
Acumos Documentation

Release 1.0

The Acumos Project. Licensed under CC BY 4.0.

Nov 01, 2019

Contents

1	Acumos Releases	3
1.1	Platform	3
1.1.1	Athena Maintenance Release, 12 December 2018	3
1.1.2	Athena Release, 7 Nov 2018	8
1.2	Component and Weekly	13
1.2.1	Release Notes	13
2	Acumos Portal and Marketplace User Guides	17
2.1	Portal and Marketplace User Guide	17
2.1.1	Platform Overview	17
2.1.2	Creating and Using an Account on Acumos	18
2.1.3	The Marketplace - For Consumers	23
2.1.4	The Portal - For Modelers	38
2.1.5	The Design Studio: For Composers	61
2.2	Portal and Marketplace Publisher Guide	84
2.2.1	Publishing Models	84
2.3	Portal and Marketplace Admin Guide	87
2.3.1	Introduction	87
2.3.2	Site Monitoring	88
2.3.3	User Management	88
2.3.4	Site Content	90
2.3.5	Site Configuration	90
2.3.6	Federation	90
2.3.7	Requests	97
2.3.8	Configure Workflows	97
2.3.9	Addendum	100
3	Acumos Operational User Guides	103
3.1	One Click Deploy User Guide	103
3.1.1	Introduction	103
3.1.2	Release Scope	104
3.1.3	Step-by-Step Guide	105
3.1.4	Logs Location	113
3.1.5	Additional Notes	113
3.2	Platform Operations, Administration, and Management (OA&M) User Guide	114
3.2.1	Acumos Elastic Stack for Log Analytics	114
3.3	System Integration User Guide	126

3.3.1	Acumos API Management with Kong	126
4	Contributors to the Acumos Platform Code	137
4.1	Platform Architecture	137
4.1.1	Architecture Guide	137
4.2	Component Guides	156
4.2.1	Component Guides	156
4.3	Documentation Guide	157
5	Indices and Tables	159

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

Note: Athena Maintenance Release There is a required database upgrade to populate Authorship data. Please see [User and Author Data Upgrade for CDS 1.18.x](#) for instructions.

1.1.1 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 search-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

1.1.2 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 *On-Boarding Design Studio Toolkit Models* 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	gdm-service	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

- Azure Client
- Kubernetes Client
- OpenStack Client

Model Onboarding

- Java Client
- Onboarding
- 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](#).

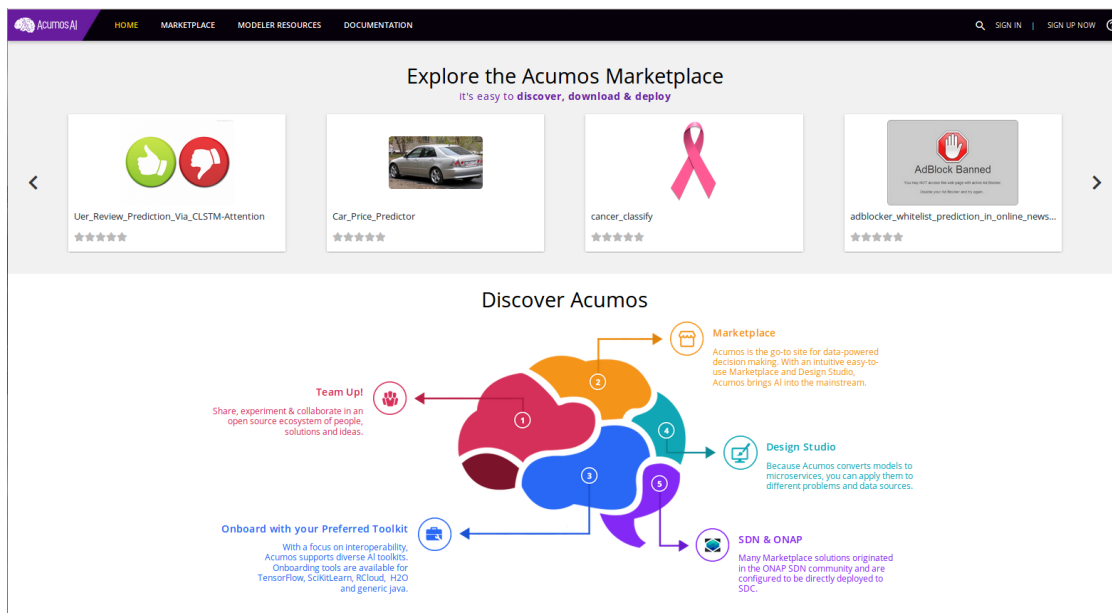
Weekly builds may be unstable and are not recommended for deployment to a production environment.

Acumos Portal and Marketplace User Guides

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

2.1 Portal and Marketplace User Guide

2.1.1 Platform Overview



The Portal is a web-based tool designed for users who wish to either explore and use machine learning models or data scientists who build models and wish to share them with a larger community.

The Acumos Portal has two sections: (1) The Marketplace and associated tools for people wishing to find, learn about and use (deploy or download) models in their own computing environments; and (2) the Modeler sections for people wishing to share, describe and market their models. Modeler tools are available in several areas of the site, including Onboarding, the Design Studio, My Models, and Manage My Model.

The Marketplace and Modeler communities may communicate and interact via the commenting and rating tools for users.

