Acumos Documentation

Release 1.0

Mar 13, 2020

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 Clio Release, 13 November 2019

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.

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
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

1.1.2 Boreas Release, 5 Jun 2019

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.

[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.

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.

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

1.1.3 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.

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.

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

1.1.4 Athena Release, 7 Nov 2018

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.

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

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

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
- 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 and PFA: ONNX and PFA 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 One Click Deploy User Guide

4.1.1 Acumos OneClick / All-in-One (AIO) User Guide

This user guide describes how to deploy Acumos platforms using the "One Click deploy" tools designed for those who want a simple and automated way to deploy an Acumos platform.

What is an AIO deploy?

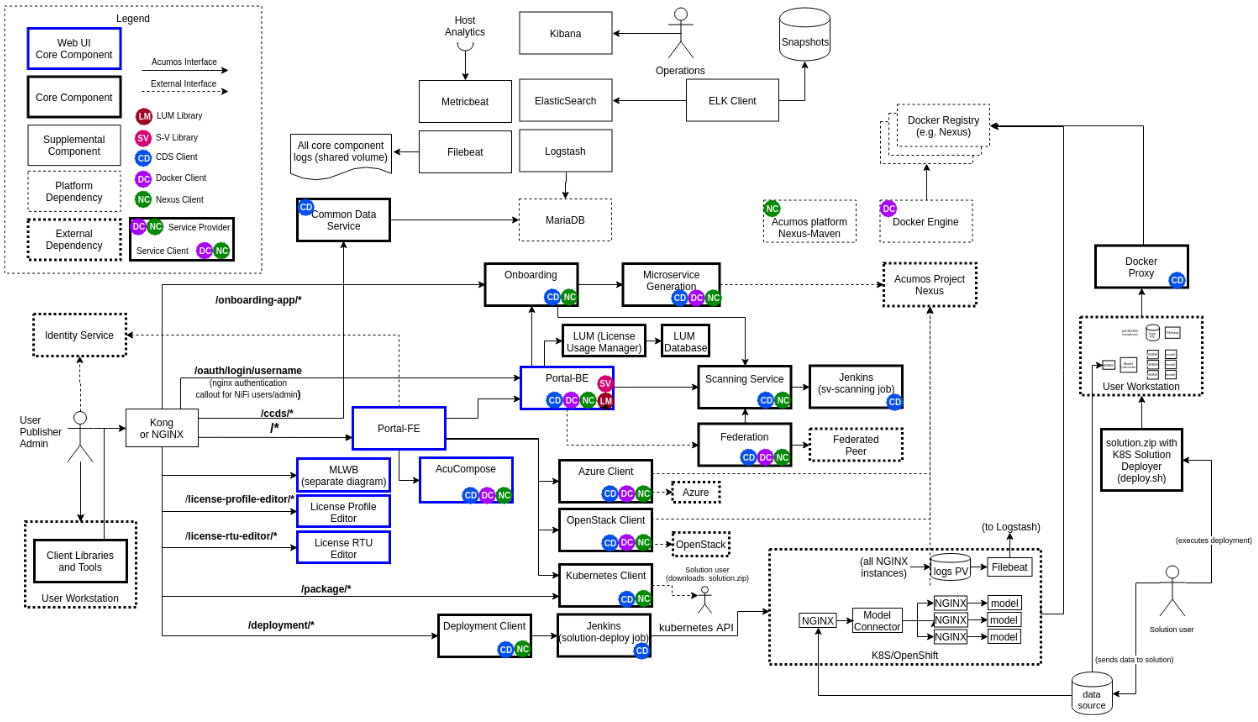
By default, the AIO deploy tools build an all-in-one instance of Acumos, with all Acumos data and components running under docker or kubernetes (k8s) on a single virtual machine or physical host machine.

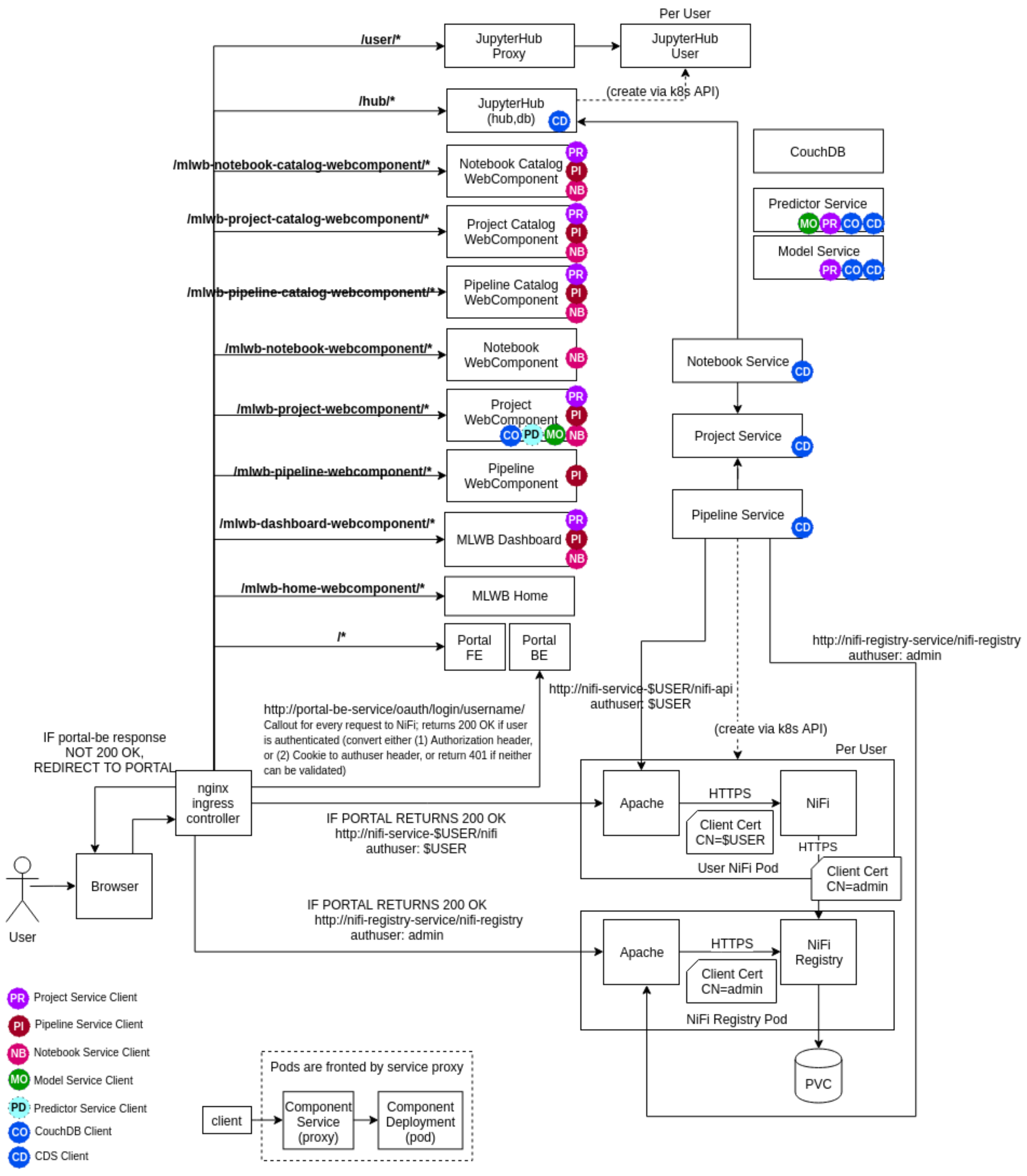
For k8s based deployments, both generic (standard k8s project tools) and OpenShift (RedHat's k8s distribution) are supported.

Options allow the user to deploy the platform:

- on a cluster of k8s nodes (note: distributing specific components across nodes based upon node labels is planned for future releases)
- with a subset of the components
- to use some components that have previously deployed somewhere, e.g. as a shared service

The resulting Acumos platform is illustrated in the following two figures, the first representing the overall architecture, and the second the architecture of the MLWB (Machine-Learning Workbench) subsystem.


Legend
Web UI Core Component
Core Component
Supplemental Component
Platform Dependency
External Dependency
Acumos Interface
External Interface
LM LUM Library
SV S-V Library
CD CDS Client
DC Docker Client
NC Nexus Client
Service Provider
Service Client
Host Analytics
Kibana
Operations
Snapshots
Metricbeat
ElasticSearch
ELK Client
Filebeat
Logstash
All core component logs (shared volume)
Docker Registry (e.g. Nexus)
Common Data Service
MariaDB
Acumos platform Nexus-Maven
Docker Engine
Onboarding
Microservice Generation
Acumos Project Nexus
Docker Proxy
/onboarding-app/*
Identity Service
LUM (License Usage Manager)
LUM Database
/oauth/login/username
(nginx authentication callout for NiFi users/admin)
Portal-BE
Scanning Service
Jenkins (sv-scanning job)
User Workstation
User Publisher Admin
Kong or NGINX
/ccds/*
/*
Portal-FE
Federation
Federated Peer
solution.zip with K8S Solution Deployer (deploy.sh)
MLWB (separate diagram)
AcuCompose
Azure Client
Azure
/license-profile-editor/*
License Profile Editor
OpenStack Client
OpenStack
/license-rtu-editor/*
License RTU Editor
Client Libraries and Tools
User Workstation
Kubernetes Client
Solution user (downloads solution.zip)
(to Logstash)
(all NGINX instances)
logs PV
Filebeat
(executes deployment)
/package/*
NGINX
Model Connector
NGINX
model
Solution user
/deployment/*
Deployment Client
Jenkins (solution-deploy job)
kubernetes API
K8S/OpenShift
(sends data to solution)
data source


Per User
/user/*
JupyterHub Proxy
JupyterHub User
(create via k8s API)
/hub/*
JupyterHub (hub,db) CD
/mlwb-notebook-catalog-webcomponent/*
Notebook Catalog WebComponent PR PI NB
/mlwb-project-catalog-webcomponent/*
Project Catalog WebComponent PR PI NB
/mlwb-pipeline-catalog-webcomponent/*
Pipeline Catalog WebComponent PR PI NB
/mlwb-notebook-webcomponent/*
Notebook WebComponent NB
/mlwb-project-webcomponent/*
Project WebComponent PR PI CO PD MO NB
/mlwb-pipeline-webcomponent/*
Pipeline WebComponent PI
/mlwb-dashboard-webcomponent/*
MLWB Dashboard PR PI NB
/mlwb-home-webcomponent/*
MLWB Home
/*
Portal FE
Portal BE
CouchDB
Predictor Service MO PR CO CD
Model Service PR CO CD
Notebook Service CD
Project Service CD
Pipeline Service CD
http://nifi-registry-service/nifi-registry
authuser: admin
http://nifi-service-$USER/nifi-api
authuser: $USER
(create via k8s API)
Per User
http://portal-be-service/oauth/login/username/
Callout for every request to NiFi; returns 200 OK if user is authenticated (convert either (1) Authorization header, or (2) Cookie to authuser header, or return 401 if neither can be validated)
IF portal-be response NOT 200 OK, REDIRECT TO PORTAL
nginx ingress controller
IF PORTAL RETURNS 200 OK
http://nifi-service-$USER/nifi
authuser: $USER
Apache
HTTPS
NiFi
Client Cert CN=$USER
HTTPS
User NiFi Pod
Client Cert CN=admin
User
Browser
IF PORTAL RETURNS 200 OK
http://nifi-registry-service/nifi-registry
authuser: admin
Apache
HTTPS
NiFi Registry
Client Cert CN=admin
NiFi Registry Pod
PVC
PR Project Service Client
PI Pipeline Service Client
NB Notebook Service Client
MO Model Service Client
PD Predictor Service Client
CO CouchDB Client
CD CDS Client
Pods are fronted by service proxy
client
Component Service (proxy)
Component Deployment (pod)

Quickstart Guide to Platform Deployment (TL;DR)

NOTICE:

- if you are deploying to a host/VM and will be executing the prerequisites step of this process, be aware that the process will remove/install software on your target host, and configure it e.g. firewall and security rules. Only execute this process if you understand the implications or are executing the process in a VM/host that you can easily re-create.

- by default, the Acumos platform is deployed with service exposure options typical for development environments. Production environments and especially public environments will need additional planning and restrictions on exposed services, that otherwise could expose your host to security risks. See *Security Considerations* for recommendations on what/how to lock down as needed, for exposure of an AIO-based Acumos platform outside development/test environments.

Please make sure you review the host prerequisite requirements under *Host/VM Preparation*.

See these specific sections based upon how you want to deploy the platform:

- if you have a server/VM or existing k8s cluster upon which you want to install the Acumos platform under k8s, using your local workstation to manage the platform, see **'Deploying from Your Workstation, via the AIO Deployer Tool'_**
- if you have a server/VM upon which you want to directly install/manage the Acumos platform under k8s, see:
  - **'Deploying as a Privileged (sudo) User'_** if you are a sudo user on that server/VM, and want to deploy/manage the platform under your own account
  - *Preparation by Host Admin with Platform Deployment by Normal (non-sudo) User* if you are a sudo user on the server/VM, and want to prepare the server/VM for another to install/manage the platform upon.
- *Docker Based Deployment*, if you want to use the legacy docker-compose based method of installation. NOTE not all Boreas release features are supported under docker-compose.

Kubernetes Based Deployment

The process below will support deployment under either a generic kubernetes distribution, or the OpenShift kubernetes distribution. The scripts will detect which distribution is installed and deploy per the requirements of that distribution. For additional notes on deploying into specific k8s cluster types (e.g. Azure-AKS), see *Deployment notes for specific k8s distributions*.

Note: the following k8s versions are explicitly supported and tested by these tools. Other versions may work, but may require further tool customization.

- "generic" kubernetes 1.13.8
- OpenShift Origin 3.11 ("OKD")

Deploying from Your Workstation, via the aio_k8s_deployer

This process supports users with the role of either a cluster admin (full rights to manage cluster resources) or namespace admin (rights to manage resources under a namespace). It also minimizes any dependencies/customization of the user's workstation, by use of a docker container built specifically for deploying and managing the Acumos platform.

A typical use case for this method is a user who will manage the Acumos platform on the k8s cluster using kubectl executed from their workstation, after basic prerequisites have been arranged by the cluster admin:

- allocation of a namespace
- allocation of a platform FQDN (with DNS registration), external IP, and setup of an ingress controller for the namespace/FQDN
- setup of persistent volume (PV) resources per Acumos requirements (recommended minimum allocations are shown below)
  - logs: 1Gi
  - if deployed as part of the platform, vs use of external instances of these services

        * MariaDB: 10Gi
        * Nexus (Maven repos and docker registry): 10Gi
        * docker-in-docker cache: 10Gi
        * NiFi Registry: 5Gi
        * NiFi users: 5Gi (for each user)
        * JupyterHub: 1Gi
        * Jupyter users: 10Gi (for each user)
        * Elasticsearch: 10Gi

As an option, a user that has cluster admin role can include these prerequisite steps in the process below.

To use this process, the script aio_k8s_deployer.sh in the tools/aio_k8s_deployer folder is used, with these prerequisite steps:

- the user has installed, or has access to, a remote k8s cluster (single/multi-node)
- if the k8s cluster does not provide an ingress service with a registered DNS name for the platform ingress, the user needs to ensure that the external IP address to be used by the ingress controller is registered in DNS or configured in the hosts file of their workstation.
- the user has installed a bash shell and docker on their workstation
- the user has created a folder (referred to here as the "staging" folder) to contain any customizations to be used in this process, as described below
- the user cloned the system-integration repo into the staging folder as subfolder "system-integration"
- the user has prepared any desired customizations in the staging folder, as described under *Customizing the aio_k8s_deployer environment*
- if not providing a k8s config file in the staging folder as described in *Customizing the aio_k8s_deployer environment*, the user has logged into the k8s cluster using the applicable client (kubectl or oc), so that the correct k8s config is present in ~/.kube/config

Given those prerequisites, in the simplest case the deployment can be launched with the single command below:

```
$ bash system-integration/tools/tools/aio_k8s_deployer/aio_k8s_deployer.sh \
  all <host> <user> <k8s distribution> [as-pod=<docker image>]
```

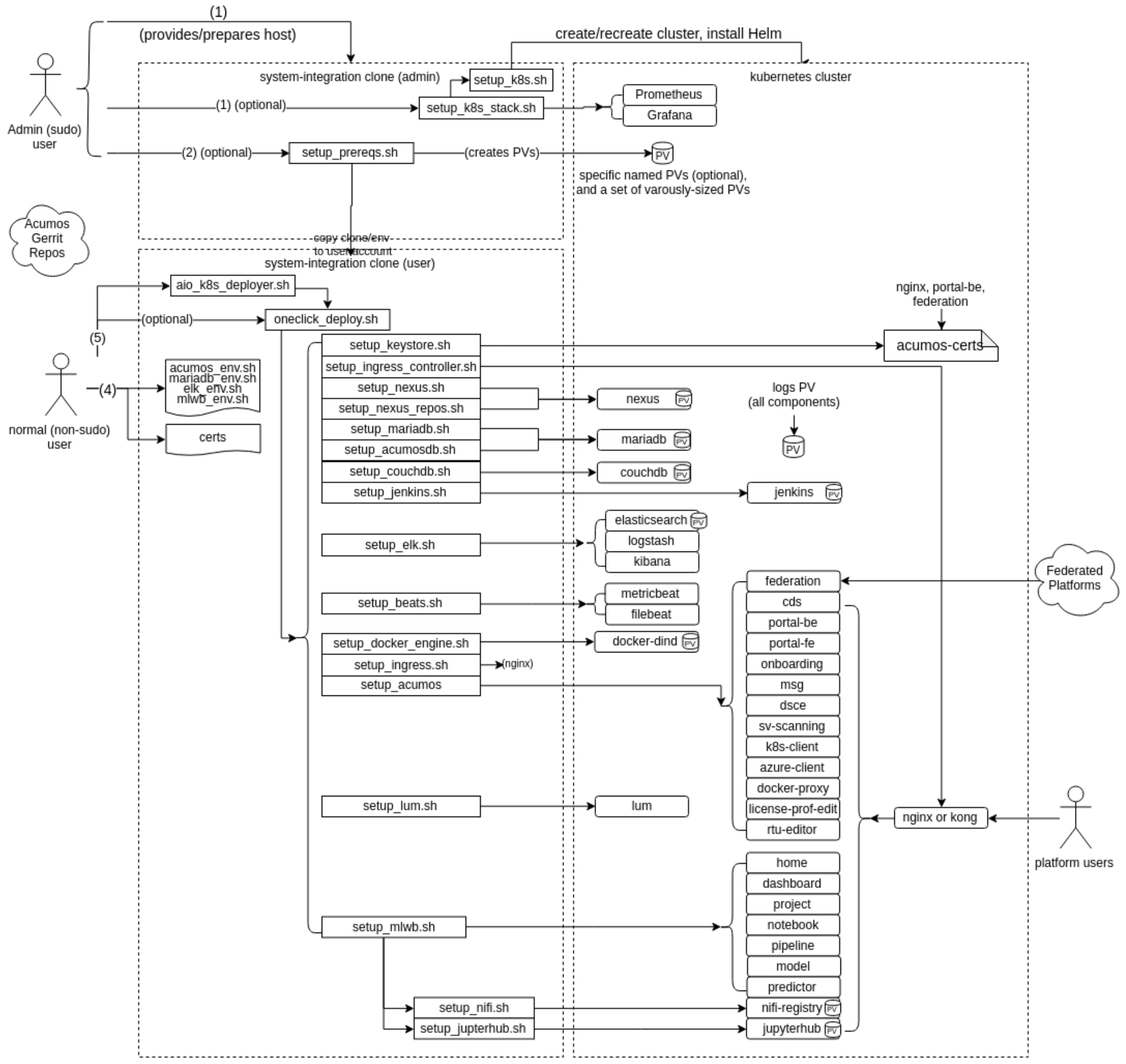
where:

- all: build the acumos_deployer image, prep, and deploy the platform
- host: hostname of k8s cluster master node
- user: SSH-enabled sudo user on the platform
- k8s distribution: type of k8s cluster (genericlopenshift)
- as-pod: (optional) run the oneclick_deploy.sh script from within the cluster as an acumos-deployer pod
  - as-pod provides the advantage of being able to run the deployer under the platform, and use it later as a platform maintenance/debug tool, containing all the customized and configured data for the deployment, as well as any additional tools/data you want to package in it
  - you can specify the image "acumos-deployer" which will be built locally by aio_k8s_deployer.sh, or a pre-built/customized image in a docker registry, e.g. blsaws/acumos-deployer in Docker Hub, which offers tags for use with generic k8s (tag: latest or kubectl) or OpenShift (tag: oc)

When aio_k8s_deployer.sh is called with "all" as the first parameter, it will:

- archive any earlier deployment in folder "aio_k8s_deployer" into a timestamped subfolder of "archived"
- create a new subfolder "aio_k8s_deployer"
- copy everything in the staging folder (except for folders "archived" and "aio_k8s_deployer") as staged customizations into "aio_k8s_deployer/deploy"
- if there is a version of aio_k8s_deployer.sh in the staging folder, copy that to "aio_k8s_deployer" since presumably the user has customized it; otherwise the current version will be copied from the system-integration clone
- kick off the build, prep, and deploy process

The steps above and overall process of the aio_k8s_deployer is shown in the following diagram:



Customizing the aio_k8s_deployer environment

Following are examples of customizations that are possible for the aio_k8s_deployer and the files/configurations it uses. In summary, all aspects of the deployment process are customizable, but the most common are described below. These customizations are made by providing files in the staging folder, or in the system-integration clone under it.

