenticate()
AcumosAuthManager.java
GET /findUserAndViews
HTTPS
AcumosUserDataService.java
JSON Response With views
JWT Token
HTTP 200 OK JSON response
JWT Token in HS256
If Auth successful, Generate a JSON Web Token (JWT)
JWT Token is stored in $localStorage & redirected to Target Page; Default Market Place
AngularJS UI Market Place
User's Views of the Portal will be based on the response
Refer Slide 8 for Market Place
JwtTokenManager.java
CommonDataDBAdapter.java
Store the generated JWT Token in In-Memory Cache for subsequent API calls
redis
Shared DB
Acumos Portal FrontEnd
Acumos Portal Backend microService
Common Data Layer


If $localStorage does not Exists, prompt for login
Refer Slide 6
AngularJS UI Market Place Catalog
LoginController.js
ModelsDetailController.js
GET /getMarketPlaceCatalog/{solutionUUID}
Authorization: Bearer ...
HTTPS
MarketPlaceCatalogService.java
getMarketPlaceCatalogDetail(solutionUUID)
MarketPlaceCatalogManager.java
GET /getMarketPlaceCatalog/{solutionUUID}
MarketPlaceCatalogDataService.java
MarketPlace CatalogDetail
JSON Response
HTTP 200 OK JSON response
User Clicks on a ML Solution on Market Place Catalog
Validate & Authenticate Token
Return the Detailed Solution Info
Process the retrieved details of Solutions and display on the Catalog
AngularJS UI Model Detail Page
Determine if the Model is owned by the User and show Edit options on the Model Detail Page
JwtTokenManager.java
CommonDataDBAdapter.java
Model Detail Page Provides User Option to download the Model on local system
Verify if Token still exists in Cache and is valid
redis
Shared DB
Acumos Portal Front End
Acumos Portal Backend microService
Common Data Layer



Inter-Component Message Flows

Following are some actual message flows between Acumos components. Some URI parameters have been abstracted to reduce the complexity of the flows. You can click on the flows to view them in native SVG form, which makes it easier to resize, scroll around, etc.

Web Onboarding

This flow shows a typical web onboarding sequence.

CLI Onboarding

This flow shows a typical web onboarding sequence.

Model Publishing

This flow shows a typical model publishing sequence.

Request for Published Solution Subscription, at Subscribing Platform

This flow shows the processing of a request for subcription to a solution published by a peer platform, at the subscribing platform. Note that some subsquent actions to these steps are not shown in this flow version, e.g. retrieval of the artifacts for the subscribed solution.

Request for Published Solution Subscription, at Publishing Platform

This flow shows the processing of a request for subcription to a solution published by a platform, when received at the publishing platform. Note that some subsquent actions to these steps are not shown in this flow version, e.g. retrieval

of the artifacts for the subscribed solution.

5.2 Component Guides

The *Component Guides* section contains a variety of information that is useful to developers who are working on the platform code. Most projects are written in Java, with the Javadoc available here .

5.2.1 Component Guides

Component guides contain a variety of information that is useful to developers who would like to work on the code. Most projects are written in Java, and the Javadoc is available here .

Note: Data Scientists who are contributing models should reference the Portal - For Modelers pages of the Portal and Marketplace User Guide.

5.2.1.1 Catalog, Data Model, and Data Management

- Common Data Service
- Federation Gateway
- Model Schema

5.2.1.2 Common Services

- H2O Model Builder
- H2O Model Runner
- H2O Java Model Runner
- Microservice Generation
- Nexus Client
- Python DCAE Model Runner
- Python Model Runner
- RDS Model Runner
- Security Verification of Models
- License Manager Client Library
- License Usage Manager

5.2.1.3 Design Studio

The Design Studio component repository includes the Composition Engine, TOSCA Model Generator Client, Generic Data Mapper Service, CSV Data Broker, and SQL Data Broker. Additional components are in separate repositories.

- Design Studio
- ML Workbench
- Proto Viewer ("Probe")
- Runtime Orchestrator ("Model Connector")

5.2.1.4 Deployment

This project maintains clients for deploying models to different environments.

- Deployment Client
- Kubernetes Client
- Azure Client
- OpenStack Client
- Predictor Management

5.2.1.5 Model On-Boarding

- On-boarding
- Java Client (Generic, H20, spark)
- Python Client, recommended version for CLIO release is 0.8.0
- R Client
- C++ Client
- Onnx Client,

5.2.1.6 Portal and Marketplace

- Acumos Hippo CMS
- Portal

5.2.1.7 Operations, Administration, and Management (OA&M)

- Platform OA&M

5.2.1.8 System Integration

- System Integration

Example Models

- Face Privacy Filter
- Image Classification
- Image Mood Classifier
- VM Predictor

5.3 Documentation Guide

docs-contributor-guide/index

Please also visit the Developer wiki, which includes sections on how to contribute to Acumos.

CHAPTER 6

Indices and Tables

- search