It has the following key features:

1. **Easy Onboarding of Models.** The Acumos Portal enables modelers to easily onboard their AI models, document them, and package them into reusable microservices. Newly onboarded models are located in the PRIVATE/UNPUBLISHED catalog, viewable only by the user. The Modeler may publish them to either their Company marketplace (viewable by others with logins on their local Acumos instance) or to the PUBLIC Marketplace, where they may be distributed to a wider community.
2. **Explore the Possibilities of AI.** The Marketplace enables users to explore, gathering high-level or detailed information about a model and how it is used. Users have access to extensive documentation, as well as ratings and comments from the greater community.
3. **Model Packaging.** Models are packaged as Dockerized microservices so they can be deployed directly to a variety of environments. As a part of the onboarding process, protobuf files are created to describe the model inputs and outputs in a standard format, and a TOSCA file is generated to allow the model to be accessed in the Design Studio.
4. **Model Access.** Once a model is selected, the Acumos Marketplace provides access to that model, either via downloading or by deploying it to any one of several commonly used cloud environments. Specialty access may be available on some instances. For example, models may be specifically designed to work on ONAP instances.
5. **Compose.** Users may also work with models in the Design Studio, chaining them together to create new custom composite models to help solve specific business problems. These composite models can be saved and managed (validated, deployed, published) just like simple models.
6. **Federation.** The Acumos Marketplace supports distributed relationships with other Acumos peer instances, allowing users to browse and procure models from remote federated partners. Federation also creates a much larger available user pool for data scientists to share their models.

Users are welcomed to Acumos on the home page, showing a carousel highlighting Acumos features and uses. Other parts of the page may show featured or trending models, upcoming Acumos events and illustrations of how Acumos can help customize solutions in many domains. The Acumos homepage is customizable, so your home page may differ from what is described here.

Note: Users do not need to be logged into Acumos or have an account to see the Home and Marketplace pages, but users must be logged in to download or deploy models.

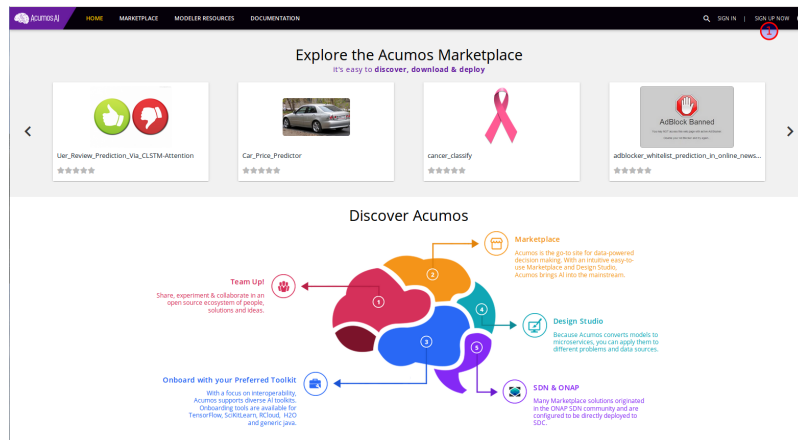
2.1.2 Creating and Using an Account on Acumos

Creating a User Account

In order to use the full capabilities of Acumos, users must create an account on the Acumos Portal. The user may also complete a user profile. Depending on the Acumos instance configuration, custom company login options may be supported.

Account management capabilities are available in the upper right corner of the user interface.

1. Click on Sign Up Now link on Acumos Home screen.



2. Fill out information in the Sign Up window and click the **Sign Up** button on the bottom of the screen.

Sign up to continue

First Name *

Last Name *

User Name *

Email ID *

Password *

Your password must contain at least eight characters, which should have at least one upper case and one lower case letter, numbers and symbols like, ! # @ \$ * &.

Confirm Password *

Sign Up

Already have an Acumos ID? [Sign In](#)

Sign In with Github

Sign In with Facebook

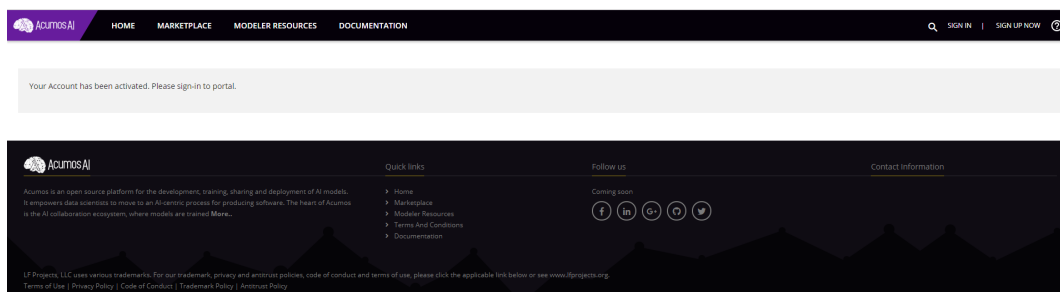
Sign In with Google

Sign In with LinkedIn

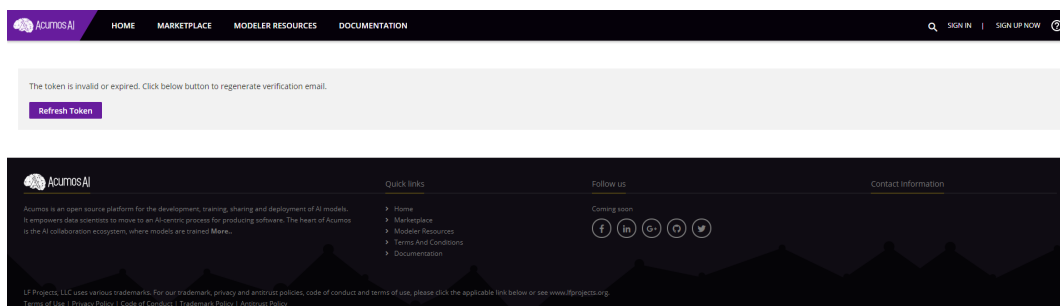
or

By signing up, you agree to our [Terms and Conditions](#) and [Privacy Policy](#).