- updating the aio_k8s_deployer.sh script
  - you may find that the script needs to be adapted to your specific docker build or execution environment, thus if a aio_k8s_deployer.sh exists in the staging folder, it will be used in this process
- providing a k8s config file
  - by default, aio_k8s_deployer.sh will copy the user's ~/.kube/config file for use in the acumos_deployer container; in cases which the user cannot use a k8s client directly on their workstation (or doesn't want to), a k8s config file can be provided in advance in the staging folder, as kube-config.
- providing a customize_env.sh script
  - by default, aio_k8s_deployer.sh will customize the default customize_env.sh script to set the three parameters to it, as described above (<host> <user> <distribution>). Any other values in the various environment files (acumos_env.sh, mlwb_env.sh) or environment file creation scripts (setup_mariadb_env.sh, setup_nexus_env.sh, setup_elk_env.sh) can also be updated by using the functions in customize_env.sh to pre-set a non-default value for any of the values in those files.
  - alternatively, the user can prepare the following files in the folders below, which will be supplemented with any unspecified parameters when the related environment setup script is run:
    - mariadb_env.sh in system-integration/charts/mariadb
    - elk_env.sh in system-integration/charts/elk-stack
    - nexus_env.sh in system-integration/AIO/nexus
  - the complete set of customizable environment parameters is described in **'Configuration'_**
- providing a Dockerfile for the desired k8s client
  - by default, aio_k8s_deployer.sh will build a docker image that contains the k8s client for the selected k8s distribution, and other tools needed by the OneClick toolset. You can customize the Dockerfile to be used by providing an updated one based upon the default in system-integration/tools/aio_k8s_deployer/deploy/kubectl (for generic k8s) or system-integration/tools/aio_k8s_deployer/deploy/oc (for OpenShift). Once you have customized the Dockerfile, copy the parent folder (kubectl or oc) to your staging folder.
- providing a post_deploy.sh script
  - aio_k8s_deploery.sh will execute a script named post_deploy.sh in the deploy folder, if present. Such a script can be used for any arbitrary purpose, e.g. to create user accounts, onboard default models, further configure the platform though the available APIs, etc.
- other customizations to the system-integration clone
  - you can provide any other customizations by updating the various scripts, templates, Helm charts, etc in the system-integration clone in the install base folder.
  - starts the acumos-deployer container
  - updates the AIO tools environment to run under the container
  - executes oneclick_deploy.sh, and saves a log
  - executes the post_deploy.sh script, if present

  - copies the updated files from the acumos-deployer container to the user's workstation deploy subfolder, incuding the log files and all updated files under the system-integration repo

Deploying via the Prep-Deploy process

The "Prep-Deploy" process is the original two-stage process which has been automated by the aio_k8s_deployer for k8s based deployments. It still can be used for k8s-based deployments, and is required for docker based deployments.

The process is called Prep-Deploy as it is broken down into two stages:

- host/cluster preparation, executed by a host/cluster admin (sudo user) through the system-integration/AIO/setup_prereqs.sh script
- platform deployment, executed by a normal user through the oneclick_deploy.sh script

These steps are described below.

Using the Prep-Deploy process as a privileged (sudo) user

This process is for a privileged (sudo, not root) user that wants to execute all steps in the deployment process using their host account. To deploy the Acumos platform with the default options, as a user on a linux host with at least 16GB RAM and admin (sudo) permission, follow the process below.

- clone the system-integration repo

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

- using bash, check if the user is part of the docker group, and add if not

```
$ if [[ "$(id -nG "$USER" | grep docker)" == "" ]]; then sudo usermod -aG docker
↪$USER; fi
```

  - if you see "usermod: group 'docker' does not exist", install docker (e.g. using setup_docker.sh in the system-integration/tools folder) and run the command above again. Once you do not see the message above, logout and re-login, to activate your docker group membership.

- if you don't have an existing k8s cluster, run the following command to setup a cluster

  - NOTE: this command will setup a single-node k8s cluster using the generic k8s distribution (for Ubuntu) or OpenShift (for Centos). It also installs docker-ce and links /var/lib/docker to /mnt/docker to avoid out of space issues on the root volume, which can destabilize your k8s cluster. Make sure you have the /mnt folder on a device with adequate disk, e.g. at least 256GB.

```
$ bash system-integration/tools/setup_k8s_stack.sh setup
```

- execute the following command to install/configure prerequisites, including k8s, MariaDB, and the ELK stack, using your user account, and the hostname or domain name you will use to access the deployed platform.

```
$ bash system-integration/AIO/setup_prereqs.sh k8s <domain> $USER
↪<generic|openshift> 2>&1 | tee aio_prep.log
```

- When you see "Prerequisites setup is complete." as the result of the command above, execute the following commands to complete platform setup

```
$ cd system-integration/AIO
$ bash oneclick_deploy.sh 2>&1 | tee aio_deploy.log
```

- The commands above include saving of the detailed deployment actions to a log file 'deploy.txt'. This can be helpful in getting support from the Acumos project team, to overcome issues you might encounter. If you don't want to save the log, just leave out the part of the commands above that starts with the 'pipe' ('|').
- As described above, if you don't need to save the deploy logs, leave out the the part of the commands above that starts with the 'pipe' ('|').
- See *When Deployment is Complete* for further steps

Preparation by Host Admin with Platform Deployment by Normal (non-sudo) User

This process is for a host Admin (sudo user, not root) to prepare the host for a normal (non-sudo) user that will complete the platform deployment, under their account on the host.

- Admin completes steps in the previous section, through setup of a k8s cluster
- Admin executes the following command to install/configure prerequisites, including k8s, MariaDB, and the ELK stack, using their account. <user> in this case is the username of the normal user that will complete the deployment.

```
$ bash system-integration/AIO/setup_prereqs.sh k8s <domain> $USER
↪<generic|openshift> 2>&1 | tee aio_prep.log
```

  - When prerequisites setup is complete, the resulting environment files and system-integration clone will have been copied to the user account.

- The user executes the following commands to complete platform setup

```
$ cd system-integration/AIO
$ bash oneclick_deploy.sh 2>&1 | tee aio_deploy.log
```

- As described above, if you don't need to save the deploy logs, leave out the the part of the commands above that starts with the 'pipe' ('|').
- See *When Deployment is Complete* for further steps

Docker Based Deployment

NOTE: Not all Acumos features will work as expected under docker, so those will not be deployed. Examples include the new services in support of model training.

To deploy the components that do work under docker, follow the instructions in the sections below.

Prerequisites for Docker Based Deployment

Prerequisites for docker based deployment:

- Deployment is supported only on Ubuntu Xenial (16.04), Bionic (18.04), or Centos 7 hosts
- All hostnames or FQDNs specified in environment files must be DNS-resolvable (entries in /etc/hosts or in an actual DNS server)
- User running this script
  - is not running as root
  - has sudo privileges

- has installed docker per system-integration/tools/setup_docker.sh
- has added themselves to the docker group (sudo usermod -aG docker $USER), and re-logged-in to activate docker group membership
- if deploying in preparation for use by a non-sudo user, has created the user account (sudo useradd -m <user>)
- has cloned or otherwise provided the system-integration repo, in the user's home folder
- has customized or created as needed
  - the main environment file system-integration/AIO/acumos-env
  - ELK-stack environment: see **'ELK Stack configuration'_** as a guide to what environment values can be customized. Customize the default values in that script, by changing the values after ':-" e.g. to change "true" to "false" replace the first line below with the second
    - export ACUMOS_DEPLOY_METRICBEAT="${ACUMOS_DEPLOY_METRICBEAT:-true}"
    - export ACUMOS_DEPLOY_METRICBEAT="${ACUMOS_DEPLOY_METRICBEAT:-false}"
  - MariaDB: as for the ELK_stack, customize system-integration/charts/mariadb/setup_mariadb_env.sh

Deploying for Yourself, as a Host Admin (sudo user)

NOTE: If you are deploying into an Azure-based VM, pay attention to this special configuration need for the docker-engine; update the acumos_env.sh (in system-integration/AIO) script to set the ACUMOS_DEPLOY_DOCKER_DIND flag to "false", which will ensure that the docker-dind service is not installed. Docker-dind has known issues under Azure.

```
export ACUMOS_DEPLOY_DOCKER_DIND=false
```

If deploying the platform for yourself, run these commands:

```
cd system-integration/AIO/
bash setup_prereqs.sh docker <domain> $USER 2>&1 | tee aio_deploy.log
bash oneclick_deploy.sh 2>&1 | tee -a aio_deploy.log
```

- where:
  - <domain> is the name you want to use for the Acumos portal. This can be a hostname or FQDN.
- See *When Deployment is Complete* for further steps

Preparing as a Host Admin, with Platform Deployment as a Normal User

If a Host Admin needs to run the privileged-user steps for a normal user that will take it from there:

- NOTE: If you are deploying into an Azure-based VM, pay attention to this special configuration need for the docker-engine; update the acumos_env.sh (in system-integration/AIO) script to set the ACUMOS_DEPLOY_DOCKER_DIND flag to "false", which will ensure that the docker-dind service is not installed. Docker-dind has known issues under Azure.

```
export ACUMOS_DEPLOY_DOCKER_DIND=false
```

- As the Host Admin, run these commands:

```
cd system-integration/AIO/
bash setup_prereqs.sh docker <domain> <user> 2>&1 | tee aio_deploy.log
```

- where:
  - <domain> is the name you want to use for the Acumos portal. This can be a hostname or FQDN.
  - <user> use the normal user's account name on the host
- As the normal user, run this command

```
bash oneclick_deploy.sh 2>&1 | tee -a aio_deploy.log
```

- As described above, if you don't need to save the deploy logs, leave out the the part of the commands above that starts with the 'pipe' ('|').

When Deployment is Complete

When deployment has completed, you should see a success message with a set of URLs to access the various platform services. You can also view the file "acumos.url" which will be in the system-integration/AIO folder (example below)

```
You can access the Acumos portal and other services at the URLs below,
assuming hostname "acumos.example.com" is resolvable from your workstation:

Portal: https://acumos.example.com
Common Data Service Swagger UI: https://acumos.example.com/ccds/swagger-ui.html
- if you have issues with using the CDS swagger over HTTPS, try the HTTP link
  http://$ACUMOS_DOMAIN:$ACUMOS_CDS_NODEPORT/ccds/swagger-ui.htm
Portal Swagger UI: https://acumos.example.com/api/swagger-ui.html
Onboarding Service Swagger UI: https://acumos.example.com/onboarding-app/swagger-ui.
↪html
Kibana: http://acumos.example.com:30561/app/kibana
Nexus: http://acumos.example.com:30881
```

By default, the platform is not configured to require email confirmation of new accounts, so you can create a new account directly on the Portal home. To create an account with the Admin role (needed for various platform admin functions), use the create_user.sh script in the system-integration/tests folder

4.1.2 Release Scope

Current Release (Clio)

The Acumos wiki describes the principle goals and related deployment scenarios supported by the AIO toolset, and regularly verified in testing.

What's included in the AIO tools

In system-integration repo folder AIO:

- setup_prereqs.sh: Script to be used by a host admin (a user with privilege to install applications and configure the host) to prepare a host for a normal user to later deploy/manage the Acumos platform there. Typically used for lab environments.
- oneclick_deploy.sh: the main script that kicks off the deployment, to setup an AIO instance of Acumos under a docker or kubernetes environment.

- acumos_env.sh: environment setup script 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.
- setup_acumosdb.sh: script that initializes the Acumos database under MariaDB.
- setup_keystore.sh: script that enables use of pre-configured CA and server certificates for an Acumos platform, or creation of new self-signed certificates.
- docker_compose.sh: Script called by the other scripts as needed, to take actions on the set of Acumos docker services. Used by oneclick_deploy.sh and clean.sh for docker-based deployments. You can also call this directly e.g. to tail the service container logs. See the script for details.
- utils.sh: utility script containing functions used by many of these scripts.
- redeploy_component.sh: Script that allows the redeployment of a single component.
- clean.sh: if needed, this script allows a privileged user to remove all components and dependencies of the Acumos platform installed by the tools above.

In AIO/beats:

- deployment scripts and templates for the Filebeat and Metricbeat services as ELK stack components deployed along with the Acumos platform.

In AIO/certs:

- setup_certs.sh: creates self-signed certificates (CA and server), keystore, and truststore for use by core platform components.
- This folder is also used to stage user-provided certs to be used in Acumos platform deployment.

In AIO/docker:

- docker-compose yaml files and deployment script for Acumos core components.

In AIO/docker-engine:

- scripts and templates to deploy docker-in-docker as the docker-engine service for k8s-based Acumos platforms, or the docker-engine service on the AIO host

In AIO/docker-proxy:

- scripts and templates for deployment of the docker-proxy core component of the Acumos platform

In AIO/elk-stack:

- scripts and templates to deploy the ELK stack core components under docker

In AIO/ingress:

- scripts and templates to deploy the NGINX Ingress Controller for Kubernetes, and ingress rules for Acumos core components.

In AIO/jenkins:

- script to deploy Jenkins as a service under k8s, supporting solution deployment and security verification functions for the Acumos platform.

In AIO/kong:

- scripts and templates to deploy the Kong service as an ingress controller for the Acumos platform, as deployed under docker or k8s

In AIO/kubernetes:

- under deployment, kubernetes deployment templates for all system components
- under service, kubernetes service templates for all system components

- under configmap, kubernetes configmap templates for all system components
- under rbac, kubernetes role-based access control templates enabling system components to invoke kubernetes cluster operations

In AIO/lum:

- scripts and templates to deploy the License Management components under k8s

In AIO/mariadb:

- scripts and templates to deploy MariaDB, as the Acumos platform database backend service

In AIO/mlwb:

- scripts and templates to deploy the MWLB components of the Acumos platform

In AIO/nexus:

- scripts and templates to deploy the Nexus service for the Acumos platform

In charts:

- scripts and templates to deploy the following components for k8s-based deployments, using Helm as deployment tool
  - couchdb: CouchDB service as used by the MLWB
  - elk-stack: ELK stack core components
  - ingress: Nginx-ingress controller
  - jenkins: the Jenkins service as used by the Deployment Client and SV Scanning Service
  - jupyterhub: the JupterHub/JupyterLab services for notebook-based model development
  - mariadb: MariaDB service
  - zeppelin: the Zeppelin service for notebook-based model development
    - NOTE: Zeppelin deployment is a single, multi-user instance which is provided for experimental use in Boreas. Single-user instance deployment is coming in the next release (Clio).

In tests:

- bootstrap_models.sh: Model package onboarding via curl, for all models in a folder.
- create_peer.sh: Automated setup of a peer relationship between two Acumos AIO deployments.
- create_subscription.sh: creates a federation subscription for all models published by a federated Acumos platform.
- create_user.sh: Automated user provisioning and role assignment. Used by scripts in this repo to create default admin accounts. Can also be used to create user accounts for testing or platform use.
- delete_user.sh: deletion of a user account
- license_scan.sh: invokes a license scan for a solution, using the Security Verification Scanning Service.
- onboard_model.sh: Model package onboarding via curl.
- peer_test.sh: Peering and marketplace subsciptions setup for two AIO platforms. Used to test federation use cases.

In tools:

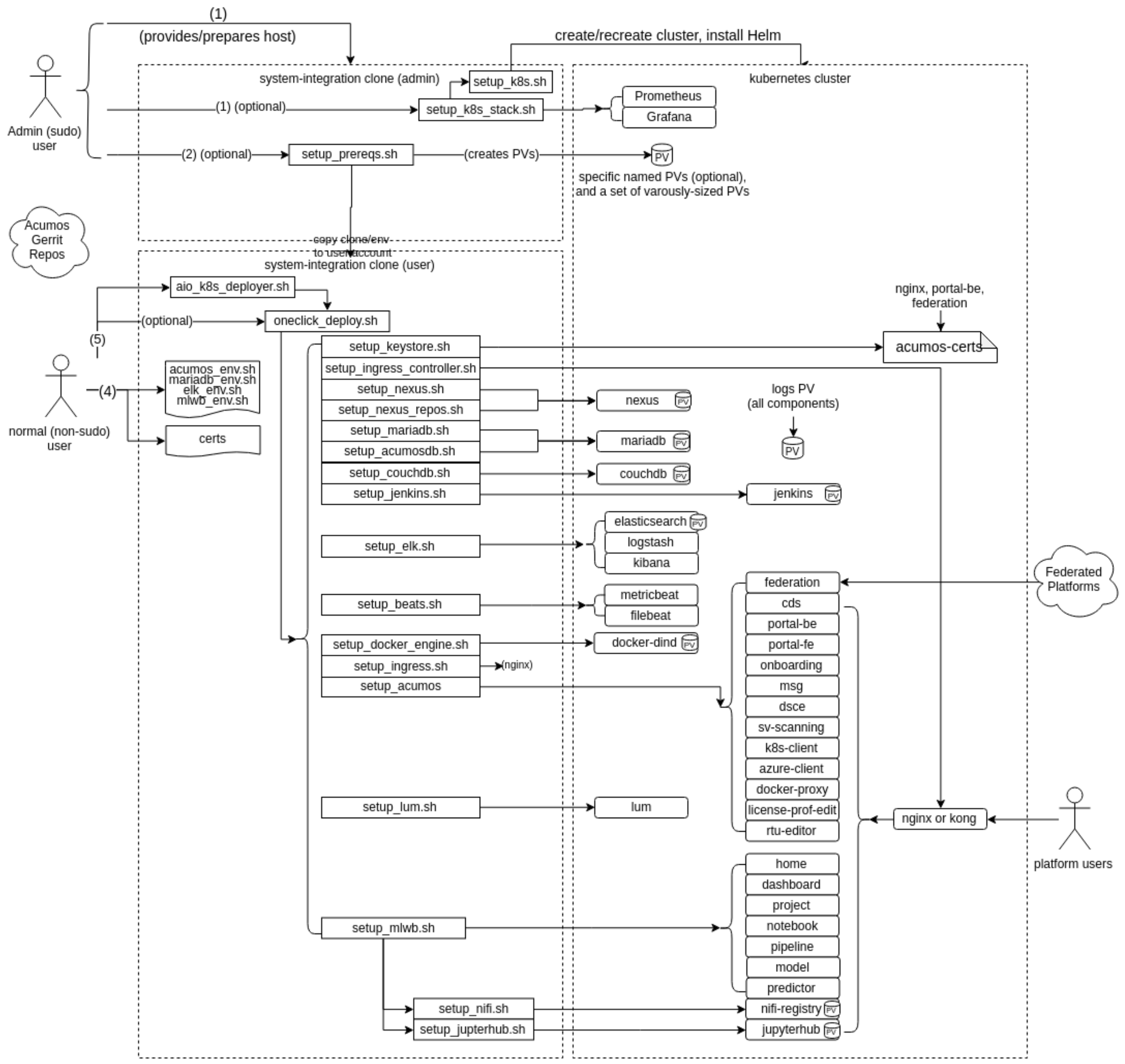
- aio_k8s_deployer: deployment script and configuration to deploy Acumos under k8s using a docker container based approach, which minimizes dependencies on the user workstation

- add_host_alias.sh: adds a host alias to an Acumos core component, e.g. for hostnames/FQDNs that are not resolvable through DNS.
- setup_docker.sh: deploys the docker version used for docker-based platform deployment and interaction.
- setup_helm.sh: deploys Helm as a service deployment tool.
- setup_k8s.sh: deploys a generic k8s cluster.
- setup_kubectl.sh: deploys and uses the kubectl tool used by other scripts and the user to manage and interact with generic k8s based deployments.
- setup_mariadb_client.sh: deploys the MariaDB client as used by other scripts to configure the Acumos database.
- setup_openshift.sh: deploys an OpenShift Origin 3.11 kubernetes cluster, for subsequent Acumos platform deploymet on Centos 7 servers.
- setup_openshift_client.sh: deploys the OpenShift client (oc) tool used by other scripts and users to manage and interact with OpenShift based platform deployments.
- setup_prometheus.sh: deploys the Prometheus monitoring service, with Grafana as a data visualization tool, for monitoring the Acumos platform's resources at the k8s level. Also deploys Grafana dashboards in the dashboards folder.
- setup_pv.sh: deploys host-based persistent volumes for use with docker and k8s-based platform deployments.
- trust_cert.sh: Adds a cert to the Acumos truststore under k8s
- update_env.sh: Compares the environment variable set between two AIO versions, and updates the latter version with the values as set in the previous version, for use in upgrading a platform to the new version

4.1.3 Deployment Step-by-Step Guide

The steps in this process are illustrated by the following figure. Note this figure refers to kubernetes, but the same basic process applies for docker.