3. An account verification email will be sent to the email address that you entered. You cannot sign-in without verifying your email address. The verification link will be valid for a finite time period, as configured by the Acumos Platform Administrator.
4. Click on the link in the account verification email to activate your account. Clicking the link will take you to the Acumos verification page, where you should see the following:



5. After successfully verifying your email address, you can log into Acumos. See the [Logging into Acumos](#) section below.
6. If the account verification link has expired, you will get an error message. Click the **Refresh Token** button to generate a new account verification email.

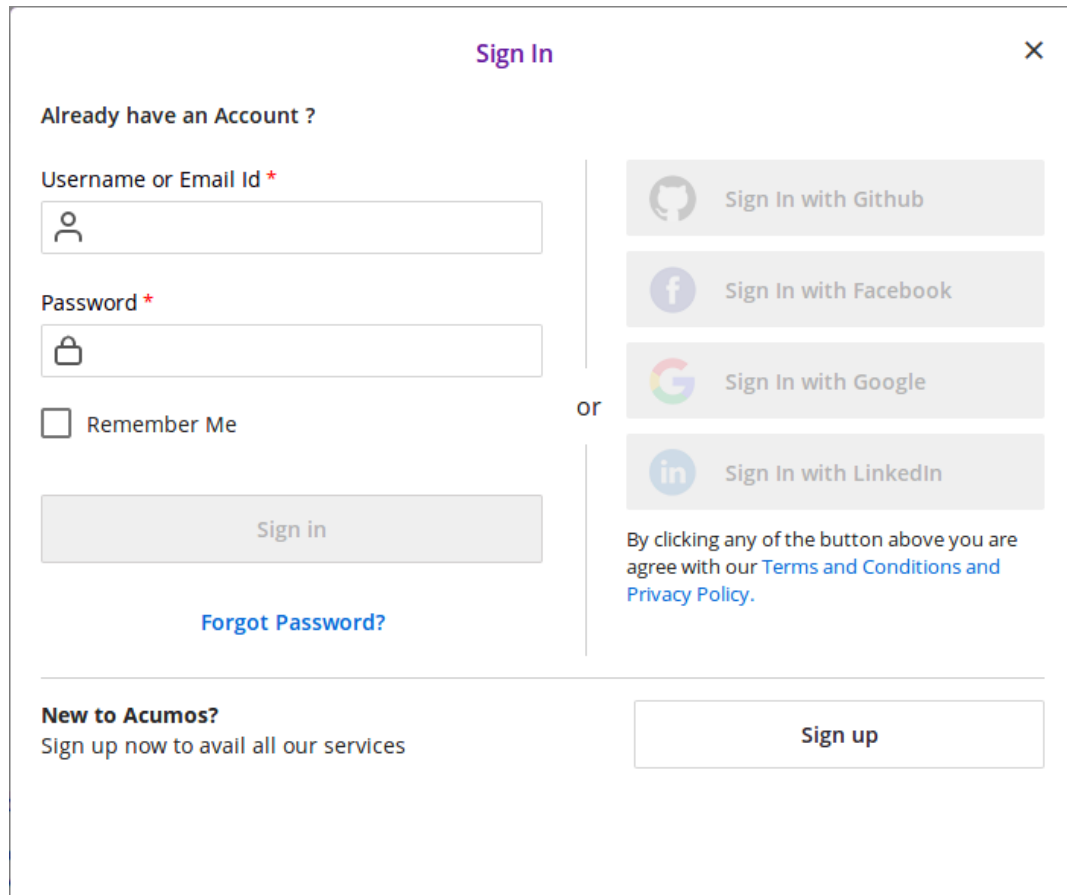


Note: Account creation may be customized on individual Acumos installations. If what you see on your local Acumos instance is different than what is in this guide, please consult your local Acumos Admin for assistance.

Logging into Acumos

The steps to login to Acumos are as follows:

1. Click on the **SIGN IN** link on the top right corner of the Home Page.



The image shows a 'Sign In' dialog box with a close button (X) in the top right corner. The title 'Sign In' is centered at the top. Below the title, the text 'Already have an Account ?' is displayed. The form is divided into two main sections. The left section contains a 'Username or Email Id *' field with a person icon, a 'Password *' field with a lock icon, a 'Remember Me' checkbox, a 'Sign in' button, and a 'Forgot Password?' link. The right section features four social login buttons: 'Sign In with Github', 'Sign In with Facebook', 'Sign In with Google', and 'Sign In with LinkedIn'. Between these sections is a vertical line with the word 'or' in the middle. At the bottom of the right section, there is a disclaimer: 'By clicking any of the button above you are agree with our [Terms and Conditions](#) and [Privacy Policy](#).' Below the main form area, there is a section for 'New to Acumos?' with the text 'Sign up now to avail all our services' and a 'Sign up' button.

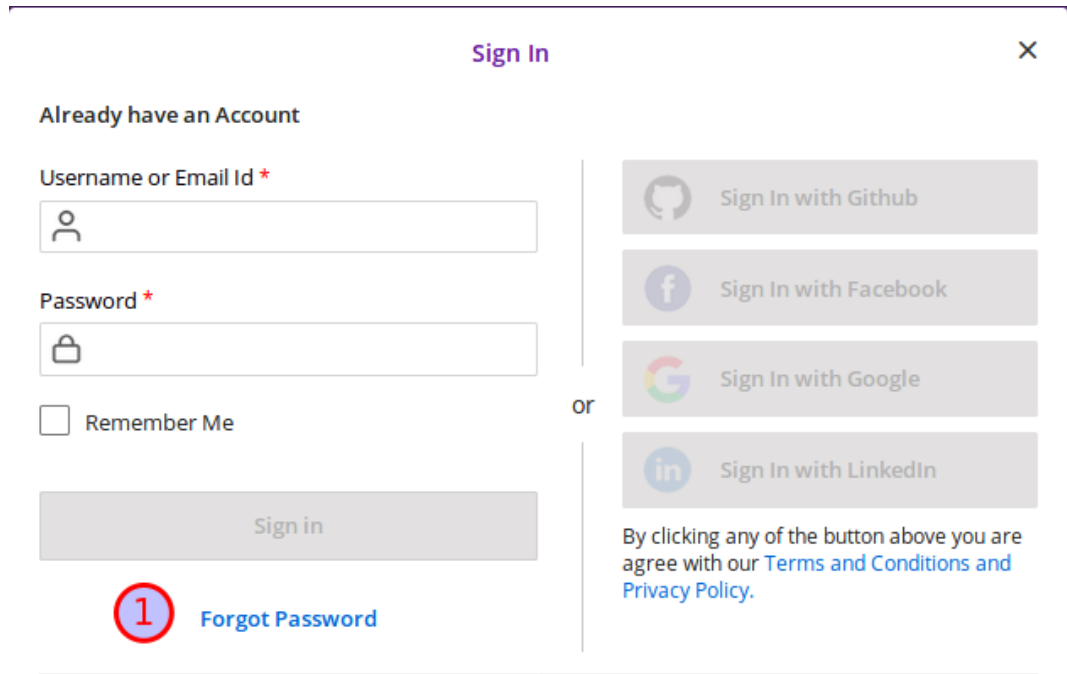
2. Fill in the username and password.
3. Click **Sign in** on the bottom of the screen.

Note: Account log in may be customized on individual Acumos installations. If what you see on your local Acumos instance is different than what is in this guide, please consult your local Acumos Admin for assistance.

Resetting a Password

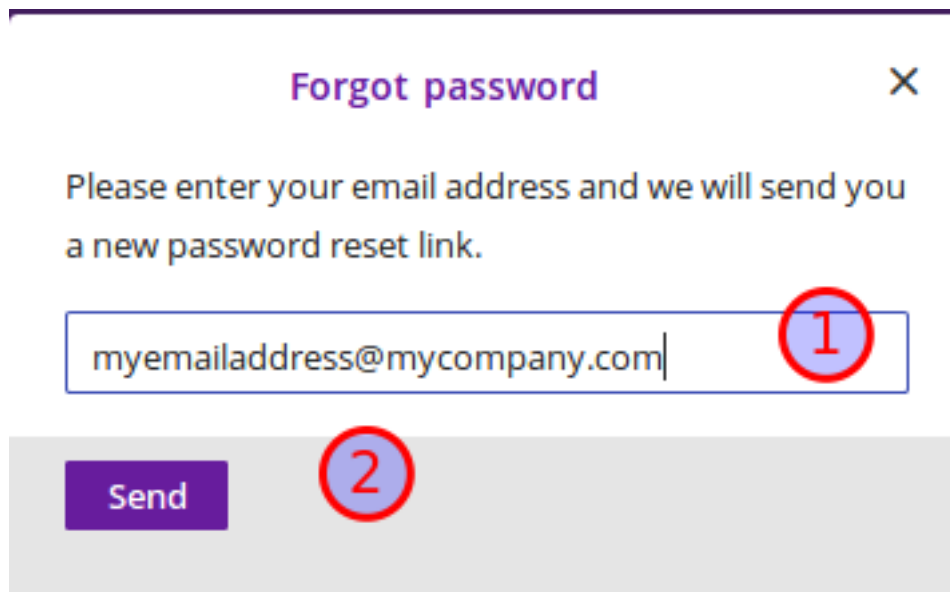
The steps to reset a password are as follows:

1. From the **Sign in** window, click the **Forget Password** link



The image shows a 'Sign In' dialog box with a title bar and a close button (X). It contains a section for users who 'Already have an Account'. This section has two input fields: 'Username or Email Id *' and 'Password *'. Below the password field is a 'Remember Me' checkbox. A 'Sign in' button is positioned below these fields. A red circle with the number '1' is placed over the 'Forgot Password' link. To the right of the input fields, there is a vertical line with the word 'or' in the center. To the right of this line are four buttons for social login: 'Sign In with Github', 'Sign In with Facebook', 'Sign In with Google', and 'Sign In with LinkedIn'. Below these buttons is a disclaimer: 'By clicking any of the button above you are agree with our Terms and Conditions and Privacy Policy.'

2. Enter the email address associated with the account and then press the **Send** button



The image shows a 'Forgot password' dialog box with a title bar and a close button (X). It contains a message: 'Please enter your email address and we will send you a new password reset link.' Below this message is a text input field containing the email address 'myemailaddress@mycompany.com'. A red circle with the number '1' is placed over the end of the input field. Below the input field is a 'Send' button. A red circle with the number '2' is placed over the 'Send' button.

Note: Log in and password reset may be customized on individual Acumos installations. If what you see on your local Acumos instance is different than what is in this guide, please consult your local Acumos Admin for assistance.

Setting Profile and Notification Preferences

Your User Profile is designed to give your users a view of your work. When you publish a model, either to your Company instance or to the Public, your profile is always available by clicking on your name.

To update your user profile, click on your name in the upper right corner and then choose **Account Settings**.

Account Settings
Home / Account Settings

PROFILE SETTINGS NOTIFICATION PREFERENCES

PROFILE

First Name *
James

Last Name *
Kirk

Change Photo
Change Password

API Token

Api Token
bd24178e53994931ad2be8831d1b3747 Refresh ①

CONTACT SETTINGS

Email *
aimeeu@research.att.com Change Email ②

1. The API Token is used to onboard models from the command line
2. When you change your email address, you will be automatically logged out of the application and must log in again

From time to time, you may wish to be notified if a process, such as requesting access to a model, has completed. To set up your notification preferences, access the **Notification Preferences** tab.