1. Host Admin prepares the platform host environment, per *Host/VM Preparation* and *Install Host Preparation by Admin*
2. Host Admin clones the system-integration repo, or uses a local/customized clone, and runs the applicable host/environment preparation script(s) as described in *Install Host Preparation by Admin*
3. The user (Admin, if installing for their self) customizes the environment files and/or certs (per *Pre-Arrangement of Ingress Certs*) as desired, either manually or through a 'customize_env.sh' script as described in **'Deploying from Your Workstation, via the AIO Deployer Tool'_**,
4. The user deploys the platform components via the 'aio_k8s_deployer.sh' script or via the 'oneclick_deploy.sh' script
    - 'aio_k8s_deployer.sh' provides a convenient docker-based wrapper environment for running 'oneclick_deploy.sh', making it easy to execute k8s-based Acumos platform deployment on any type of workstation, and snapshot the resulting deployment tools state for later use/sharing.



Host/VM Preparation

For developer-focused AIO deployments, it's assumed that the developer has a minimum of one host machine (physical workstation/server or VM) that they will use for the platform.

The AIO platform is also deployable on a cluster of machines or in a multi-node kubernetes cluster, but note:

- for docker-based deployment, the AIO toolset supports limited distribution of components across nodes, primarily the backend services (Nexus, MariaDB, ELK, . . .), in addition to the core platform components in one node
- for kubernetes-based deployment, the components will be distributed across nodes in a kubernetes cluster per the default scheduling configuration of the cluster

Following are basic requirements for single-node/AIO machines:

- minimum 16 GB RAM (32 GB or more recommended)

- minimum 2 core/vCore (4 or more recommended)
- minimum 1 network interface
- network security rules in place to allow incoming traffic on the following ports:

Install Host Preparation by Admin

NOTE: If you are deploying under k8s into an Azure-based VM, pay attention to the special configuration need for the docker-engine, as described below.

Prerequisites:

- Ubuntu Xenial/Bionic or Centos 7 server
- All hostnames specified in acumos_env.sh must be DNS-resolvable on all hosts (entries in /etc/hosts or in an actual DNS server)
- For deployments behind proxies, set ACUMOS_HTTP_PROXY and ACUMOS_HTTPS_PROXY in acumos_env.sh
- Admin user running this script has:
  - Installed docker per system-integration/tools/setup_docker.sh
  - Added themselves to the docker group (sudo usermod -aG docker $USER)
  - Logged out and back in, to activate docker group membership
- Initial basic setup (manual)
  - If you are an Admin and deploying the platform for a normal user, assuming the non-sudo user is "acumos"

    ```
    sudo useradd -m acumos
    ```

This process prepares the host with prerequisites that normal users do not have permission to arrange. This includes:

- installing software packages
- configuring host settings
- creating folders for host-mapped volumes

The Admin user will follow this process:

- 'install root folder' refers to the Admin user's home folder. Installation in other root folders is a work in progress, and not yet fully verified.
- create in the install root folder a subfolder "acumos" and folders "env", "logs", "certs" under it.
- in the install root folder, clone the system-integration repo (branch, tag, commit, or master), and make any desired updates to it (e.g. checkout a specific patch)
- if you are installing under k8s and don't have a pre-installed k8s cluster, install a cluster e.g. using the setup_k8s_stack.sh script (in system-integration/tools).
- If you are deploying the platform under k8s in an Azure VM, update acumos_env.sh (in system-integration/AIO) script to set the ACUMOS_DEPLOY_DOCKER_DIND flag to "false", which will ensure that the docker-dind service is not installed. Docker-dind has known issues under Azure.

  ```
  export ACUMOS_DEPLOY_DOCKER_DIND=false
  ```

- If you are deploying under docker, run the command

```
bash setup_prereqs.sh <under> <domain> <user>
```

- under: docker (install prereqs for docker or k8s based deployment)
- domain: FQDN of platform
- **user: user that will be completing Acumos platform setup via** oneclick_deploy.sh (if installing for yourself, use $USER)
- If you are deploying under k8s, and do not have an existing k8s cluster or need to deploy a new cluster e.g. an AIO cluster on a VM, run the command below on the host for the new cluster

  ```
  bash system-integration/tools/setup_k8s_stack.sh setup
  ```

- If you are deploying under k8s, run the command

  ```
  bash system-integration/AIO/setup_prereqs.sh k8s <domain> $USER
  ↪<generic|openshift>
  ```

  - k8s: indicates deployment under k8s
  - user: non-sudo user account (use $USER if deploying for yourself)
  - domain: domain name of Acumos platorm (resolves to this host)
  - genericlopenshift: use generic k8s or openshift

When the process is complete, the updated system-integration clone and environment will have been copied to the platform deployment user's home folder. If you are deploying the platform for yourself, proceed to the next section. If preparing the platform for a normal user, the user should execute the process in the next section.

Pre-Arrangement of Ingress Certs

If you deploy the AIO platform often or in multiple test environments, you may find it useful to pre-arrange the ingress certs that will be used to access the platform(s), either using commercial cert providers, or the self-signed cert tools provided in the system-integration repo. This allows you and users for example to use web browser tools to trust self-signed certs, avoiding browser warnings.

The Acumos tool supporting creation of self-signed certs is in system-integration/AIO/certs/setup_certs.sh. An example is given below, showing how to select the parameters for setting up a cert and related files that are usable on multiple hosts, by hostname/domain and IP address:

- This script is invoked as:

```
bash setup_certs.sh <name> <subject-name> ["alt-names"] ["alt-ips"]
```

Where:

- name: name prefix to use in the generated files (e.g. acumos)
- subject-name: primary domain name to associate
- alt-names: quoted, space-delimited set of alternate names
- alt-ips: quoted, space-delimited set of alternate IP addresses

as in the example:

```
cd system-integration/AIO/certs
bash setup-certs.sh acumos acumos \
  "test01 test02 test03 acumos-test04.eastus.cloudapp.azure.com" \
  "10.1.0.2 10.1.0.3 10.1.0.4 10.1.0.5"
```

This will generate the following files:

- acumos-ca.key: self-signed CA private key
- acumos-ca.crt: self-signed CA certificate
- acumos.key: server cert private key
- acumos.crt: server cert
- acumos-keystore.p12: PKCS12 format keystore with server cert
- acumos-truststore.jks: JKS format truststore with CA cert
- cert_env.sh: environment file with the related passwords

NOTE: the process below has not been verified. If you need to following this process and encounter issues, reach out to the Acumos Community mail list for help.

To use commercial certs with the Acumos AIO platform, follow these steps:

- place the server cert and private key in folder system-integration/AIO/certs
- update related values in system-integration/AIO/acumos_env.sh and put these commands into a file system-integration/AIO/cert_env.sh
  - export ACUMOS_CERT_PREFIX=<prefix you want to use for your keystore/truststore
  - export ACUMOS_CERT=<name of the server cert file)
  - export ACUMOS_CERT_KEY=<name of the server cert private key file>
  - export ACUMOS_CERT_KEY_PASSWORD=<passphrase for the cert private key>
- run the commands below, which create the keystore and truststore for Acumos

```
cd system-integration/AIO/certs
source cert_env.sh
KEYSTORE_PASSWORD=$(uuidgen)
echo "export KEYSTORE_PASSWORD=$KEYSTORE_PASSWORD" >>cert_env.sh
openssl pkcs12 -export \
  -in $ACUMOS_CERT \
  -inkey $ACUMOS_CERT_KEY \
  -passin pass:$CERT_KEY_PASSWORD \
  -certfile $ACUMOS_CERT \
  -out $ACUMOS_CERT_PREFIX-keystore.p12 \
  -passout pass:$KEYSTORE_PASSWORD

TRUSTSTORE_PASSWORD=$(uuidgen)
echo "export TRUSTSTORE_PASSWORD=$TRUSTSTORE_PASSWORD" >>cert_env.sh
keytool -import \
  -file $ACUMOS_CERT \
  -alias $ACUMOS_CERT_PREFIX-ca \
  -keystore $ACUMOS_CERT_PREFIX-truststore.jks \
  -storepass $TRUSTSTORE_PASSWORD -noprompt
```

Platform Deployment

The script supporting this step is system-integration/AIO/oneclick_deploy.sh.

Prerequisites:

- User workstation is Ubuntu Xenial/Bionic, Centos 7, or MacOS
- prerequisites setup via setup_prereqs.sh, or otherwise provided

This process deploys the Acumos platform with options selected by the user, e.g.

- as described in the **'Acumos OneClick / All-in-One (AIO) Configuration Guide'_**
- use of pre-created certs per *Pre-Arrangement of Ingress Certs*

To deploy the platform components, run the commands:

```
cd system-integration/AIO
bash oneclick_deploy.sh
```

When the process is complete, you will see a set of URLs to the main platform component/UI features, as described above.

Updating Configuration and Components

Changes to the configuration can be applied as described in the previous section. Note that if you are making changes to the configuration of a deployed platform, some changes may break some aspects of the platform, so be careful.

- docker-compose templates in AIO/docker/acumos or kubernetes templates in AIO/kubernetes
  - Note: make sure the template modifications are compatible with previously deployed components, and the version of the related Acumos component you are deploying/re-deploying

Update options and tools related to specific components are described in the following sections.

Two supported options for updating Jenkins are:

- upgrading the Helm release (deployed Helm chart), using an updated values.yaml file in system-integration/charts/jenkins/deploy/values.yaml

```
source system-integration/AIO/acumos_env.sh
helm upgrade $ACUMOS_NAMESPACE-jenkins stable/jenkins \
  -f system-integration/charts/jenkins/deploy/values.yaml
```

- adding a host alias, e.g. so a Jenkins job can contact a host that does not have a DNS-resolvable hostname; examples include a kubernetes cluster to be used with a solution-deploy job, or a code scanning service to be used with a security-verification-scan job.

```
bash system-integration/charts/jenkins/setup_jenkins.sh alias:<HOST>:<IP>
```

  - where
    - HOST: hostname/FQDN of the system to add as a hostAlias
    - IP: IP address of the system
  - after you have added the alias and Jenkins has restarted, you will need to login to the Jenkins UI at https://<ACUMOS_DOMAIN>/jenkins/, and execute the initial-setup job

Stopping, Restarting, Redeploying

Note: the following sections assume that you have deployed the Acumos platform from the system-integration folder in your user home directory, i.e. "~/".

If you just want to redeploy Acumos components, without affecting any data in the MariaDB or Nexus, be sure to set these variables in AIO/acumos_env.sh:

```
export ACUMOS_DEPLOY_MARIADB=false
export ACUMOS_SETUP_DB=false
export ACUMOS_DEPLOY_NEXUS=false
```

To stop components running under docker and remove the containers, execute the following commands from the "docker" folder related to the type of component, referencing the related docker-compose yaml file as "<yml>":

```
cd ~/system-integration/<docker folder>
source ~/system-integration/AIO/acumos_env.sh
docker-compose -f acumos/<yml> down
```

The related docker folders are:

- AIO/docker, for Acumos core components azure-client, common-data-svc, dsce (AcuCompose), federation, kubernetes-client, microservice-generation, onboarding, portal-be, portal-fe, sv-scanning
- AIO/docker-proxy/docker, for the docker-proxy core component
- AIO/mlwb/docker, for the MLWB components
- AIO/nexus/docker, for nexus
- AIO/mariadb/docker, for mariadb
- AIO/kong/docker, for kong
- AIO/elk-stack/docker, for the core ELK-stack components elasticsearch, logstash, kibana
- AIO/beats/docker, for the "beats" components filebeat, metricbeat

To restart these components, e.g. after updating the related configuration files, issue the following command:

```
cd ~/system-integration/<docker folder>
source ~/system-integration/AIO/acumos_env.sh
docker-compose -f acumos/<yml> up -d --build
```

If you want to automatically stop and redeploy the components in one command:

- for Acumos core components (azure-client-service, cds-service, dsce-service, federation-service, kubernetes-client-service, msg-service, onboarding-service, portal-be-service, portal-fe-service, sv-scanning-service)

  ```
  bash ~/system-integration/AIO/redeploy_component.sh <component>
  ```

- for the other components, a specific redeployment script is provided in the related folder (docker-proxy, mlwb, nexus, mariadb, kong, elk-stack, beats)

  ```
  bash ~/system-integration/AIO/<folder>/setup_*.sh  ~/system-integration/AIO/
  ```

Because kubernetes-based components may depend upon a variety of other kubernetes resources specific to them or shared with other components (e.g. configmaps, secrets, PVCs), simply redeploying the specific components after any required configuration updates is recommended.

The configuration files specific the components are generally under a subfolder "kubernetes", and are specific to the type of resource (e.g. service, deployment, configmap, secret, PVC, etc). Once you have updated these as needed, you can' redeploy the component and any resources specific to it (not shared) via the command:

- for core components under AIO/kubernetes/deployment, using the component names per the "app:" value in the related deployment template (azure-client, cds, dsce, federation, kubernetes-client, msg, onboarding, portal-be, portal-fe, sv-scanning):

  ```
  bash ~/system-integration/AIO/redeploy_component.sh <component>
  ```

- for the other components, running the related "setup_*.sh" command as described for docker

If you just need to stop a component, use the following command and reference the related "app" label:

```
kubectl delete deployment -n acumos -l app=<app>
```

You can see all the component-related "app" labels via the command:

```
kubectl get deployment -n acumos -o wide
```

After stopping the component, you can redeploy it as needed using the methods described above.

Deployment notes for specific k8s distributions

Azure supports its own variant of the generic k8s distribution, and has some features that require the following pre-requisites, adaptations, and specific steps in Acumos platform deployment, at least as tested/supported so far with the OneClick toolset.

- creation of namespace(s) as needed for the Acumos core platform and Nexus (if not externally deployed already)
- (recommended) pre-registered DNS names (A-records) for the Acumos platform services below, associated with IP addresses in the Azure resource group for the target k8s cluster:
  - Acumos nginx-ingress controller service
  - Federation service
  - Nexus service (if deployed as part of the platform)

In all cases, if using the aio_k8s_deployer, the following settings are recommended to be provided in customize_env.sh:

```
...
update_acumos_env ACUMOS_DEPLOY_AS_POD true
update_acumos_env ACUMOS_NAMESPACE <namespace>
update_acumos_env DEPLOYED_UNDER k8s
update_acumos_env K8S_DIST generic
update_acumos_env ACUMOS_HOST_USER $USER
update_acumos_env ACUMOS_DOMAIN <Acumos Portal FQDN>
update_acumos_env ACUMOS_HOST $ACUMOS_DOMAIN
update_acumos_env ACUMOS_CERT_PREFIX $ACUMOS_DOMAIN
# Use controller.service.loadBalancerIP       for nginx-ingress helm chart
update_acumos_env ACUMOS_INGRESS_LOADBALANCER true
# Azure provides dynamic PV allocation, so no need to create PVs explicitly
update_acumos_env ACUMOS_CREATE_PVS false
update_acumos_env ACUMOS_PVC_TO_PV_BINDING false
# Set to true to clean any existing PV data for related PVCs (default: false)
update_acumos_env ACUMOS_RECREATE_PVC false

update_nexus_env ACUMOS_NEXUS_DATA_PV_SIZE 100Gi
```

```
# Azure-AKS is incompatible with Acumos log collection design (for now)
update_acumos_env ACUMOS_DEPLOY_ELK false
update_acumos_env ACUMOS_DEPLOY_ELK_FILEBEAT false
# Disable use of log PVs

function clean_yaml() {
  for f in $fs; do
    for s in $ss; do
      sed -i -- "/$s/d" $1/$f.yaml
    done
  done
}
ss="volumeMounts logs volumes persistentVolumeClaim claimName"
fs="azure-client-deployment cds-deployment deployment-client-deployment dsce-
↪deployment kubernetes-client-deployment license-profile-editor-deployment license-
↪rtu-editor-deployment msg-deployment onboarding-deployment portal-fe-deployment sv-
↪scanning-deployment"
clean_yaml system-integration/AIO/kubernetes/deployment
fs="mlwb-dashboard-webcomponent-deployment mlwb-model-service-deployment mlwb-
↪predictor-service-deployment mlwb-home-webcomponent-deployment mlwb-notebook-
↪webcomponent-deployment mlwb-pipeline-catalog-webcomponent-deployment mlwb-pipeline-
↪webcomponent-deployment mlwb-project-service-deployment mlwb-project-catalog-
↪webcomponent-deployment mlwb-project-webcomponent-deployment"
clean_yaml system-integration/AIO/mlwb/kubernetes
ss="logs persistentVolumeClaim claimName"
fs="portal-be-deployment federation-deployment"
clean_yaml system-integration/AIO/kubernetes/deployment
fs="nifi-registry-deployment"
clean_yaml system-integration/AIO/mlwb/nifi/kubernetes
fs="mlwb-notebook-service-deployment mlwb-pipeline-service-deployment"
clean_yaml system-integration/AIO/mlwb/kubernetes

ss="logs persistentVolumeClaim claimName"
fs="docker-proxy-deployment"
clean_yaml system-integration/AIO/docker-proxy/kubernetes
sed -i -- '/mountPath: \/var\/log\/acumos/d' system-integration/AIO/docker-proxy/
↪kubernetes/docker-proxy-deployment.yaml
```

If using the aio_k8s_deployer, the pre-registered domain names for the services describe above just need to be included along with the following other recommended options in customize_env.sh, as shown below.

```
...
update_acumos_env ACUMOS_FEDERATION_DOMAIN <Acumos Federation service FQDN>
update_nexus_env ACUMOS_NEXUS_DOMAIN <Nexus FQDN>
```

Given those settings and the base options for customize_env.sh described above, an example scripted process for a clean install of the Acumos platform is:

```
#!/bin/bash
set -x -e
WORK_DIR=$(pwd)
mkdir -p deploy
cp customize_env.sh deploy/customize_env.sh
echo "Copy kube-config"
cp ~/.kube/config deploy/kube-config
# Prerequisite: save helm/tiller version compatible with your Azure-AKS
echo "Copy helm"
wget https://get.helm.sh/helm-v2.12.3-linux-amd64.tar.gz | unzip
```

```
tar -xvf helm-v2.12.3-linux-amd64.tar
cp linux-amd64/helm deploy/.
cp linux-amd64/tiller deploy/.
echo "Clone system-integration"
rm -rf deploy/system-integration
git clone "https://gerrit.acumos.org/r/system-integration" deploy/system-integration
# optional: env scripts to be copied into system-integration/AIO/ by customize_env.sh
rm deploy/env/*
cp -r env deploy/.
# optional: pre-arranged certs to be copied into system-integration/AIO/certs by␣
↪customize_env.sh
cp -r certs deploy/.
cp deploy/system-integration/tools/aio_k8s_deployer/aio_k8s_deployer.sh .
bash aio_k8s_deployer.sh deploy as-pod=blsaws/acumos-deployer:latest
```

This process is appropriate for test environments or platforms for which you will register FQDNs for the allocated loadBalancer IP addresses, after deployment of the platform. The example process below modifies the example above by:

- pre-creating certs for the platform, without specifying the allocated IP address in the cert (optional, anyway). This is required since:
  - the assigned ingress IP address is not known until it is allocated, and
  - it will not be allocated until the nginx-ingress controller is deployed, but
  - the deployment of the controller depends upon the applicable cert/key being specified in the deployment process… a "catch-22"
- installing the ingress controller first, so that the chosen platform domain name and allocated ingress IP address can be used in adding hostAlias entries for all platform components that need to access the platform via the ingress controller; this happens though use of the "add-host" option of aio_k8s_deployer.sh.

```
#!/bin/bash
set -x -e
ACUMOS_NAMESPACE=acumos-prod
ACUMOS_DOMAIN=acumos-prod.westus.cloudapp.azure.com
ACUMOS_FEDERATION_DOMAIN=federation-$ACUMOS_DOMAIN
WORK_DIR=$(pwd)
mkdir -p deploy
cp customize_env.sh deploy/customize_env.sh
echo "Copy kube-config"
cp ~/.kube/config deploy/kube-config
# Prerequisite: save helm/tiller version compatible with your Azure-AKS
echo "Copy helm"
wget https://get.helm.sh/helm-v2.12.3-linux-amd64.tar.gz | unzip
tar -xvf helm-v2.12.3-linux-amd64.tar
cp linux-amd64/helm deploy/.
cp linux-amd64/tiller deploy/.
echo "Clone system-integration"
rm -rf deploy/system-integration
git clone "https://gerrit.acumos.org/r/system-integration" deploy/system-integration
# optional: env scripts to be copied into system-integration/AIO/ by customize_env.sh
rm deploy/env/*
cp -r env deploy/.
cd deploy/system-integration/AIO/certs/
bash setup_certs.sh $ACUMOS_DOMAIN "$ACUMOS_FEDERATION_DOMAIN"
cd $WORK_DIR
cp -r deploy/system-integration/AIO/certs deploy/.
```

```
sed -i -- 's/ACUMOS_INGRESS_LOADBALANCER=false/ACUMOS_INGRESS_LOADBALANCER=true/' \
  deploy/system-integration/AIO/acumos_env.sh
certs="$(pwd)/deploy/system-integration/AIO/certs"
bash deploy/system-integration/charts/ingress/setup_ingress_controller.sh \
  $ACUMOS_NAMESPACE $certs/$ACUMOS_DOMAIN.crt $certs/$ACUMOS_DOMAIN.key
while [[ "$(kubectl get svc -n $ACUMOS_NAMESPACE $ACUMOS_NAMESPACE-nginx-ingress-
↪controller -o json | jq -r ".status.loadBalancer.ingress[0].ip")" == 'null' ]]; do
  sleep 10
done
ACUMOS_DOMAIN_IP=$(kubectl get svc -n $ACUMOS_NAMESPACE $ACUMOS_NAMESPACE-nginx-
↪ingress-controller -o json | jq -r ".status.loadBalancer.ingress[0].ip")
sed -i -- 's/ACUMOS_DEPLOY_INGRESS true/ACUMOS_DEPLOY_INGRESS false/' deploy/
↪customize_env.sh
# env scripts to be copied into system-integration/AIO/. by customize_env.sh
cp -r env deploy/.
cp deploy/system-integration/tools/aio_k8s_deployer/aio_k8s_deployer.sh .
bash aio_k8s_deployer.sh deploy as-pod=blsaws/acumos-deployer:latest add-host=$ACUMOS_
↪DOMAIN:$ACUMOS_DOMAIN_IP
```

4.1.4 Logs Location

If the ELK stack is deployed, logs should be available through the Kibana UI. If you want to look at the logs on the platform or through the k8s client API, use one of the methods below. Note that while generally the Acumos logs design pattern is to create logs under a subfolder of the logs folder, not all components yet support that fully. So you will see some miscellaneous log files associated with those components.

- for docker-based deployments, the logs are easily accessible on the AIO host under /mnt/<ACUMOS_NAMESPACE>/logs directory ('<ACUMOS_NAMESPACE>' is by default 'acumos'). That directory is mounted by most Acumos components as their log directory.
- for k8s-based deployments, logs in most cases will be available through one or more PVCs.
  - If you did not customize the SERVICE_LABEL values for the components, by default the logs can be located under the "logs" PVC, e.g.:

    ```
    kubectl get pvc logs -o yaml | awk '/volumeName/{print $2}'
    pv-10gi-3
    ```

    * In the example above, if you are using the default hostPath-based PV service setup by the OneClick tools for a single-node k8s cluster, and you have access to the k8s master node, you can find the logs under /mnt/$ACUMOS_NAMESPACE/<pv name>, e.g.

      ```
      $ pvdir=$(kubectl get pvc logs -n $ACUMOS_NAMESPACE -o yaml | awk '/
      ↪volumeName/{print $2}')
      $ ls -lat /mnt/$ACUMOS_NAMESPACE/$pvdir
      total 176
      drwxr-xr-x  2 root   root    4096 Nov 27 00:28 portal-fe
      drwxr-xr-x  3 root   root    4096 Nov 27 00:28 on-boarding
      drwxr-xr-x  3 root   root    4096 Nov 27 00:28 microservice-generation
      drwxr-xr-x  2 root   root    4096 Nov 27 00:28 federation-gateway
      drwxr-xr-x  2 root   root    4096 Nov 27 00:28 ds-compositionengine
      drwxr-xr-x  2 root   root    4096 Nov 27 00:27 project-service
      drwxr-xr-x  2 root   root    4096 Nov 27 00:27 predictor-service
      drwxr-xr-x  2 root   root    4096 Nov 27 00:27 pipeline-service
      drwxr-xr-x  2 root   root    4096 Nov 27 00:27 model-service
      drwxr-xr-x  2 root   root    4096 Nov 27 00:04 notebook-service
      -rw-r--r--  1 ubuntu ubuntu 19085 Nov 26 22:11 nifi-registry-app.log
      ```

```
-rw-r--r--  1 ubuntu ubuntu 10870 Nov 26 22:11 nifi-registry-bootstrap.log
drwxrwxrwx 15 ubuntu ubuntu  4096 Nov 26 22:11 .
-rw-r--r--  1 root   root    4750 Nov 26 22:10 docker-proxy.log
-rw-r--r--  1 ubuntu ubuntu 19087 Nov 26 19:27 nifi-registry-app_2019-11-
↪26_19.0.log
drwxr-xr-x  2 root   root    4096 Nov 26 19:24 portal-be
drwxr-xr-x  2 root   root    4096 Nov 26 19:22 cmn-data-svc
-rw-r--r--  1 ubuntu ubuntu  5435 Nov 25 04:42 nifi-registry-bootstrap_
↪2019-11-25.log
-rw-r--r--  1 ubuntu ubuntu  5435 Nov 24 17:33 nifi-registry-bootstrap_
↪2019-11-24.log
-rw-r--r--  1 ubuntu ubuntu  5435 Nov 23 19:10 nifi-registry-bootstrap_
↪2019-11-23.log
-rw-r--r--  1 ubuntu ubuntu 27175 Nov 22 19:23 nifi-registry-bootstrap_
↪2019-11-22.log
-rw-r--r--  1 ubuntu ubuntu     0 Nov 22 00:40 nifi-registry-event.log
-rw-r--r--  1 root   root       0 Nov 22 00:40 access.log
-rw-r--r--  1 root   root       0 Nov 22 00:40 docker-proxy-access.log
-rw-r--r--  1 root   root       0 Nov 22 00:40 docker-proxy-error.log
drwxr-xr-x  5 root   root    4096 Nov 22 00:37 deployment
drwxr-xr-x 32 ubuntu ubuntu  4096 Nov 22 00:02 ..
```

* You can also access the logs though one of the containers that has mounted the logs PVC, e.g. as in the following example which uses an aio_k8s_deployer container to access the cluster information remotely. In this case you see that specifying the namespace is not required, since that is automatically scoped per the kube-config file provided when the container was created.

```
$ docker exec -it acumos-deploy-opnfv04 bash
root@a29adaebcc84:/# pod=$(kubectl get pods | awk '/portal-be/{print $1}')
root@a29adaebcc84:/# kubectl exec -it $pod -- ls -lat /maven/logs
total 164
-rw-r--r--    1 1000     1000         19087 Nov 29 02:56 nifi-registry-
↪app.log
-rw-r--r--    1 1000     1000          5435 Nov 29 02:56 nifi-registry-
↪bootstrap.log
drwxrwxrwx   15 1000     1000          4096 Nov 29 02:56 .
-rw-r--r--    1 root     root          2850 Nov 29 02:56 docker-proxy.log
drwxr-xr-x    2 root     root          4096 Nov 29 02:38 portal-fe
drwxr-xr-x    3 root     root          4096 Nov 29 02:38 on-boarding
drwxr-xr-x    3 root     root          4096 Nov 29 02:38 microservice-
↪generation
drwxr-xr-x    2 root     root          4096 Nov 29 02:38 federation-
↪gateway
drwxr-xr-x    2 root     root          4096 Nov 29 02:38 ds-
↪compositionengine
drwxr-xr-x    2 root     root          4096 Nov 29 02:37 project-service
drwxr-xr-x    2 root     root          4096 Nov 29 02:37 predictor-service
drwxr-xr-x    2 root     root          4096 Nov 29 02:37 notebook-service
drwxr-xr-x    2 root     root          4096 Nov 29 02:37 model-service
drwxr-xr-x    2 root     root          4096 Nov 29 02:13 cmn-data-svc
drwxr-xr-x    2 root     root          4096 Nov 29 02:13 portal-be
drwxr-xr-x    2 root     root          4096 Nov 29 00:01 pipeline-service
-rw-r--r--    1 1000     1000         19085 Nov 28 04:03 nifi-registry-
↪app_2019-11-28_04.0.log
-rw-r--r--    1 1000     1000          5435 Nov 28 04:03 nifi-registry-
↪bootstrap_2019-11-28.log
-rw-r--r--    1 1000     1000         19084 Nov 27 21:45 nifi-registry-
↪app_2019-11-27_21.0.log
```

```
-rw-r--r--    1 1000     1000         21740 Nov 27 21:45 nifi-registry-
↪bootstrap_2019-11-27.log
-rw-r--r--    1 1000     1000             0 Nov 27 01:27 nifi-registry-
↪event.log
-rw-r--r--    1 root     root             0 Nov 27 01:00 access.log
-rw-r--r--    1 root     root             0 Nov 27 01:00 docker-proxy-
↪access.log
-rw-r--r--    1 root     root             0 Nov 27 01:00 docker-proxy-
↪error.log
drwxr-xr-x    5 root     root          4096 Nov 27 00:39 deployment
drwxrwxrwx    1 root     root          4096 Nov  7 11:16 ..
```

– If you did customize the SERVICE_LABEL values, the logs will be associated with PVCs named per the SERVICE_LABEL values you chose. Each PVC will host the logs for the components that you associated with the same SERVICE_LABEL value.

– If instead of an AIO deployment, you are using a multi-node k8s cluster for which you don't have direct access to the logs PV, the only way to access the logs without using a pod running under the same cluster and mounting the same PVC folder, is to execute a command that runs in one of the pods that is mounting the PVC. This is pretty easy, e.g.:

```
root@7b13a31255f2:/# source /deploy/system-integration/AIO/acumos_env.sh
root@7b13a31255f2:/# pod=$(kubectl get pods -n $ACUMOS_NAMESPACE | awk '/
↪portal-be/{print $1}')
root@7b13a31255f2:/# kubectl exec -it -n $ACUMOS_NAMESPACE $pod -- ls -lat /
↪maven/logs
total 20
drwxr-sr-x    2 root     1000          4096 Nov 28 03:26 portal-fe
drwxr-sr-x    2 root     1000          4096 Nov 28 01:16 portal-be
drwxr-sr-x    3 root     1000          4096 Nov 27 16:36 on-boarding
drwxrwsr-x    5 root     1000          4096 Nov 27 02:51 .
drwxrwxrwx    1 root     root          4096 Nov 26 12:12 ..
```

  * The example above accessing the logs from within the aio_k8s_deployer container, but will also work in other envs, just use the correct path for your acumos_env.sh script.
  * In the example above you also see that not all of the log folders in the previous (AIO) example are shown; this is because in this case the SERVICE_LABEL values in acumos_env.sh have been configured to distribute the components across the nodes of the k8s cluster in a cluster that only supports hostPath PVs, and thus a separate PVC has been used for each set of components scheduled on a node.

4.1.5 Security Considerations

As noted in **'Introduction'_**, the AIO deployment approach includes various development/test environment-enabling aspects, that for a more "production" or publicly-exposed deployment, should be reconsidered and as needed, locked down.

For k8s-based platforms, some of these aspects are related to work-in-progress on more fully supporting platform exposure through ingress controllers. An ingress controller is a convenient place to apply subnet-based restrictions on service access. See the NGINX ingress annotations guide for whitelist-source-range for more information. Following are the services exposed in the Acumos platform that can be access-controlled using a whitelist-source-range source annotation, by updating the related ingress template:

Service	Recommendation	Ingress template
CDS (Common Data Service)	Admin access only	AIO/ingress/templates/cds-ingress.yaml
NiFi Registry	Admin access only	AIO/mlwb/nifi/kubernetes/ingress-registry.yaml
NiFi User	User access (required)	AIO/mlwb/nifi/templates/ingress.yaml
JupyterHub (Hub and User)	User access (required)	See note below
Kubernetes client	User access (required)	AIO/ingress/templates/k8s-client-ingress.yaml
Onboarding	User access (required)	AIO/ingress/templates/onboarding-ingress.yaml
Portal	User access (required)	AIO/ingress/templates/portal-ingress.yaml
MLWB Dashboard	User access (required)	AIO/mlwb/kubernetes/mlwb-dashboard-webcomponent-ingress.yaml
MLWB Home	User access (required)	AIO/mlwb/kubernetes/mlwb-home-webcomponent-ingress.yaml
MLWB Notebook	User access (required)	AIO/mlwb/kubernetes/mlwb-notebook-webcomponent-ingress.yaml
MLWB Notebook Catalog	User access (required)	AIO/mlwb/kubernetes/mlwb-notebook-catalog-webcomponent-ingress.yaml
MLWB Pipeline	User access (required)	AIO/mlwb/kubernetes/mlwb-pipeline-webcomponent-ingress.yaml
MLWB Pipeline Catalog	User access (required)	AIO/mlwb/kubernetes/mlwb-pipeline-catalog-webcomponent-ingress.yaml
MLWB Project	User access (required)	AIO/mlwb/kubernetes/mlwb-project-webcomponent-ingress.yaml
MLWB Project Catalog	User access (required)	AIO/mlwb/kubernetes/mlwb-project-catalog-webcomponent-ingress.yaml

Notes on the table above:

- JupyterHub ingress rules are currently created by the deployment script in charts/jupyterhub/setup_jupyterhub.sh.

For other components, k8s nodeports are currently used for primary access or supplementatl access. Note that if possible, these services will be migrated to access via the ingress controller, early in the next release. However, in many cases these services may be provided as shared application services, and deployed outside the Acumos platform. The AIO toolset supports that as an option, which eliminates any concern about exposing these services as part of an Acumos platform:

Service	NodePort (default)	Rational for NodePort Use	Security Recommendation
CDS (Common Data Service)	30800	Alternative swagger UI (see note)	Apply firewall rule if needed
Docker Proxy	30883	Ingress rule is WIP (see note)	Leave open (required)
Federation (peer access)	30984	requires SSL termination	Leave open (required)
Federation (local access)	30985	requires SSL termination	Leave open (required)
Nexus (Maven)	30881	Admin access to Nexus UI (see note)	Apply firewall rule if needed, or use external service
Nexus (Docker)	30882	Admin access to Nexus Docker Registry (see note)	Apply firewall rule if needed, or use external service
JupyterHub	(dynamic ports)	Access to the Jupyter proxy (see note)	Apply firewall rule if needed
Security Verification (SV) Scanning Service	30982	Admin access to the SV service (see note)	Apply firewall rule if needed

Notes on the table above:

- The CDS NodePort addresses current issues (under investigation) with access to the CDS swagger UI via HTTPS thru the ingress controller
- The Docker Proxy NodePort is currently required because an ingress rule for it has not been implemented/tested. An update will be published as soon as this has been done.
- Configuraton of JupyterHub to avoid NodePort use is a WIP.
- Access to the Security Verification Scan API is provided so that Admins can invoke scans manually or through external systems, and also for the future support of external scan result notifications.

4.1.6 Debugging Hints

Accessing Logs

For k8s-based deployments, the simplest way to view/tail log files is to run the following command in the pod for the specific component, modifying the example as needed per the specific component, as shown in the table following:

```
source system-integration/AIO/acumos_env.sh
pod=$(kubectl get pods -n $ACUMOS_NAMESPACE -l app=cds | awk '/cds/{print $1}')
kubectl exec -n $ACUMOS_NAMESPACE $pod -- cat maven/logs/cmn-data-svc/cmn-data-svc.log
```

Enable Debug Log Level

For components deployed through kubernetes templates (not via Helm), the AIO toolset creates parameter-substituted component deployment templates in a “deploy” subfolder under:

- AIO (for most core components)
- AIO/docker-proxy
- AIO/mlwb

- AIO/mariadb
- AIO/nexus
- AIO/beats

For the Java Springboot-based Acumos core components, you can enable debug logs using the following example bash commands, which will enable debug for the CDS, Portal-FE, and Portal-BE components.

For the Security Verification Client Library (SVCL) which is built into the Portal-BE image, enable debug logs via:

4.1.7 Known Issues

Following are some things to watch out for in the process of deploying the Acumos platform with the AIO toolset. Solutions to improve the reliability of platform deployment are being implemented with each release, but this is a work-in-progress, and given that the Acumos platform deployment depends upon a number of upstream projects/components, issues such as the below may be encountered. This list will be updated as needed after the final Clio release user guide is published, and will be available on the Acumos wiki at Hints and Known Issues

Out of Space Issues

"Out of space" issues can happen periodically and need to be managed manually, using the techniques below. A goal of the Acumos project is to develop monitoring and cleanup tools to help address some of these issues, but for now the following advice should work.

The most commonly affected components are:

- the docker engine, on the host or in a container, which has run out of space in /var/lib/docker; this
-

The OneClick toolset by default creates fairly small PVs for k8s deployments, and you may find that some services

Nexus service is out of space in the nexus-data PV

The AIO default PV size for the nexus-data PVC and the backing PV is 10Gi. This can be exhausted after a few images have been generated, and needs to be periodically cleaned. The effect is that artifacts or images can't be saved by the Nexus service. Onboarding service logs such as the below can indicate such issues are occuring:

```
2019-11-29T10:46:46.724Z      http-nio-8090-exec-2    ERROR   org.acumos.onboarding.
↪services.impl.CommonOnboarding              Severity=DEBUG
Fail to upload artificat for OnboardingLog.txt - Failed to transfer file: http://
↪nexus-service.acumos-prod.svc.cluster.local:8081
/repository/acumos_model_maven/org/acumos/64cd4781-d5b0-4ced-89e2-7fb3616d56ba/
↪OnboardingLog/1.0.2/OnboardingLog-1.0.2.txt.
Return code is: 500   org.apache.maven.wagon.TransferFailedException: Failed to␣
↪transfer file: http://nexus-service.acumos-prod.svc.cluster.local:8081
/repository/acumos_model_maven/org/acumos/64cd4781-d5b0-4ced-89e2-7fb3616d56ba/
↪OnboardingLog/1.0.2/OnboardingLog-1.0.2.txt.
Return code is: 500
```

To verify that there actually is an out-of-space issue:

```
kubectl exec -it $(kubectl get pods | awk '/nexus/{print $1}') -- df -k
```

Note that expanding a PV while preserving the data on it may be a complicated process and require certain PV features such as Dynamic Provisioning which may not be available. As a fallback, the process below will expand the PV but does require recreation of the Acumos database and all artifacts. For development/test platforms this is usually not a major concern, but for production platforms the process below should only be used if you are OK with losing all data in the platform.

```
kubectl exec -it $(kubectl get pods | awk '/nexus/{print $1}') -- df -k
sed -i 's/ACUMOS_NEXUS_DATA_PV_SIZE=.*/ACUMOS_NEXUS_DATA_PV_SIZE=100Gi/' \
  system-integration/AIO/nexus/nexus_env.sh
bash system-integration/AIO/nexus/setup_nexus.sh all
bash system-integration/AIO/nexus/setup_nexus_repos.sh
bash system-integration/AIO/setup_acumosdb.sh
```