Account Settings
Home / Account Settings

PROFILE SETTINGS **NOTIFICATION PREFERENCES**

* How do you want to be notified?

☒ Email

Enter Email ID *
aimeeu.opensource@gmail.com

Set Notification Priority
Select ▼
Low
Medium
High

Cancel Save

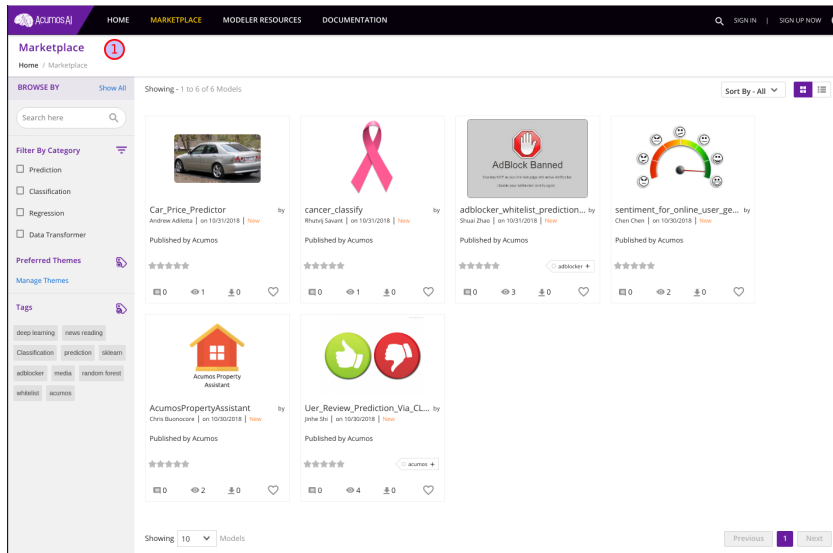
2.1.3 The Marketplace - For Consumers

Overview

The Acumos Marketplace is designed to make it easy to discover, explore, and use AI models. In addition to displaying models from the local platform's catalog, the Public Marketplace can be configured to display models from peer Public Marketplaces. Users may view the details of a peer Model. However, users aren't able to work with, download, or

deploy a peer Model without first requesting access to use that model. Public Marketplace peer relationships are set up and managed by the administrators of each Acumos instance.

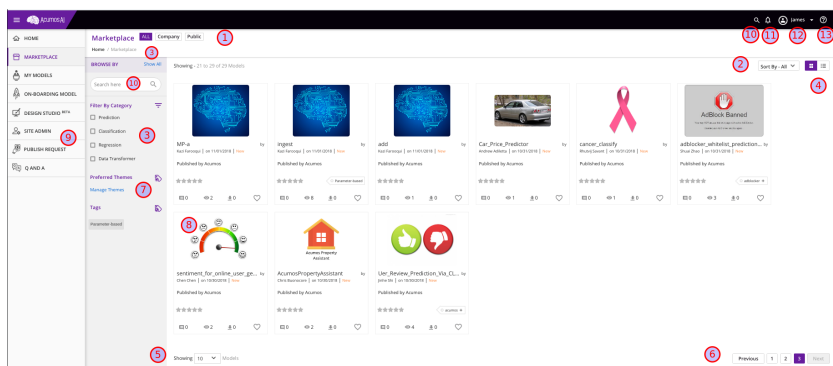
The Marketplace only displays Public models to users who have not logged in. There is no option to display models from marketplaces other than Public.



The Marketplace has three main views to logged-in users:

- **Company:** The Company Marketplace only shows models which have been onboarded by local users and published to the local marketplace catalog
- **Public:** Models in the Public Marketplace are visible to all Acumos instances with a peer relationship
- **ALL:** This choice displays models in both the user's Company and Public Marketplace

After you have used models on the Marketplace, you can share your experiences by using the comments and ratings capabilities on the Marketplace.



1. Select **Company** or **Public** to display models in those marketplaces, or select **ALL** to view models in both marketplaces
2. Select an option from the **Sort By** drop down to sort the displayed models
3. Select one or more checkboxes to **Filter By Category**; clicking the **Show All** link display additional categories if they exist; see the *Filtering by Category* section for details
4. Change from grid view to list view by selecting the corresponding icon
5. Select an option from the **Showing** drop down to change the number of models displayed on a page

6. Page navigation
7. Click the **Manage Themes** link to select, update, or remove your tag choices; see the [Manage Themes](#) section below for instructions
 - 7a) Select a **tag** to filter the displayed models by tag; see the [Filtering by Tag](#) section for details
8. Click a model's image to access the **Model Detail** page
9. **SITE ADMIN** and **PUBLISH REQUEST** are menu items only available to users with those roles
10. Model search; see the [Searching by Keyword](#) section for details
11. Click the **Bell** icon to review your notifications
12. Click the down arrow next to your name to access **Account Settings** and **Manage Themes**
13. **Help** and **Log Out**

Manage Themes

Themes in this context refers to which types of models should be displayed before the rest. Themes equate to tags that have been associated with models. All the model tags are displayed initially in the **Manage Themes** window.

What are your interests?

Start Finish

Choose your tags to customize your marketplace experience Filter

acumos	adblocker	Array-based	C	Cars	Classificat...
CNN	Collator	Databroker	deep learning	entertain...	flask
Fraud-detection	keras	LSTM	media	news reading	Parameter-based

Continue

1. You can type in the **Filter** field to narrow the displayed list

What are your interests? ×

Start Progress Bar Finish

Choose your tags to customize your marketplace experience

Filter 1 🔍

✓ 2

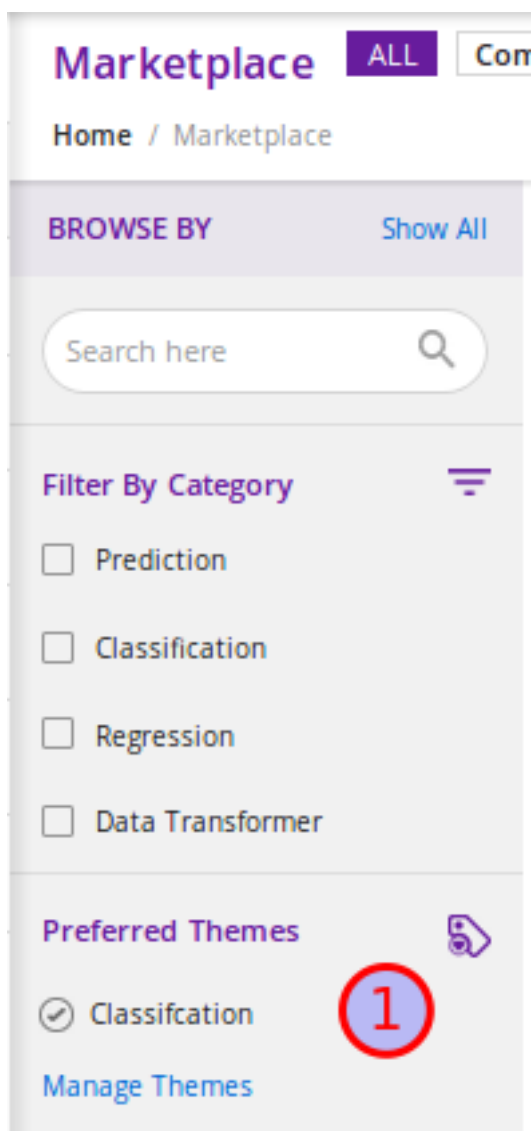
Selected Tags : Classification,

3

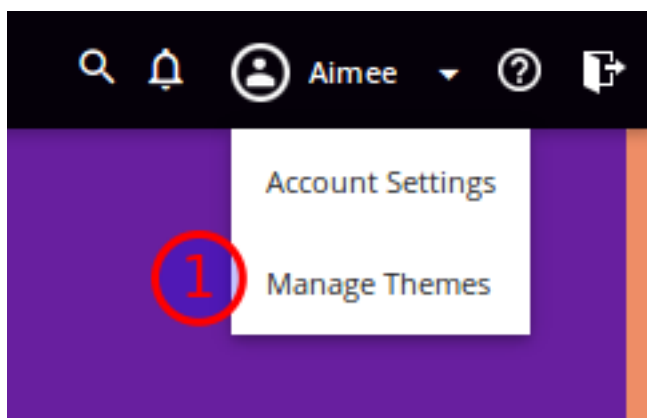
2. Select the themes that interest you

3. Press **Continue** and then in the following window, press **Done** to save your changes

Preferred themes are displayed on the left side of the UI.



Manage Themes may also be accessed from User drop-down in the upper right corner of the UI.



Searching by Keyword

You can search models in the marketplace to find keyword matches in the following fields: name, description, author, publisher, solution ID and revision ID. To search the Marketplace by keywords, follow these steps:

1. Enter keywords in the search field near the top of the left navigation bar
2. Hit return or click the search icon to start the search
3. The result of your query is shown, with only the models that meet your search criteria

Filtering by Category

To filter your view of the Marketplace by Category, follow these steps:

1. From the Marketplace left inner menu, select **Show All** to show all categories
2. Click on a Category to select it
3. The screen is updated with only models that have your selected Category

Filtering by Tag

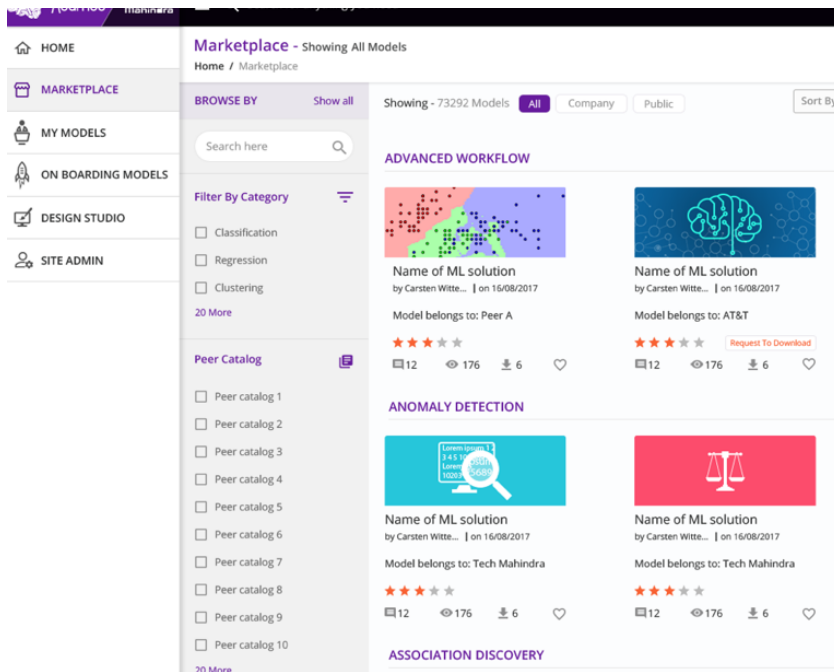
To search the Marketplace using Tags, follow these steps:

1. From the Marketplace left inner menu, click on the Tag of interest
2. The Tag will become highlighted
3. The screen is updated with only the models that have your selected tag

Filtering by Peer

To see models from federated peers, choose the PUBLIC marketplace. The default view shows all public models from your local Acumos as well as all models from all peers. This functionality is only available if your Acumos installation has been connected to other Acumos installations.

To restrict your view to a particular peer, or set of peers, adjust the filters for Peer Catalog.