Docker-dind service is out of space in the docker-volume PV

The AIO default PV size for the docker-volume PVC and the backing PV is 10Gi. This can be exhausted after a few images have been generated, and needs to be periodically cleaned. The effect is that image generation fails, with logged errors in the msg log, that you can see by:

```
# kubectl log $(kubectl get pods | awk '/msg/{print $1}') | \
  grep -e "ERROR" -e "E: You don't have enough free space"
...
2019-11-28T11:31:35.429Z    http-nio-8336-exec-2    ERROR   org.acumos.microservice.
↪services.impl.GenerateMicroserviceController
User=f3c25ebe-572e-4e69-a9a4-84728b06d836, RequestID=55d69f4e-4489-48e6-b089-
↪08d5a0640240, ServiceName=/onboarding-app/v2/models,
ResponseDescription=com.github.dockerjava.api.exception.DockerClientException: Could␣
↪not build image:
The command '/bin/sh -c apt-get clean && apt-get update && apt-get -y install libgtk2.
↪0-dev' returned a non-zero code: 100
...
2019-11-28T14:15:04.570Z    dockerjava-netty-28-1   INFO    org.acumos.microservice.
↪component.docker.cmd.CreateImageCommand
Severity=DEBUG      E: You don't have enough free space in /var/cache/apt/archives/.
...
# kubectl log $(kubectl get pods | awk '/docker-dind/{print $1}') -c docker-daemon |␣
↪grep -e "no space"
log is DEPRECATED and will be removed in a future version. Use logs instead.
time="2019-11-28T08:06:50.520301700Z" level=error msg="Download failed: write /var/
↪lib/docker/tmp/GetImageBlob259950183: no space left on device"
time="2019-11-28T08:06:58.323230667Z" level=info msg="Attempting next endpoint for␣
↪pull after error:
write /var/lib/docker/tmp/GetImageBlob406994074: no space left on device"
time="2019-11-28T10:15:56.663192359Z" level=info msg="Attempting next endpoint for␣
↪pull after error: failed to register layer:
Error processing tar file(exit status 1): write /usr/local/lib/python3.5/lib-dynload/_
↪pickle.cpython-35m-x86_64-linux-gnu.so: no space left on device"
```

To detect, fix, and verify the fix for out of space on the docker-volume, run these commands:

```
# kubectl exec -it $(kubectl get pods | awk '/docker-dind/{print $1}') -c docker-
↪daemon -- sh
/ # df -k
Filesystem           1K-blocks      Used Available Use% Mounted on
overlay              101584140  41489332  60078424  41% /
tmpfs                    65536         0     65536   0% /dev
tmpfs                   3556832         0   3556832   0% /sys/fs/cgroup
/dev/sda1             101584140  41489332  60078424  41% /dev/termination-log
/dev/sda1             101584140  41489332  60078424  41% /etc/resolv.conf
/dev/sda1             101584140  41489332  60078424  41% /etc/hostname
/dev/sda1             101584140  41489332  60078424  41% /etc/hosts
shm                       65536         0     65536   0% /dev/shm
/dev/sdd               10190136  10109740     64012  99% /var/lib/docker
tmpfs                   3556832        12   3556820   0% /run/secrets/kubernetes.io/
↪serviceaccount
none                    3556832         0   3556832   0% /tmp
overlay                10190136  10109740     64012  99% /var/lib/docker/overlay2/
↪5be16850f14618b331e8728ed5750078998fc278b5a77c4c772edb808c06dd53/merged
shm                       65536         0     65536   0% /var/lib/docker/containers/
↪f30e1f576078b8fc98b1afbcbcd9ecdfca7be8fb01ced32c2b97486bb43110a5/mounts/shm
/ # exit

# kubectl exec -it $(kubectl get pods | awk '/docker-dind/{print $1}') -c docker-
↪daemon -- docker system prune -a -f
Deleted Containers:
9a4124be5079ce7cd78ef7d4d934bfb2d9745cc983ba910f1938826c4f7611dd
fd952e94d799cce8dda9e7971d96dd03338773a7aec63b7308c15a1b46da8cd7
be6c614b693f3fd4ded60a538dae437f5266be1ba964fcb3a6f01b3f19a5c31e
...
deleted: sha256:7c82a79e7c32df5cd4d9f9ec8da86396c06e6dcfa99d5e991c2e98b8e804e8d0
deleted: sha256:8823818c474862932702f8a920abea43b2560ddceb910d145be9ba0eb149a643

Total reclaimed space: 9.407GB

# kubectl exec -it $(kubectl get pods | awk '/docker-dind/{print $1}') -c docker-
↪daemon -- df -k
Filesystem            1K-blocks      Used Available Use% Mounted on
overlay               101584140  41489424  60078332  41% /
tmpfs                     65536         0     65536   0% /dev
tmpfs                   3556832         0   3556832   0% /sys/fs/cgroup
/dev/sda1             101584140  41489424  60078332  41% /dev/termination-log
/dev/sda1             101584140  41489424  60078332  41% /etc/resolv.conf
/dev/sda1             101584140  41489424  60078332  41% /etc/hostname
/dev/sda1             101584140  41489424  60078332  41% /etc/hosts
shm                       65536         0     65536   0% /dev/shm
/dev/sdd               10190136    201992   9971760   2% /var/lib/docker
tmpfs                   3556832        12   3556820   0% /run/secrets/kubernetes.io/
↪serviceaccount
none                    3556832         0   3556832   0% /tmp
overlay                10190136    201992   9971760   2% /var/lib/docker/overlay2/
↪5be16850f14618b331e8728ed5750078998fc278b5a77c4c772edb808c06dd53/merged
shm                       65536         0     65536   0% /var/lib/docker/containers/
↪f30e1f576078b8fc98b1afbcbcd9ecdfca7be8fb01ced32c2b97486bb43110a5/mounts/shm
```

MariaDB Mirror Reliability

Periodically, the list of active mirrors for the MariaDB project may change. This can break Acumos installation at the point of setting up the MariaDB client, which is the only MariaDB component that needs to be installed on the platform host. The client enables the AIO tools to setup/upgrade the Acumos database as needed. If you encounter an AIO install failure of the type below, you will need to update the MariaDB mirror found in charts/mariadb/setup_mariadb_env.sh:

```
...
Reading package lists...
W: The repository 'http://ftp.utexas.edu/mariadb/repo/10.2/ubuntu xenial Release'␣
↪does not have a Release file.
E: Failed to fetch http://ftp.utexas.edu/mariadb/repo/10.2/ubuntu/dists/xenial/main/
↪binary-i386/Packages  403  Forbidden
E: Some index files failed to download. They have been ignored, or old ones used␣
↪instead.
++ fail
++ set +x

fail:42 (Wed Jul 17 17:06:41 UTC 2019) unknown failure at setup_mariadb_client 70
```

If you see such a failure, select and configure a new MariaDB mirror from the MariaDB mirror list, as needed using a site such as the status of MariaDB mirrors to know which mirrors are active. Update the following line in charts/mariadb/setup_mariadb_env.sh and restart the deployment, selecting a new mirror host to replace 'sfo1.mirrors.digitalocean.com'.

```
export MARIADB_MIRROR="${MARIADB_MIRROR:-sfo1.mirrors.digitalocean.com}"
```

You may also need to manually remove old/inactive MariaDB mirrors from /etc/apt/sources.list, if you get later errors from using 'apt-get update', via these commands:

```
sudo sed -i -- '/mariadb/d' /etc/apt/sources.list
sudo apt-get update
```

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).

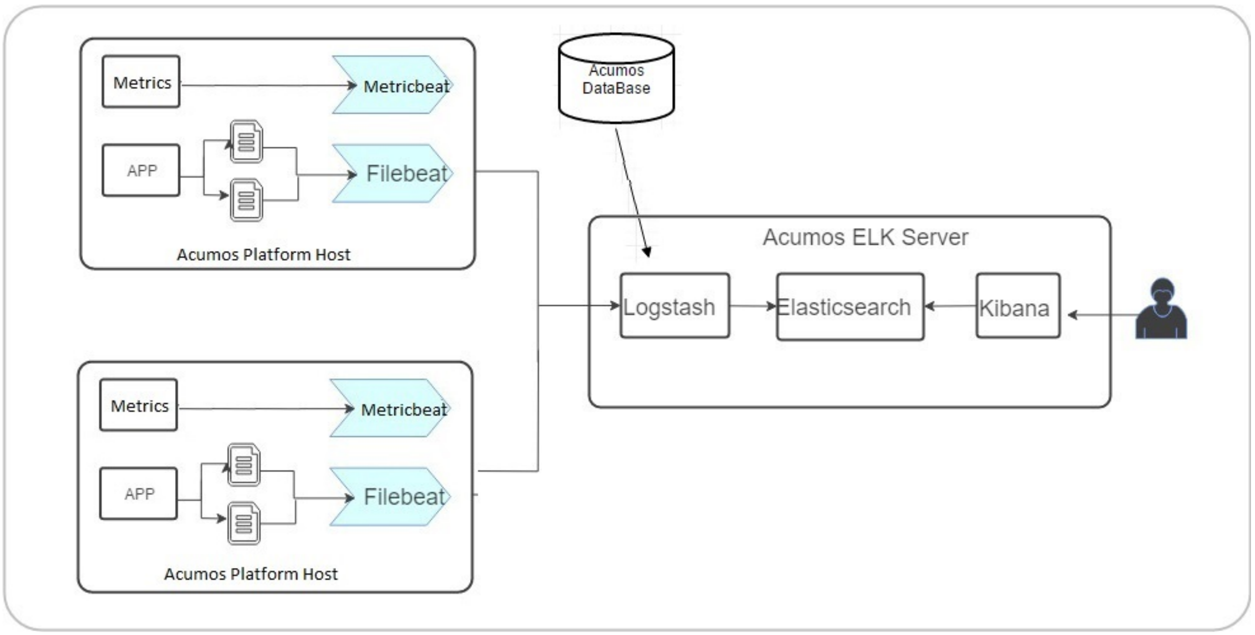
Target Users

Acumos Platform super admins

Assumptions

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

Elastic Stack Architecture



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.

Elastic Stack Component Versions

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

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.

Prerequisites

Docker and Docker Compose installed

Steps

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.

```
$ 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
```

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
```

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
```

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>.

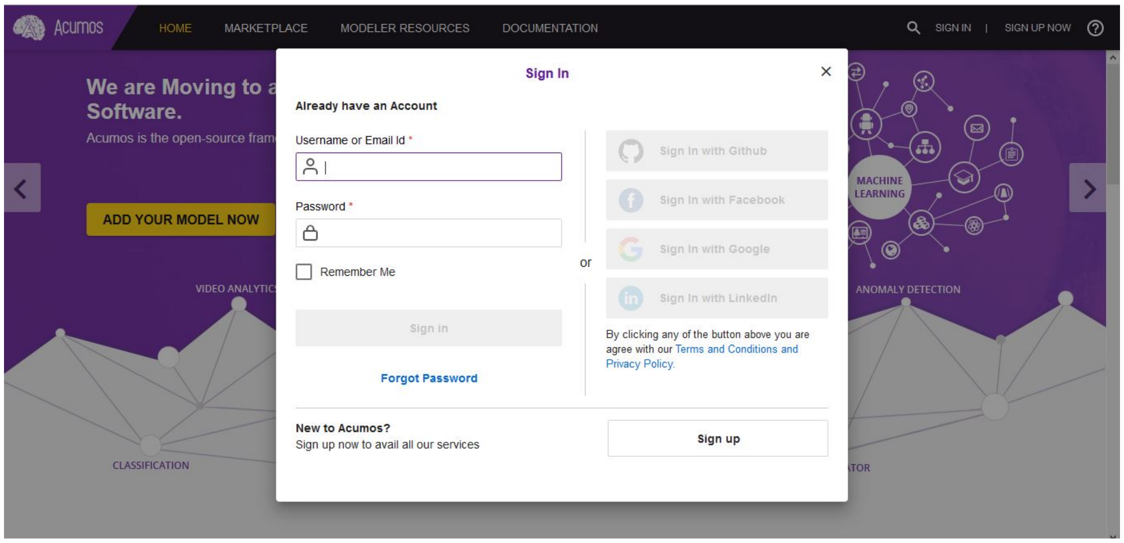
```
filebeat.prospectors:
  - input_type: log
    paths:
      - /filebeat-logs/portal-be/*.log
```

Elastic Stack UI Tour

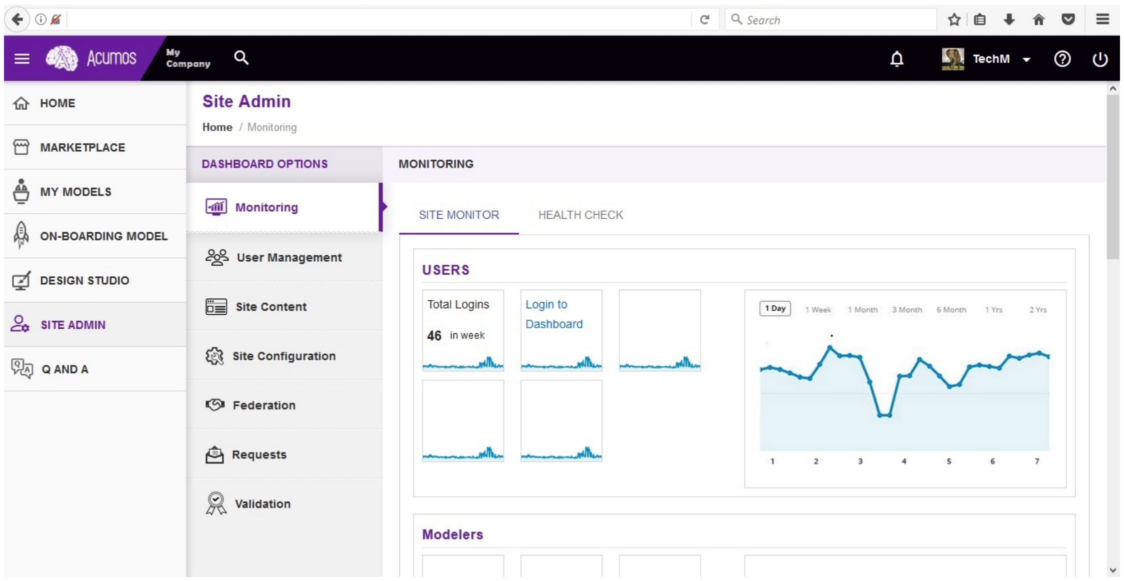
According to the Kibana website, Kibana is an open source analytics and visualization platform designed to work with Elasticsearch. You use Kibana to search, view, and interact with data stored in Elasticsearch indices. You can easily perform advanced data analysis and visualize your data in a variety 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.

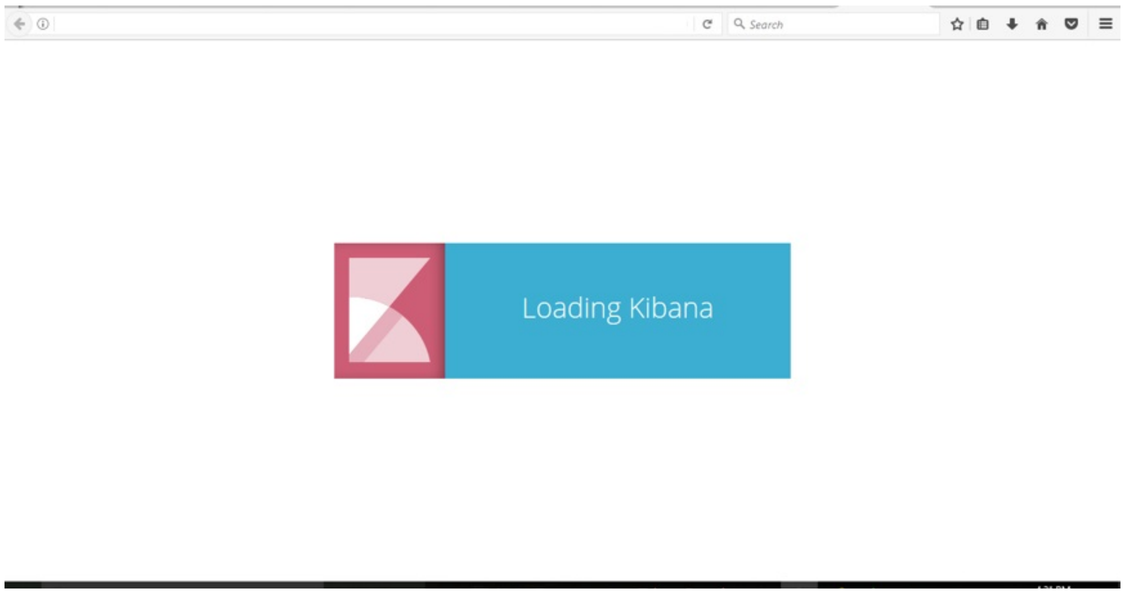
Site admins have access to Elastic Stack's Kibana Dashboard. Login to the dashboard:



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



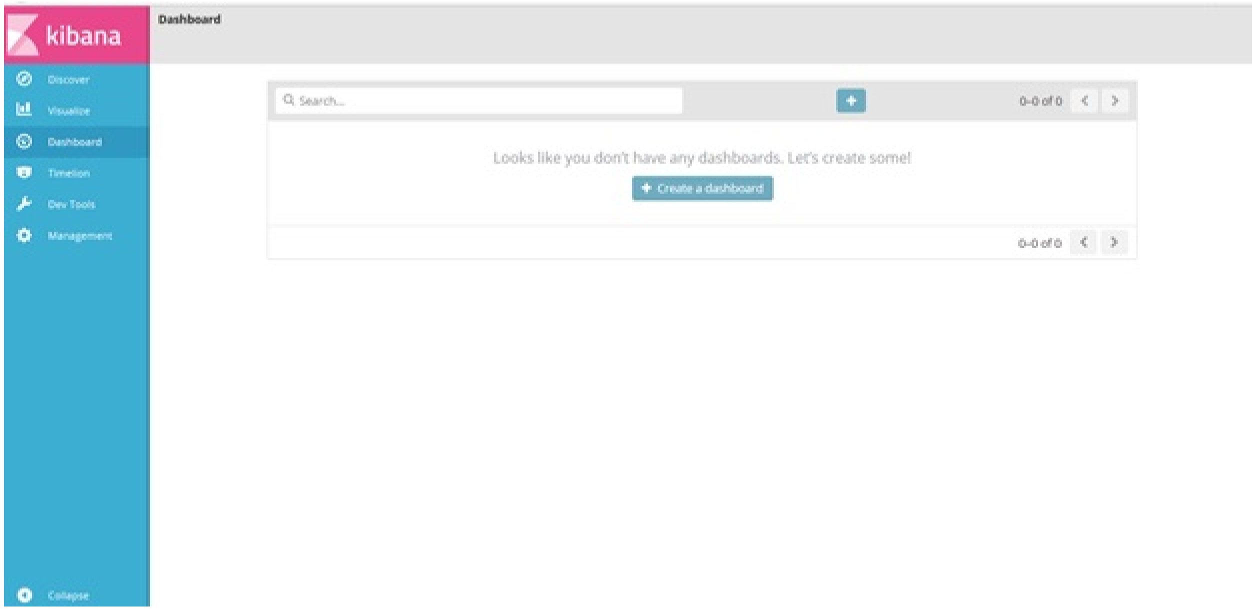
Redirects to Loading Kibana visualization platform



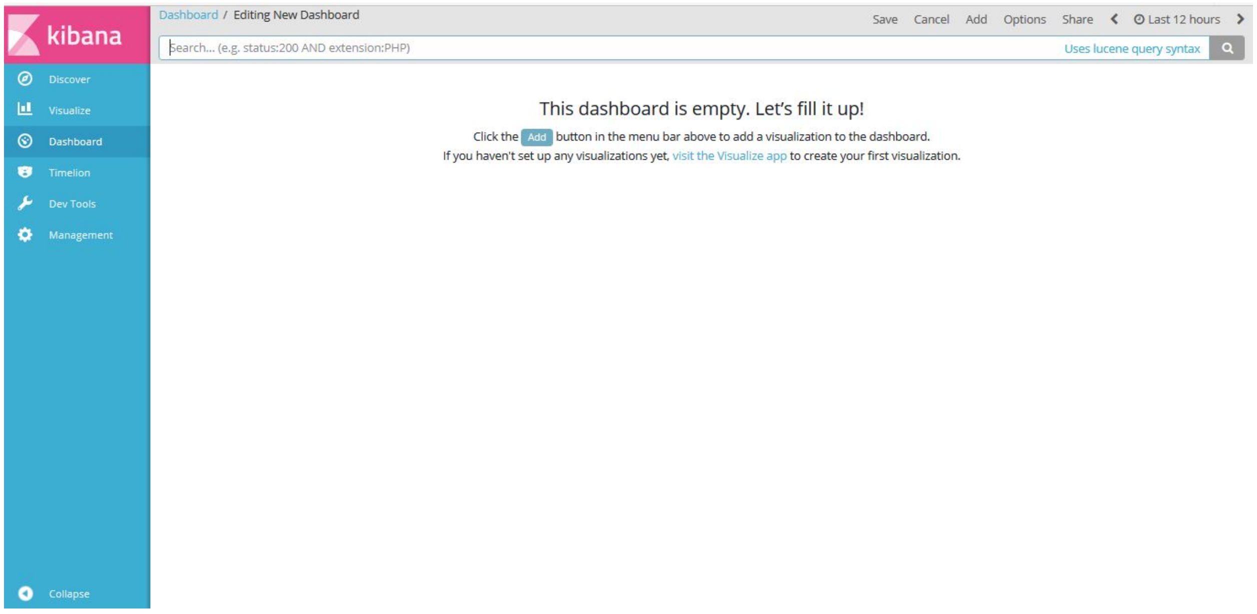
Acumos Kibana Dashboard Creation

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.