Viewing Model Details

Much more information about a model is available on a Model Details page. From the search results, clicking on any model image shows the Model Detail page for that model. Sections on the Model Detail page include:

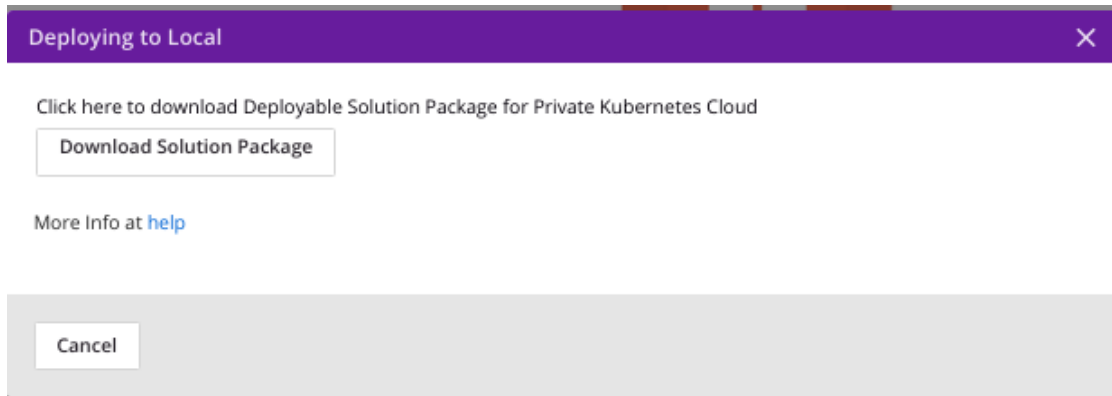
- Introduction
- Ratings
- Comments
- Signatures: the description of the model incoming data feed and output
- Documents: any supporting documentation the Modeler wishes to include
- Version History
- Author/Publisher Details

Most of the information on the detail page is contributed by the creator of the model to showcase the model and tell potential users about its capabilities and use. Ratings and Comments are contributed by other users of this model.

Deploying a Model

Deploying to Local

Clicking the **Deploy to Local** button opens a pop-up window from which you are able to download the solution package for deployment to a local/private Kubernetes cloud.



Deploying to Azure

Logged-in users may deploy a model's microservice to Microsoft Azure.

Microsoft Azure is a cloud computing service created by Microsoft for building, testing, deploying, and managing applications and services through a global network of Microsoft-managed data centers. Acumos is able to deploy models and composite models easily in an Azure cloud. It creates a new virtual machine and deploys the model in that VM.

A Composite Solution is combination of more than one model. The Model Connector is also deployed with a composite solutions since it is used for communication between models in the VM. Databroker details can be specified for a composite model. If a Data Broker image is available in the composite solution, then it will also be deployed. Upon successful deployment, the user is notified of the model connector endpoint and optionally, the Zip or CSV Databroker.

This guide is intended to help you:

- setup a Microsoft Azure account for use in deploying Acumos models
- access the deployed Azure VMs for testing, etc

This guide assumes that you have an Azure account. If not, you can sign up for a free trial account at <https://azure.microsoft.com>.

Configuring Your Azure Account

This section will help you setup your Azure account for use in launching Acumos models and save key account attributes for use in the Acumos portal "Deploy to Cloud" dialog. A summary of the minimum attributes you will need is below, followed by instructions on creating/obtaining these attributes.

- Application Id: "Application Id" value for your Azure Active Directory application
- Tenant Id: the "Directory ID" value under "Azure Active Directory" / "Properties"
- Secret key: application key "Value" created through "Settings" / "Keys" for your Azure Active Directory application
- Subscription Key: your "Subscription Id"
- Resource Group: Name of your Resource group
- Acr Name: Name of your Azure Container Registry (ACR)
- Storage Account: Name of your Azure Storage Account

Copy Tenant Id

The Acumos platform needs your Azure account “Azure Active Directory” ID during model launch in Azure. Note this ID is only provided by you in the “Launch in Azure” dialog, and is not retained by the Acumos platform. To copy this ID for later use, under “Azure Active Directory”:

- select “Properties”
- copy the “Directory ID” value and save for later use as the “Tenant Id” in the “Launch in Azure” dialog

Copy Subscription Key

The Acumos platform needs your Azure account “Subscription Id” during model launch in Azure. Note your “Subscription Id” is only provided by you in the “Launch in Azure” dialog and is not retained by the Acumos platform. To copy this ID for later use, under “Cost Management + Billing”:

- select “Subscriptions”
- select “My subscriptions”
- copy your “Subscription Id” and save for later use as the “Subscription Key” in the “Launch in Azure” dialog

Create an Azure Active Directory application entry for the Acumos portal

As described at [here](#), the Acumos platform represents “code that needs to access or modify resources” in Azure, and to be given those permissions, needs to be registered as an Azure Active Directory (AD) application. The ID assigned to this application will be used by the Acumos platform during model launch in Azure. Note this ID is only provided by you in the “Launch in Azure” dialog and is not retained by the Acumos platform. To register the Acumos platform as an application and copy its ID for later use, under “Azure Active Directory”:

- select “App registrations”
- select “+” (create new)
- enter a name
- enter any URL (this is not used for Acumos)
- set “Application type” to “Web app / API”
- select “Create”
- copy the displayed “Application Id” for later use as the “Application Id” in the “Launch in Azure” dialog

Create a Secret Key for the Azure Active Directory Application

The Acumos platform needs to be assigned a secret key in order to access your Azure account during model launch in Azure. Note this secret key is only provided by you in the “Launch in Azure” dialog and is not retained by the Acumos platform. To create and copy this secret key for later user, under “Azure Active Directory”:

- select “App registrations”
- select the Azure Active Directory application registered above
- select “Settings”
- select “Keys”
- under “Passwords” enter a key name and duration

- select “Save”
- copy the displayed application key “Value” for later use as the “Secret Key” in the “Launch in Azure” dialog

Add Admin User permission for Your Azure Subscription

The Acumos platform needs to be able to create resources under your subscription, in order to deploy models there. To enable this, you need to assign the the Azure Active Directory application registered above with permissions as “Contributor” and “Reader”, as below. To create the needed permissions, under “Cost Management + Billing”:

- select “Subscriptions”
- select “My subscriptions”
- select your subscription
- select “Access control (IAM)”
- select “+Add”
- set “Role” to “Contributor”
- set “Assign access to” to “Azure AD user, group, or application”
- under “Select” enter the Azure Active Directory application name you created above
- select the Azure Active Directory application name when it is displayed below
- select “Save”

Repeat this process, selecting this time the “Role” as “Reader”.