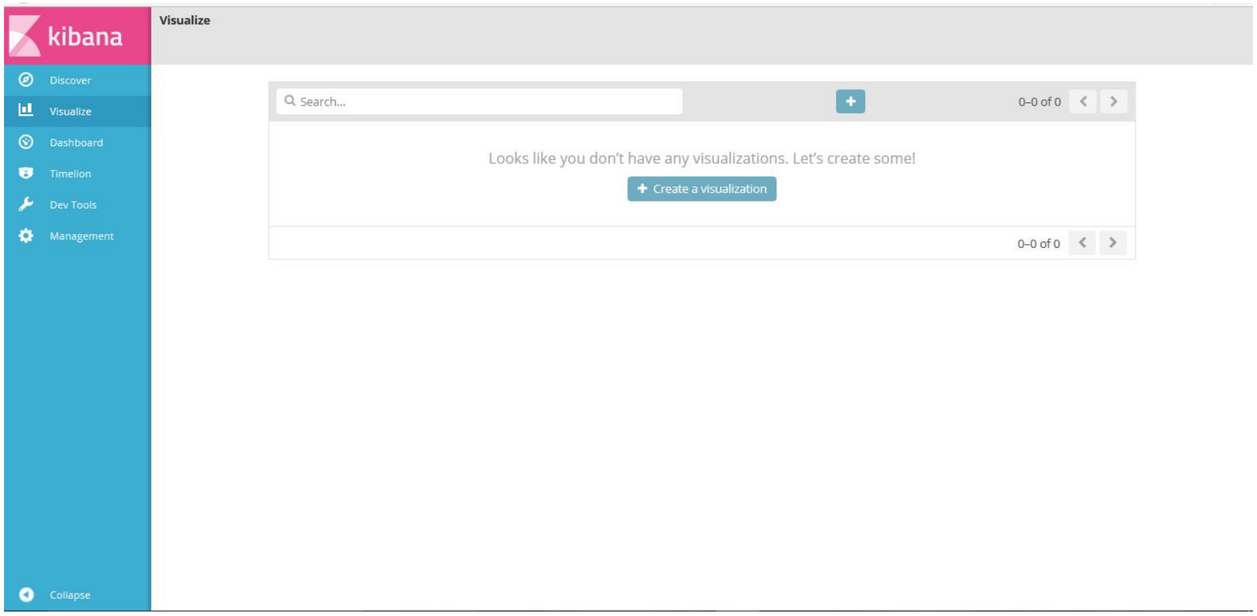
click on Visit the Visualize app



This dashboard is empty. Let's fill it up!

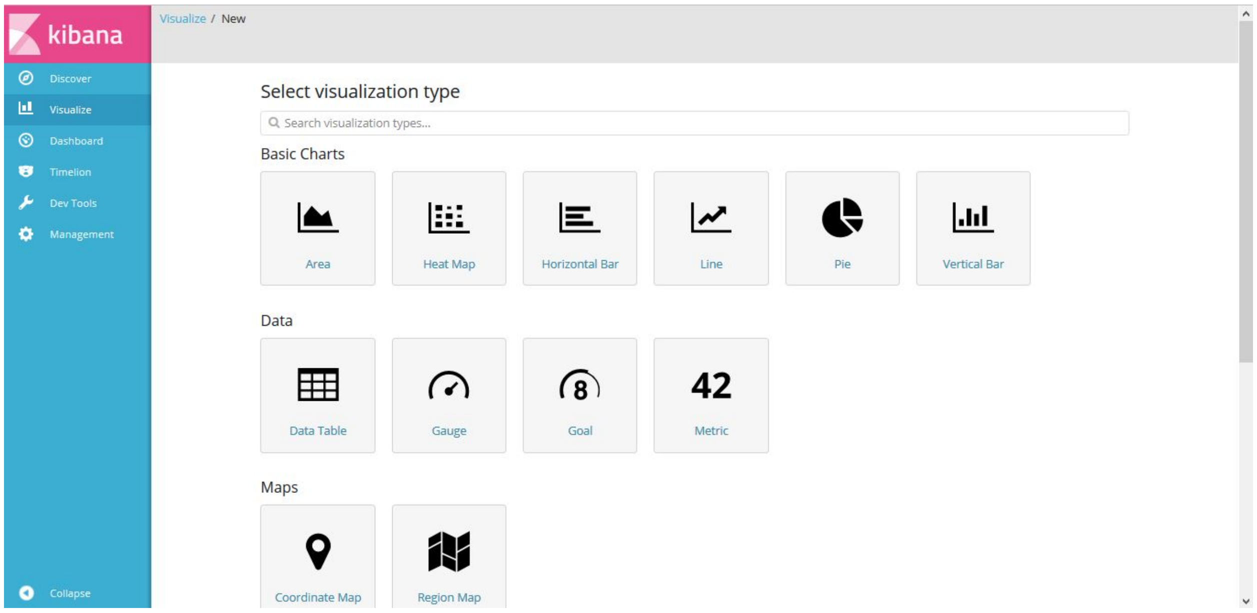
Click the Add button in the menu bar above to add a visualization to the dashboard.
If you haven't set up any visualizations yet, visit the Visualize app to create your first visualization.

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

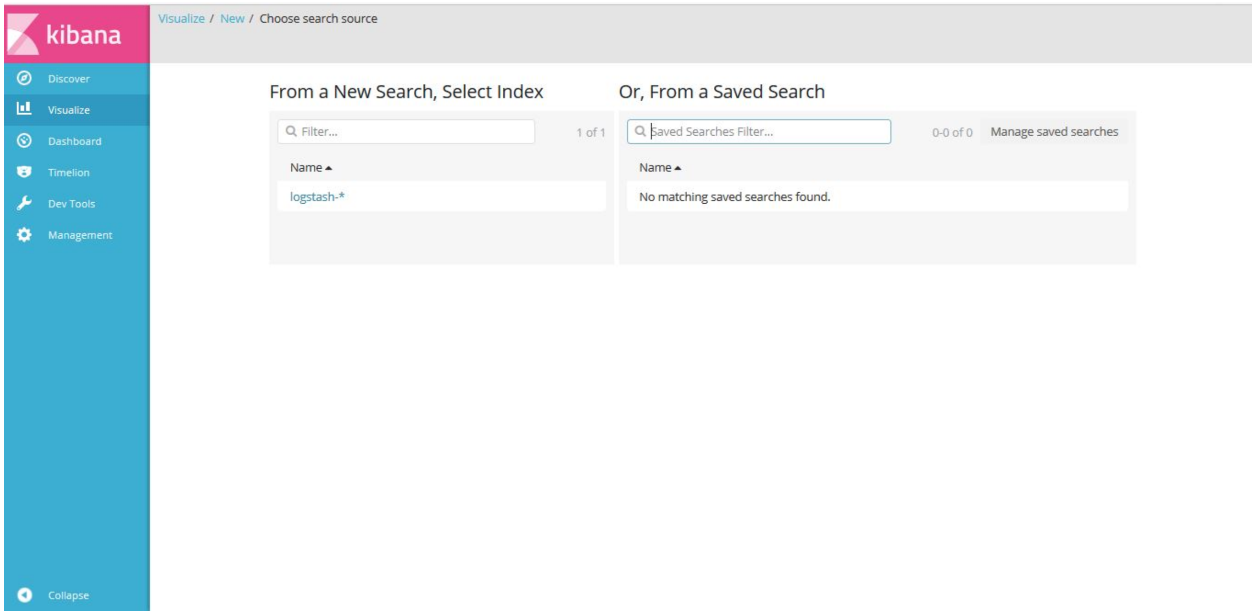


Looks like you don't have any visualizations. Let's create some!

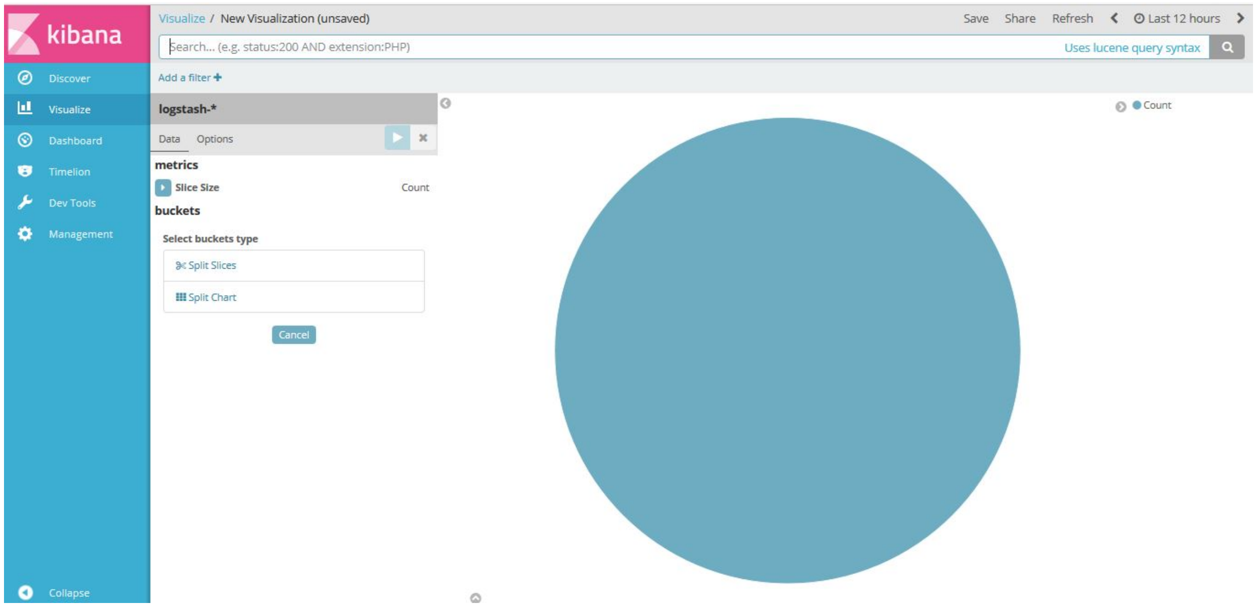
Select visualization type. For example click on "Pie".



Choose search source as `logstash-*`

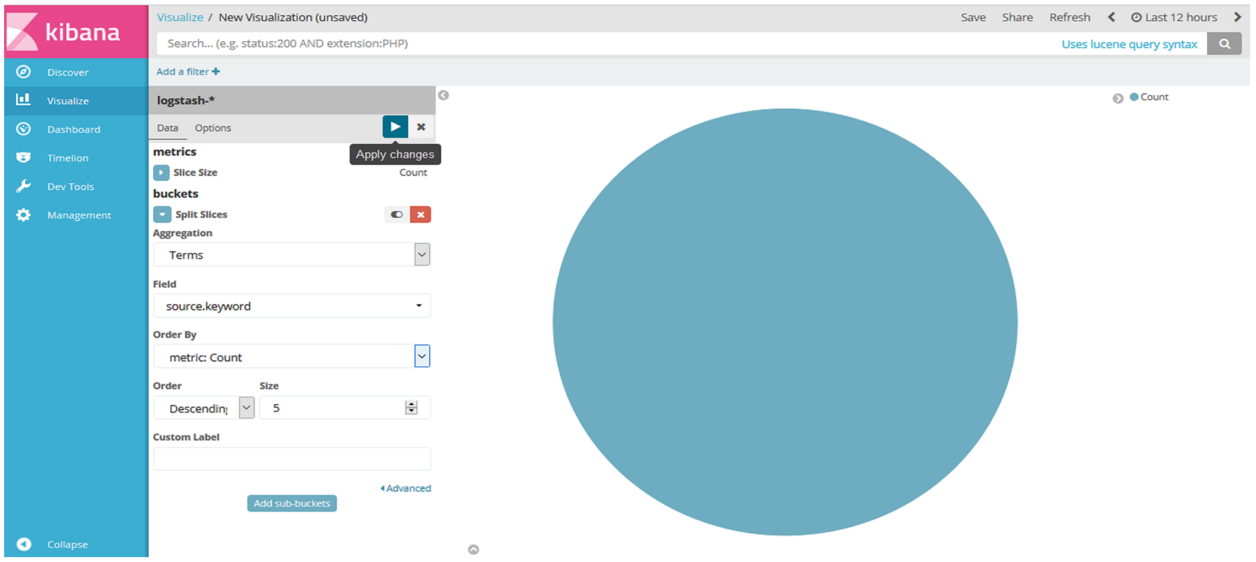


Click on Split Slices

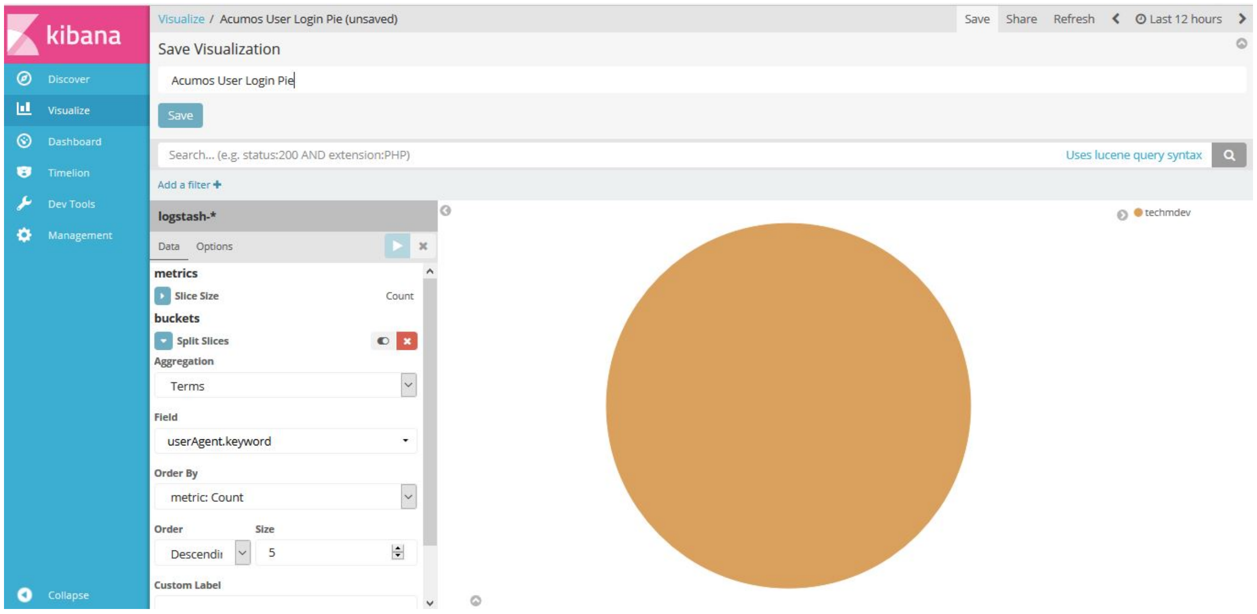


Select Aggregation as "Terms" and Field as "userAgent.keyword", Click on "Apply changes"

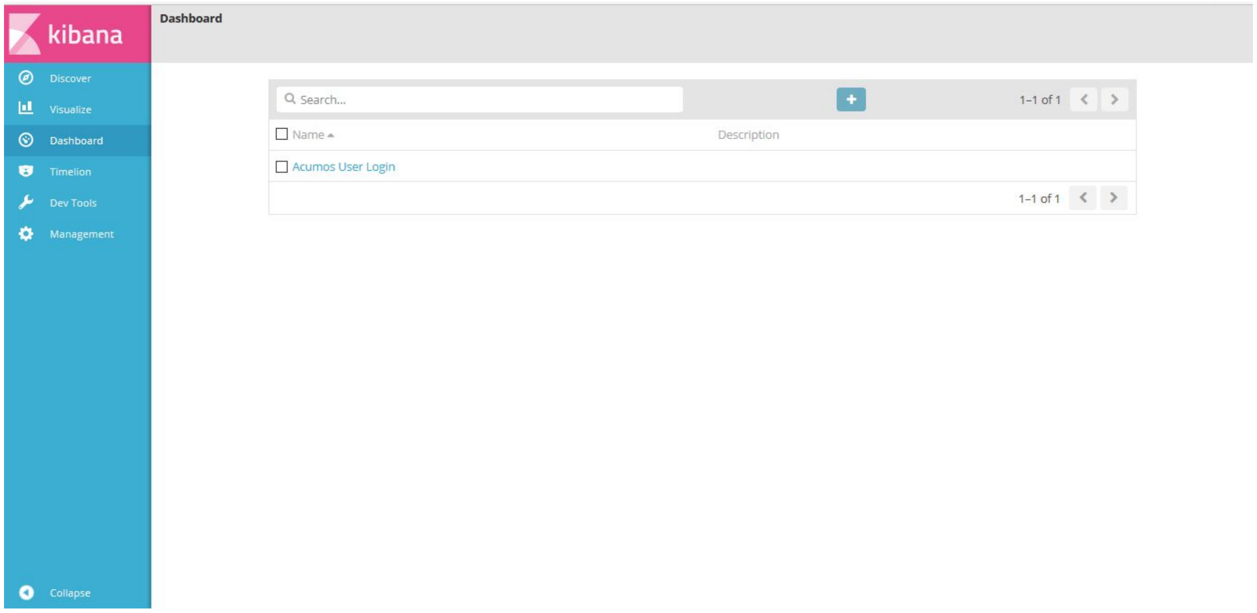
Note: Elasticsearch aggregations are to extract and process your data.



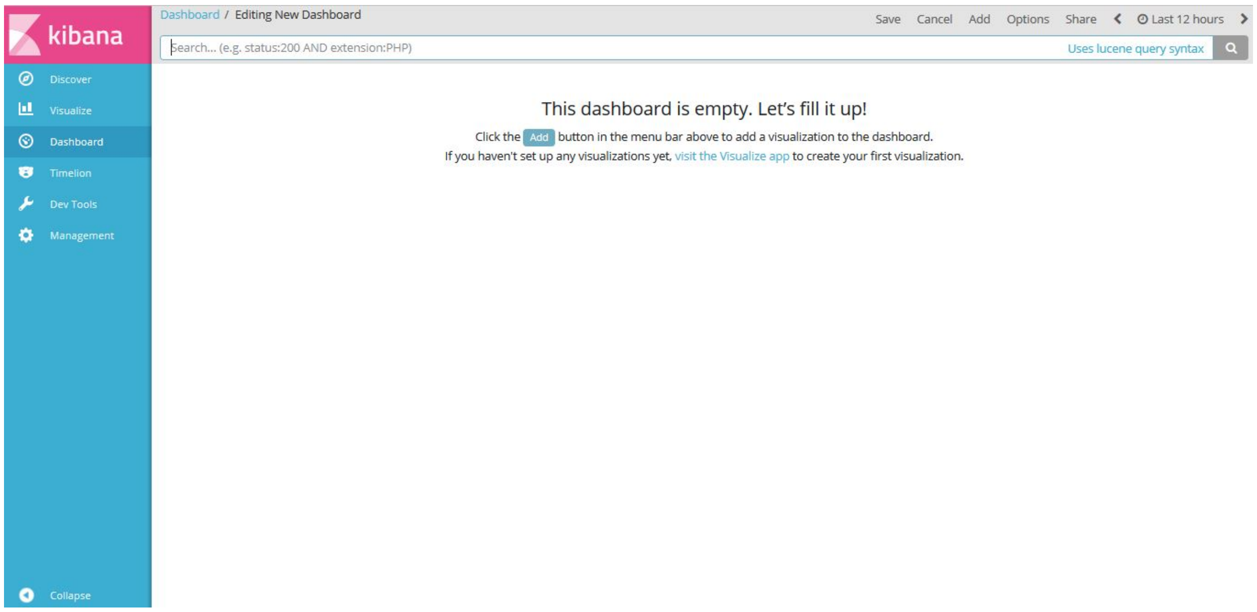
To save this chart click on "Save", Enter a name appropriate name. For example "Acumos User Login".



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.



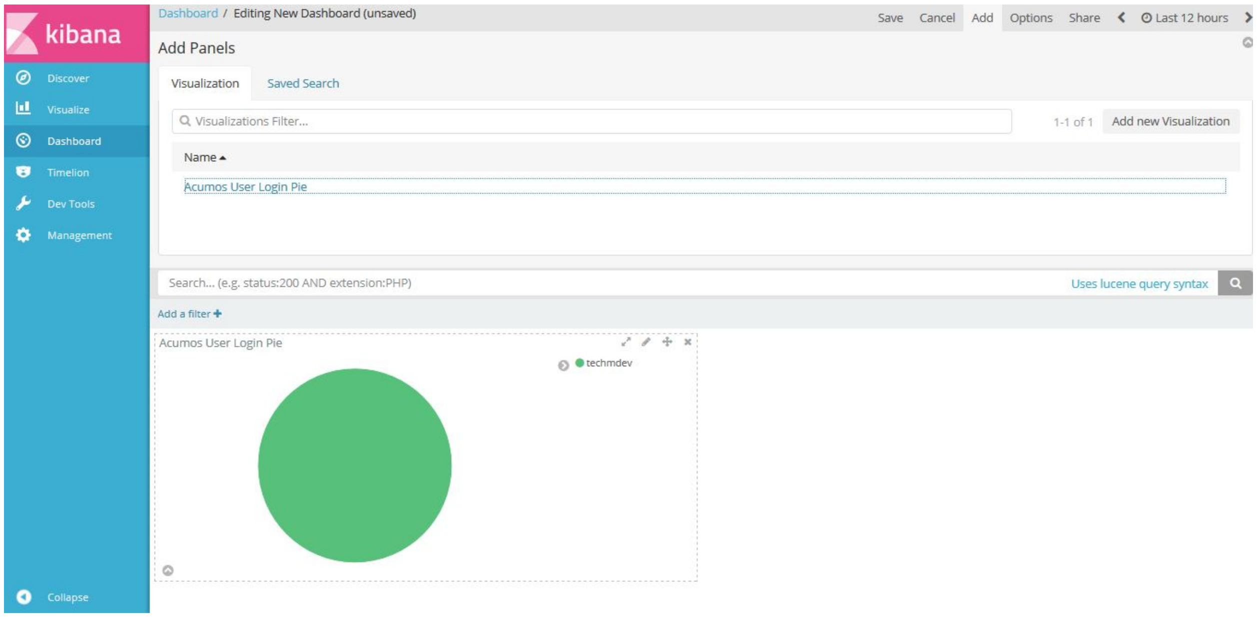
Click on “Add” button, to add the visualization.



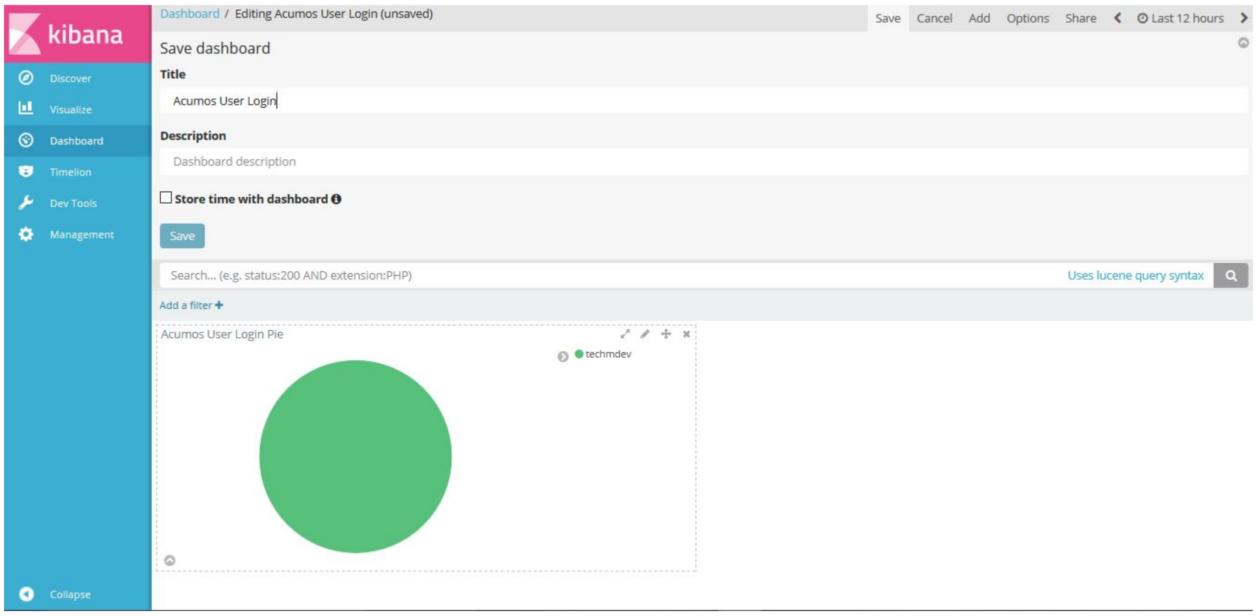
This dashboard is empty. Let's fill it up!

Click the Add button in the menu bar above to add a visualization to the dashboard.
If you haven't set up any visualizations yet, visit the Visualize app to create your first visualization.

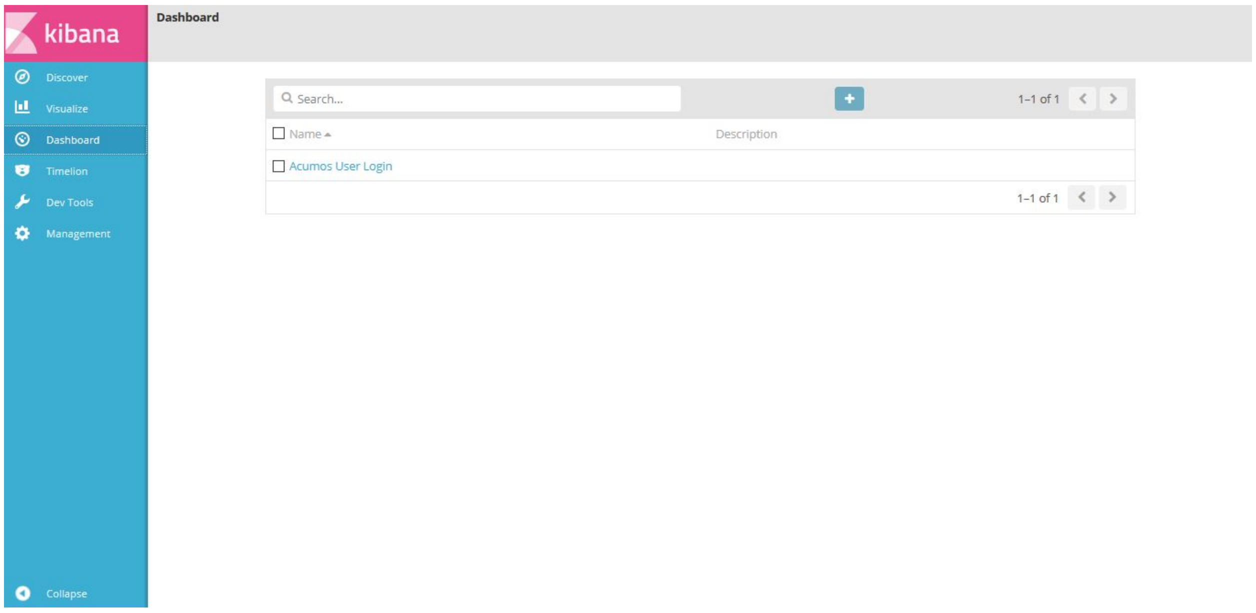
Select the visualization for example here we have visualization namely “Acumos User Login”.



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

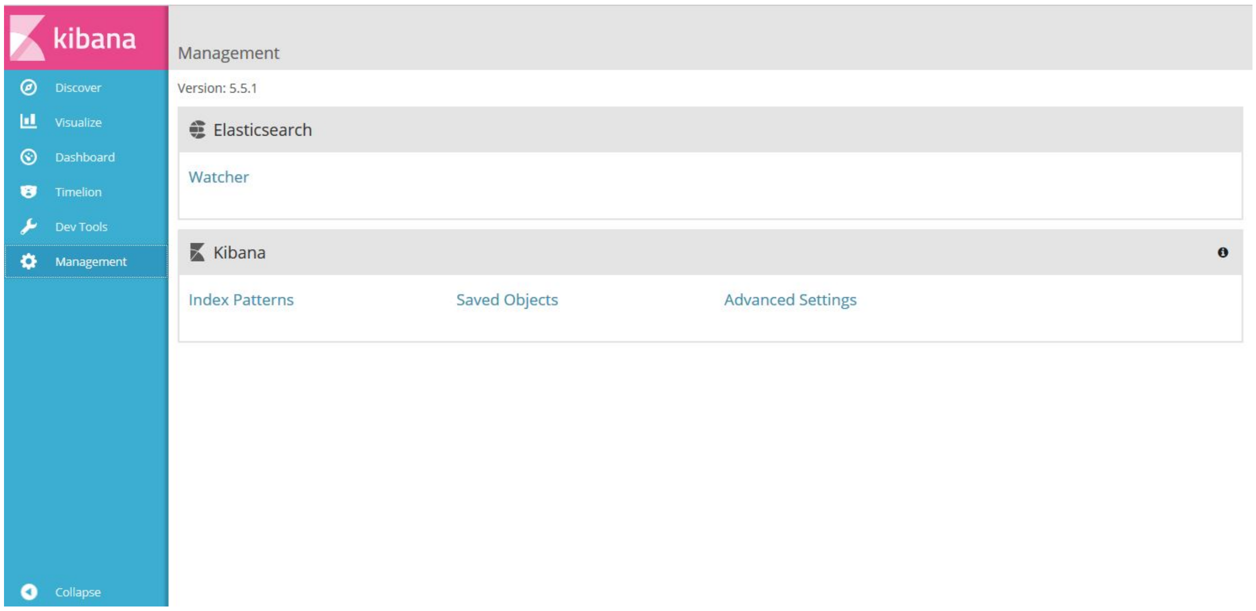


Click on "Dashboard", On the below screen created dashboard can be viewed namely "Acumos User Login".

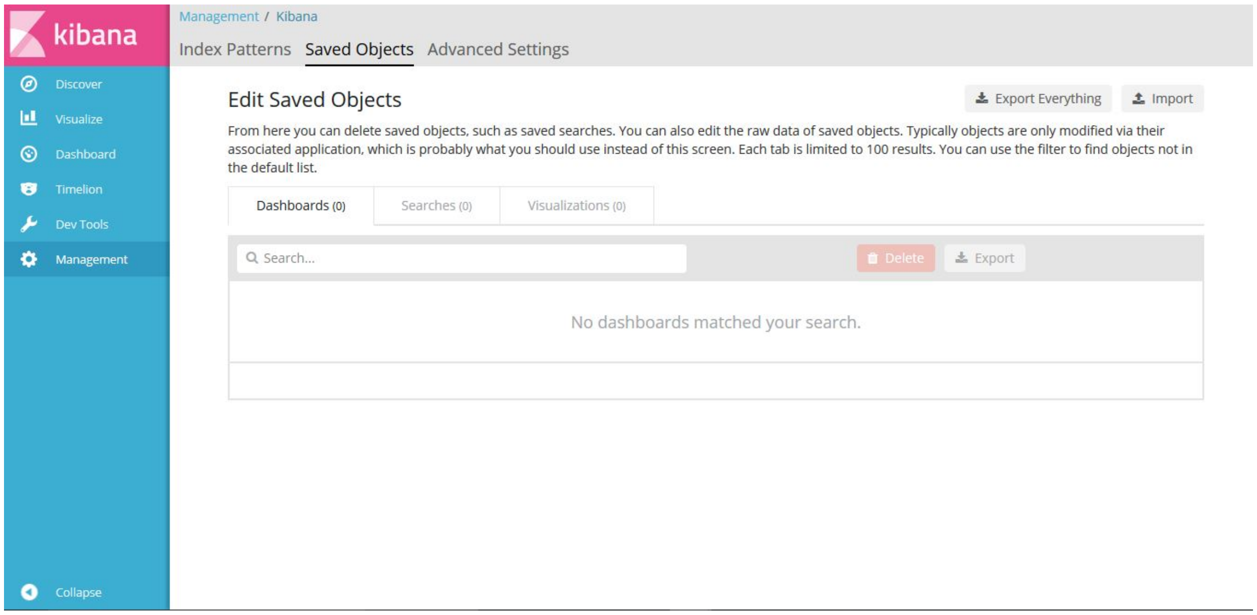


Acumos Kibana Dashboard Save

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.



From here you can delete saved objects, such as saved searches. You can also edit the raw data of saved objects. Typically objects are only modified via their associated application, which is probably what you should use instead of this screen. Each tab is limited to 100 results. You can use the filter to find objects not in the default list.

No dashboards matched your search.

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.

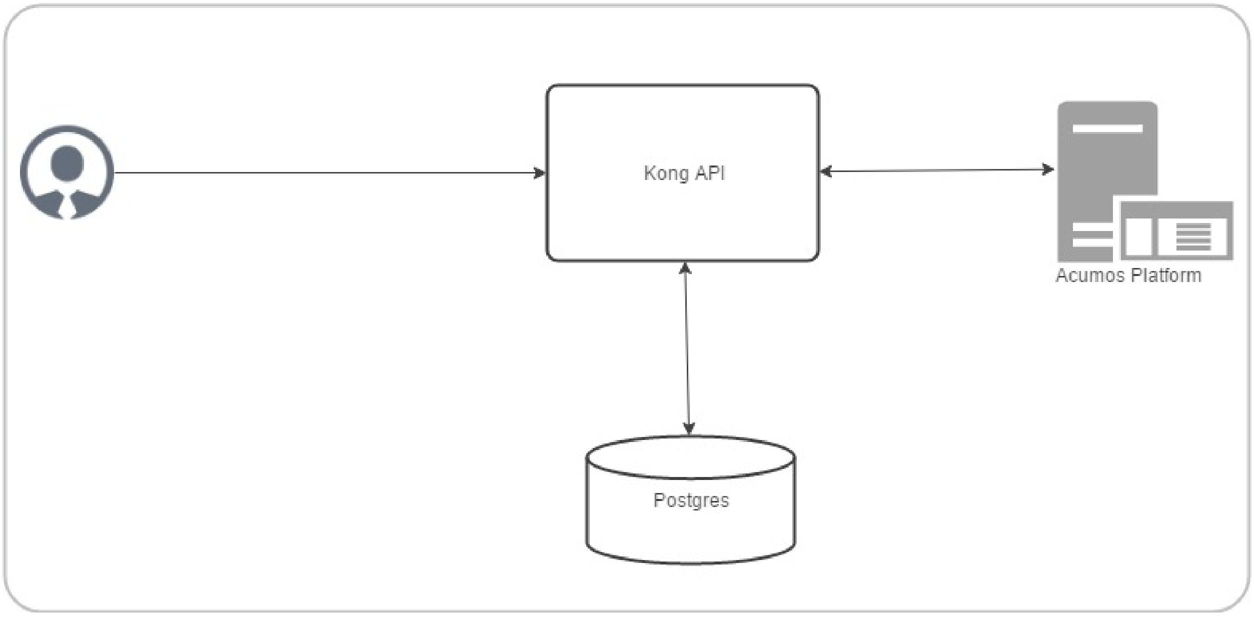
4.3 System Integration User Guide

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.

Kong API component versions

- postgres:9.4
- kong:0.11.0

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.

Prerequisites

Docker and Docker Compose installed

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
```

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
```

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
```

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,

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 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.

Future Releases

Future releases may include these new features:

- Scaling up, monitoring health tool.

Prerequisites

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

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.

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
```

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 pometheus 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 config map to send email alerts notification.

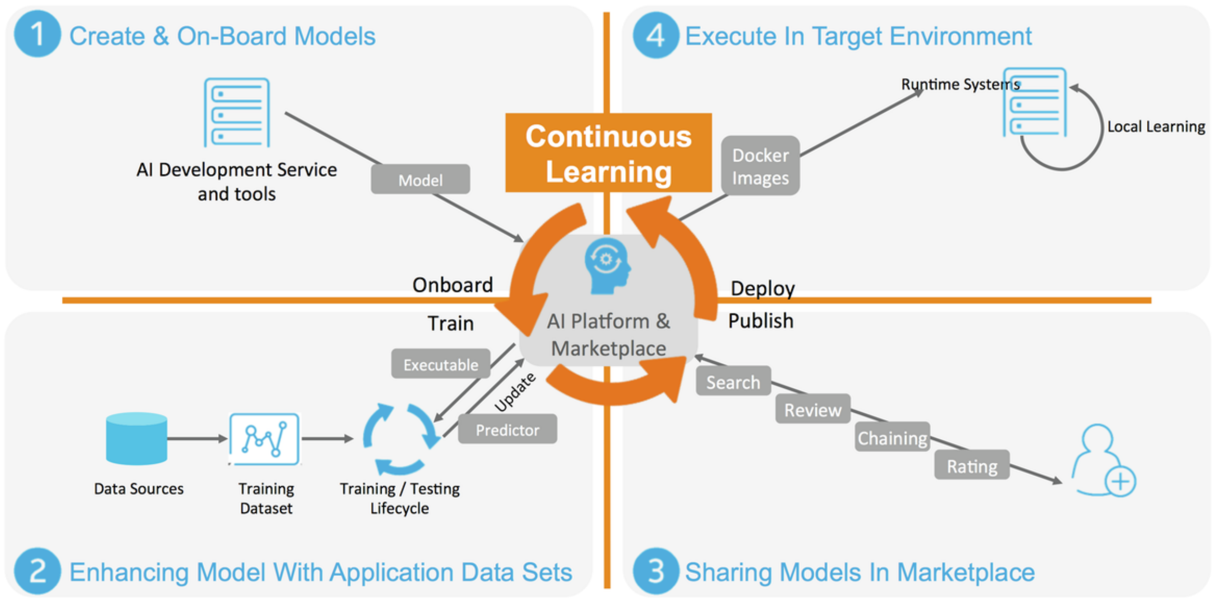
CHAPTER 5

Contributors to the Acumos Platform Code

5.1 Platform Architecture

5.1.1 Architecture Guide

Introduction


1 Create & On-Board Models
AI Development Service and tools
Model
4 Execute In Target Environment
Runtime Systems
Local Learning
Docker Images
Continuous Learning
Onboard
Train
Deploy
Publish
AI Platform & Marketplace
Executable
Update
Predictor
Search
Review
Chaining
Rating
Data Sources
Training Dataset
Training / Testing Lifecycle
2 Enhancing Model With Application Data Sets
3 Sharing Models In Marketplace

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.

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

Requirements

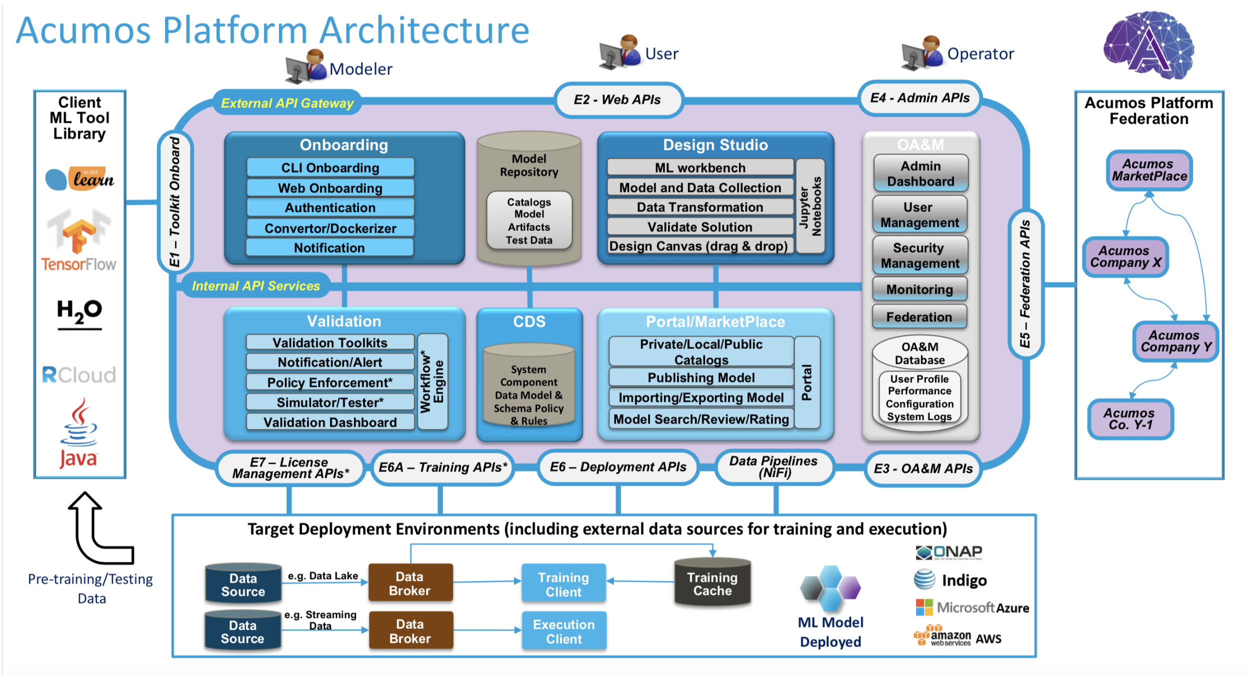
As described on the Acumos.org website, Acumos AI is a platform and open source framework that makes it easy to build, share, and deploy AI apps, and operate the Acumos portals that enable those capabilities. Acumos standardizes the infrastructure stack and components required to run an out-of-the-box 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

Architecture

Architecture Overview



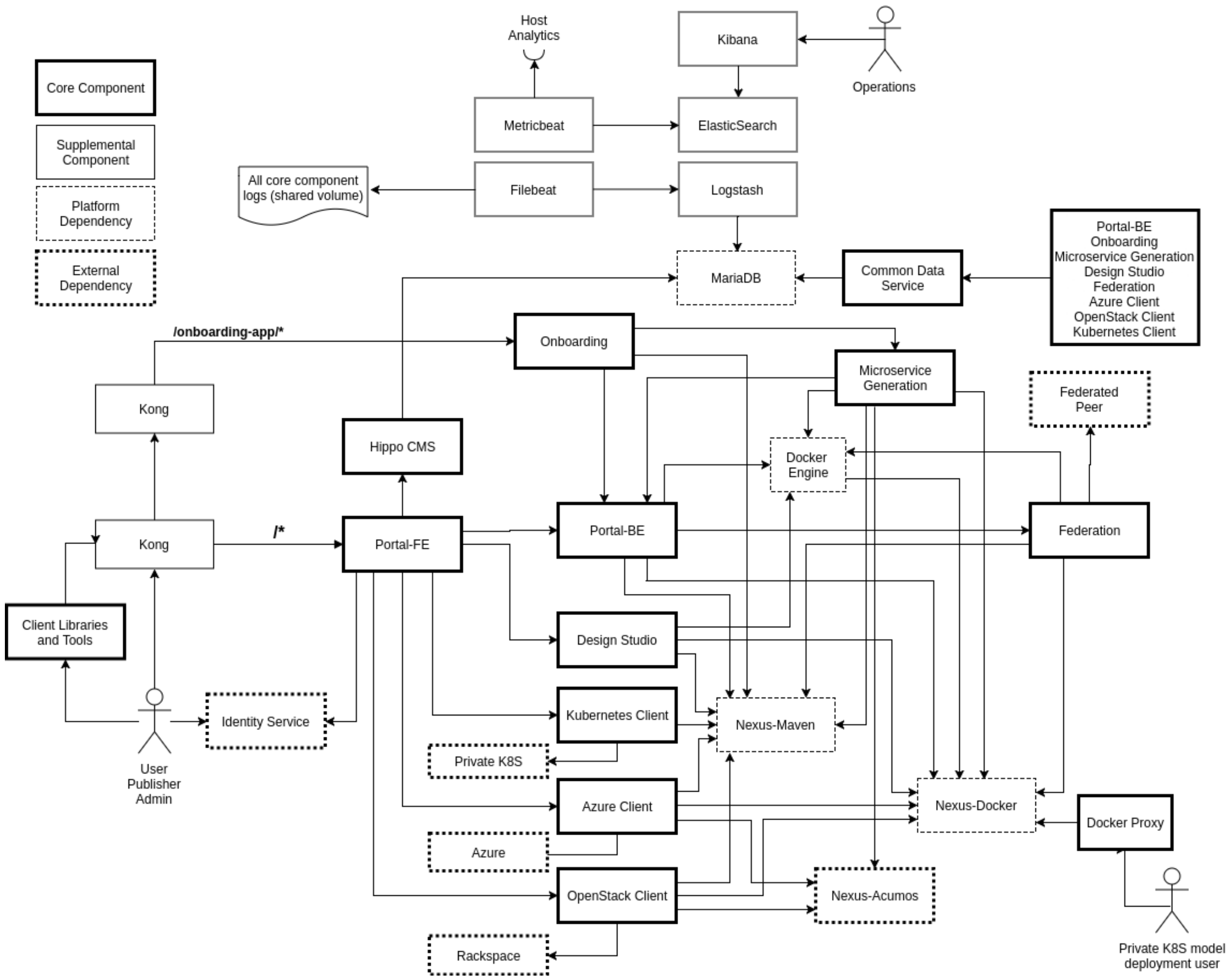
Component Interactions

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

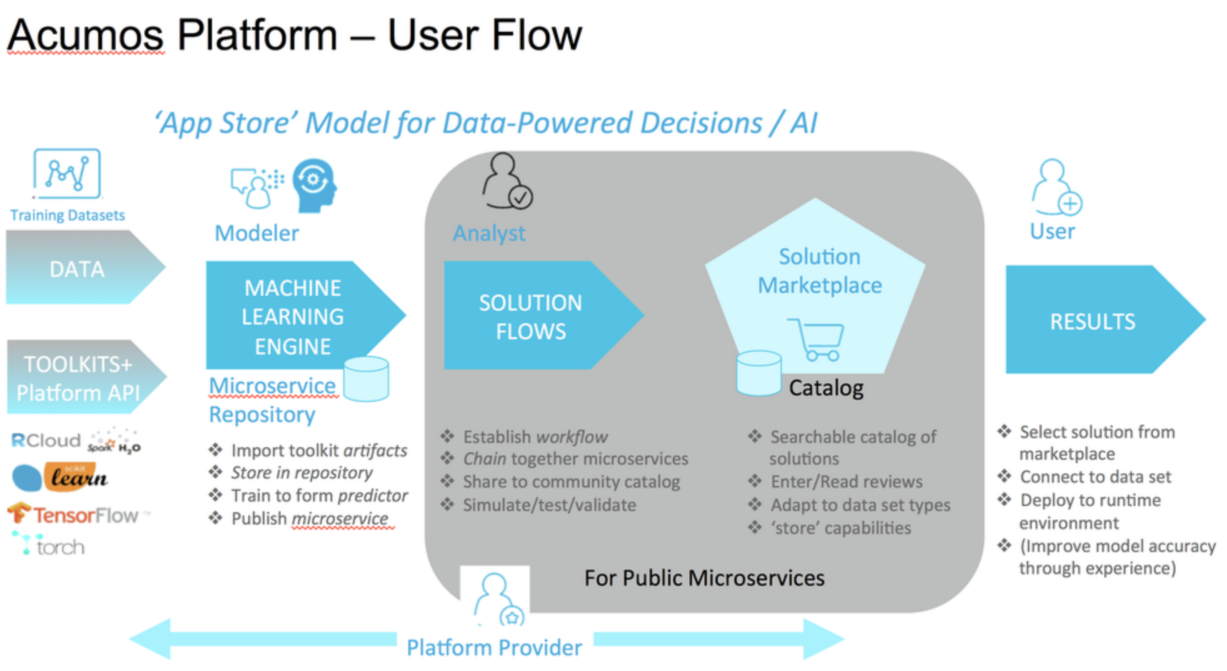
Kubernetes provides a container management environment in which the Acumos platform (as a collection of docker image components) and models can be deployed. Kubernetes cluster installation tools are provided by the kubernetes-client repo, and can be used for establishing a private kubernetres cluster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

Platform Flow

User Journeys

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

Acumos Platform User Flow



Acumos User Signup Flow

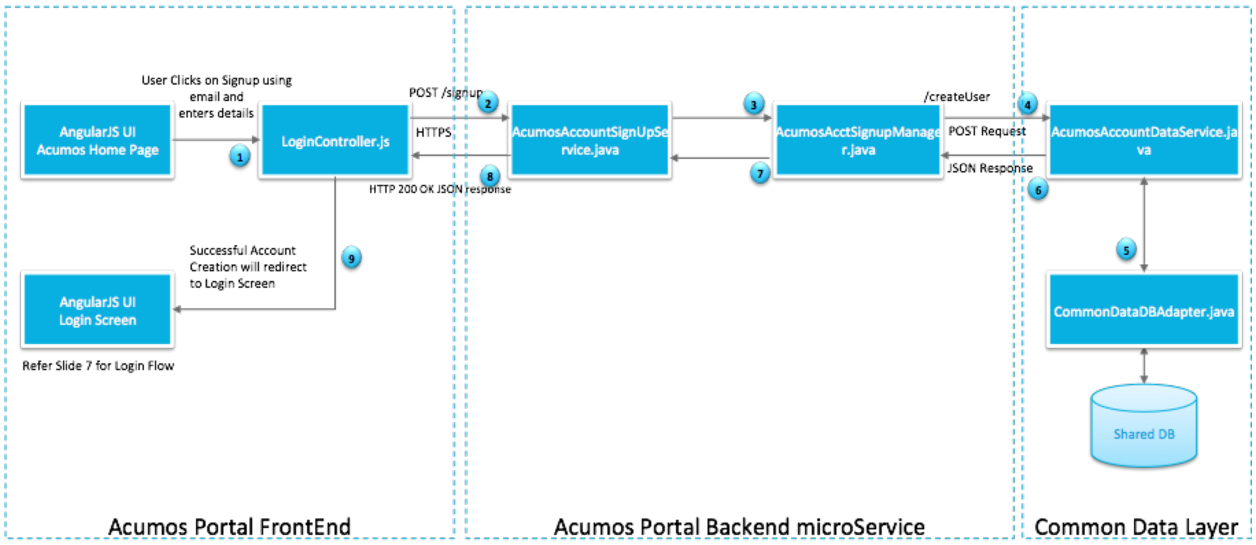
Acumos User Login Flow

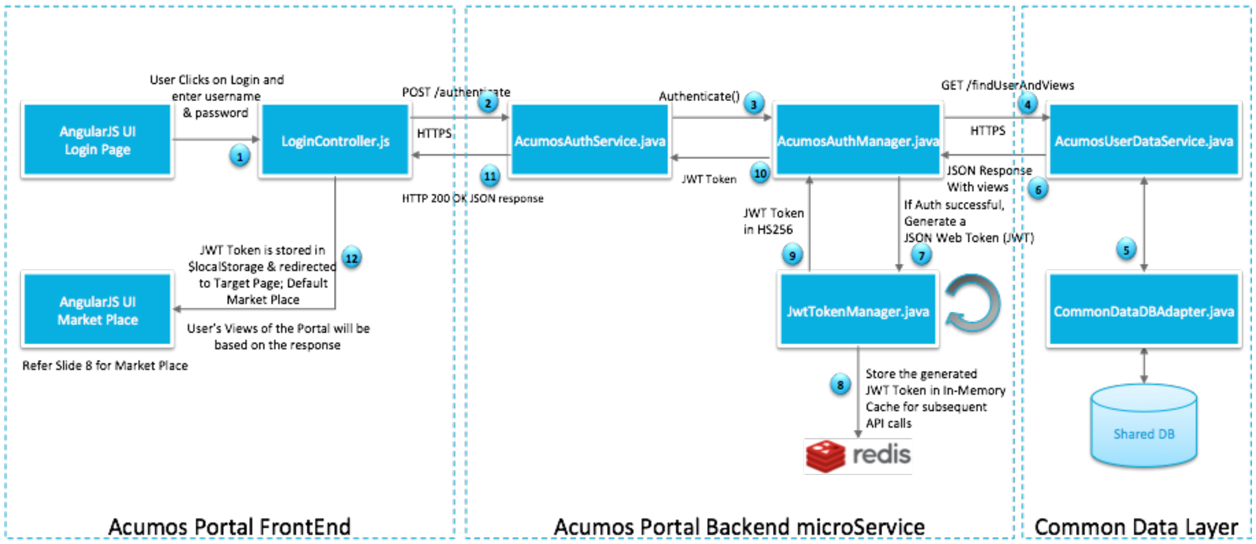
Component Interaction

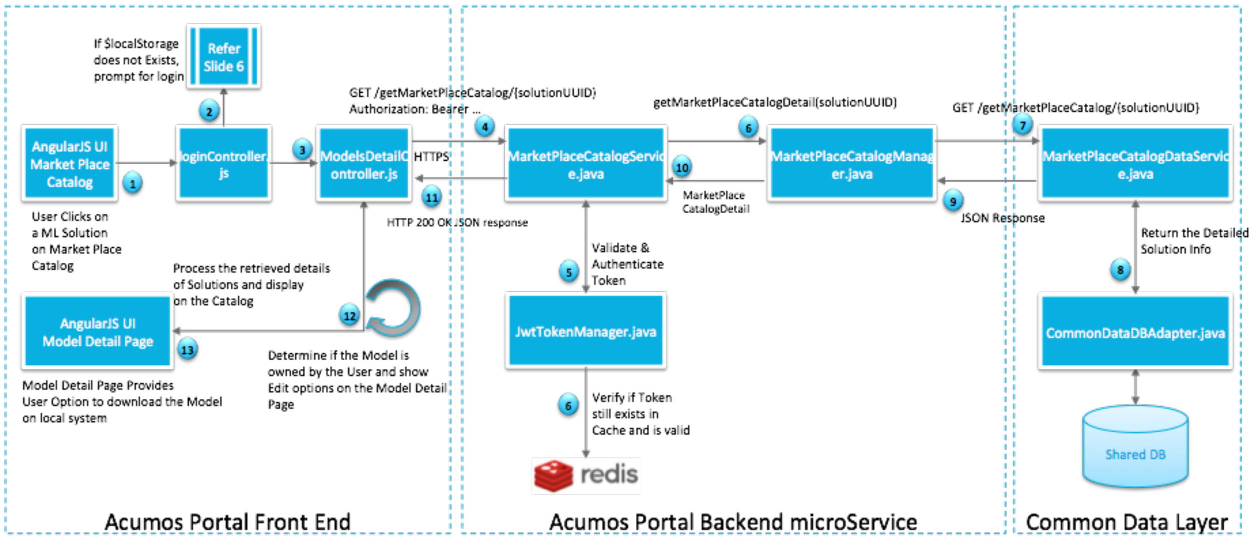
Following are some illustrative diagrams for common Acumos component interactions.

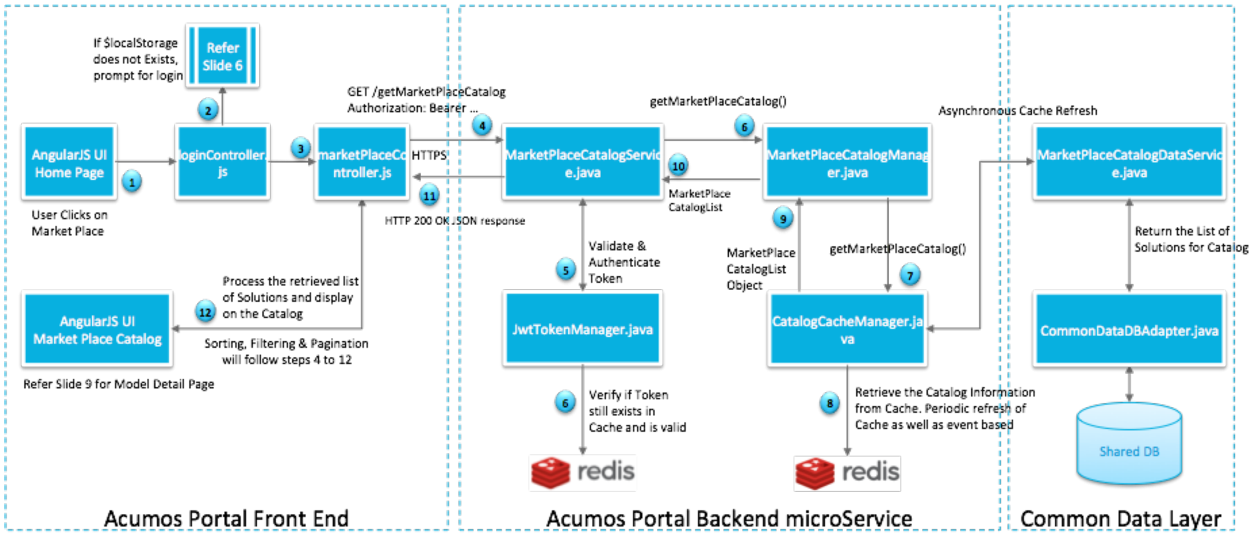
Acumos Model Detail Flow

Acumos Catalog Flow


User Clicks on Signup using email and enters details
AngularJS UI Acumos Home Page
1
LoginController.js
POST /signup
2
HTTPS
AcumosAccountSignUpService.java
3
AcumosAcctSignupManager.java
/createUser
4
POST Request
AcumosAccountDataService.java
JSON Response
6
7
8
HTTP 200 OK JSON response
5
CommonDataDBAdapter.java
Shared DB
Successful Account Creation will redirect to Login Screen
9
AngularJS UI Login Screen
Refer Slide 7 for Login Flow
Acumos Portal FrontEnd
Acumos Portal Backend microService
Common Data Layer


User Clicks on Login and enter username & password
AngularJS UI Login Page
1
LoginController.js
POST /authenticate
2
HTTPS
AcumosAuthService.java
Authenticate()
3
AcumosAuthManager.java
GET /findUserAndViews
4
HTTPS
AcumosUserDataService.java
JSON Response With views
6
11
HTTP 200 OK JSON response
JWT Token
10
JWT Token in HS256
9
If Auth successful, Generate a JSON Web Token (JWT)
7
5
JwtTokenManager.java
CommonDataDBAdapter.java
Shared DB
8
Store the generated JWT Token in In-Memory Cache for subsequent API calls
redis
JWT Token is stored in $localStorage & redirected to Target Page; Default Market Place
12
AngularJS UI Market Place
User's Views of the Portal will be based on the response
Refer Slide 8 for Market Place
Acumos Portal FrontEnd
Acumos Portal Backend microService
Common Data Layer


If $localStorage does not Exists, prompt for login
Refer Slide 6
2
AngularJS UI Market Place Catalog
1
loginController.js
3
ModelsDetailController.js
GET /getMarketPlaceCatalog/{solutionUUID} Authorization: Bearer ...
4
HTTPS
MarketPlaceCatalogService.java
getMarketPlaceCatalogDetail(solutionUUID)
6
MarketPlaceCatalogManager.java
GET /getMarketPlaceCatalog/{solutionUUID}
7
MarketPlaceCatalogDataService.java
11
HTTP 200 OK JSON response
10
MarketPlace CatalogDetail
9
JSON Response
User Clicks on a ML Solution on Market Place Catalog
Process the retrieved details of Solutions and display on the Catalog
12
AngularJS UI Model Detail Page
13
Model Detail Page Provides User Option to download the Model on local system
Determine if the Model is owned by the User and show Edit options on the Model Detail Page
5
Validate & Authenticate Token
JwtTokenManager.java
6
Verify if Token still exists in Cache and is valid
redis
Return the Detailed Solution Info
8
CommonDataDBAdapter.java
Shared DB
Acumos Portal Front End
Acumos Portal Backend microService
Common Data Layer


If $localStorage does not Exists, prompt for login
Refer Slide 6
AngularJS UI Home Page
loginController.js
marketPlaceController.js
GET /getMarketPlaceCatalog Authorization: Bearer ...
HTTPS
User Clicks on Market Place
HTTP 200 OK JSON response
Process the retrieved list of Solutions and display on the Catalog
AngularJS UI Market Place Catalog
Sorting, Filtering & Pagination will follow steps 4 to 12
Refer Slide 9 for Model Detail Page
MarketPlaceCatalogService.java
getMarketPlaceCatalog()
MarketPlaceCatalogManager.java
MarketPlace CatalogList
Validate & Authenticate Token
JwtTokenManager.java
MarketPlace CatalogList Object
getMarketPlaceCatalog()
CatalogCacheManager.java
Verify if Token still exists in Cache and is valid
Retrieve the Catalog Information from Cache. Periodic refresh of Cache as well as event based
redis
Asynchronous Cache Refresh
MarketPlaceCatalogDataService.java
Return the List of Solutions for Catalog
CommonDataDBAdapter.java
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.

Catalog, Data Model, and Data Management

- Common Data Service
- Federation Gateway
- Model Schema

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

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")

Deployment

This project maintains clients for deploying models to different environments.

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

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

Portal and Marketplace

- Acumos Hippo CMS
- Portal

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

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