Note that all resources you create below will now inherit these Admin user permissions, so you do not have to set permissions for each resource.

Create a Resource Group

Resource Groups allow management of resources with a common (“inherited”) set of permissions and other options. The Acumos platform needs the name of a resource group under which you have created some basic resources (as described later) during model launch in Azure. Note this Resource Group name is only provided by you in the “Launch in Azure” dialog and is not retained by the Acumos platform.

If you don’t have an existing resource group in the East US region, you will need to create one. To create a resource group, under “Resource groups”:

- select “Create”
- enter a “Name” (you will use this name later in the “Launch in Azure” dialog)
- select “East US” (this location is currently required by the Acumos platform)
- select “Create”
- refresh the list of Resource groups and the new one should be displayed
- select the resource group

Create an Azure Container Registry (ACR)

An ACR is needed so the Acumos platform can push container images to your Azure account, for launch inside VMs created in the process of model deployment. Using an ACR under your Azure account can speed up model deployment. The Acumos platform needs the name of this ACR during model launch in Azure. Note this ACR name is only provided by you in the “Launch in Azure” dialog and is not retained by the Acumos platform.

If you don’t have an existing ACR in the East US region, you will need to create one. To create a new ACR, under “All resources”:

- select “+Add”
- select “Containers”
- select “Azure Container Registry”
- enter a “Name” (you will use this name later in the “Launch in Azure” dialog)
- set “Resource group” to “Use existing”
- from the drop-down list, select the resource group created above
- select “East US” (this location is currently required by the Acumos platform)
- select “Create”

Create a Storage Account

A storage account is needed so the Acumos platform can install applications and save data to disk as needed during model deployment and execution. The Acumos platform needs the name of this storage account during model launch in Azure. Note the storage account name is only provided by you in the “Launch in Azure” dialog and is not retained by the Acumos platform.

If you don’t have an existing storage account in the East US region, you will need to create one. To create a storage account, under “Storage accounts”:

- select “+Add”
- enter a “Name” (you will use this name later in the in the “Launch in Azure” dialog)
- set “Resource group” to “Use existing”
- from the drop-down list, select the resource group created above
- select “East US” (this location is currently required by the Acumos platform)
- select “Create”

Create a Network Security Group and Access Rules

A network security group (NSG) is required so that the Acumos platform can configure access to necessary host ports on the deployed VMs. A specific NSG name is required by the Acumos platform. To create this NSG, under “Resource groups”:

- select the resource group created above
- select “+Add”
- in the search box, enter “Security”
- in the resulting list, select “Network Security Group”

- select “Create”
- set “Name” to “E6E-NSG”
- set “Resource group” to “Use existing”
- from the drop-down list, select the resource group created above
- select “East US” (this location is currently required by the Acumos platform)
- select “Create”

To enable the Acumos platform to access VMs it deploys, deploy model microservices, and deploy additional components that help orchestrate and connect the models to your data sources, you will need to define NSG rules to open the following TCP ports to the Acumos platform. The Acumos platform will be identified here by IP address; you can get the IP address using a reverse-DNS lookup e.g. ‘nslookup marketplace.acumos.org’. Ports that need to be opened, and their purpose, are described below:

- TCP port 22: SSH, enabling the Acumos platform to configure the deployed VM, e.g. install docker and the various microservices and platform components
- TCP port 8555: Acumos Blueprint Orchestrator, used in Composite Solution deployment
- TCP port 8556: Acumos DataBroker, a component deployed when a user wants assistance in mapping a data source to the protobuf interface of a deployed model (details will be provided for when this applies and how the user selects it)
- TCP port 5006: Acumos Probe, a component enabling the user to access and visualize the protobuf interfaces of their deployed solutions

If you need to provide access to your model microservices from outside the Azure virtual network, e.g. to push data to the microservice, you will need to create additional NSG rules to open the following ports to the IP addresses of systems to be connected to the microservices:

- TCP port 8557: microservice #1, i.e. for deployment of a single model microservice (“Simple Solution”) or the first microservice in a multi-model deployment (“Composite Solution”)
- TCP port 8558: microservice #2
- TCP port 8559: microservice #3
- TCP port 8560: microservice #4
- and so on

To add NSG rules, under “Resource groups”:

- select the resource group created above
- select NSG “E6E-NSG”
- select “Inbound security rules”
- select “+Add”
- set “Source” to the IP address of the system that needs the access
- select “Protocol” “TCP”
- set “Destination port ranges” to the specific port or range of ports that applies to the rule
- set “Name” to whatever helps you remember what the rule is related to
- select “Add”

Repeat this for any other hosts you want to have access to the VM, and for any other access rules that are needed for your deployed model or applications to be installed on or connected to the deployed VM.

Note: it is recommended to NOT set “Source” to “Any” and “Destination port ranges” to “*” as these settings can expose your VM to security risks.

Create a Virtual Network

A virtual network and subnet is required so that required ports can be opened on the VM in which Acumos will launch your model. Acumos requires a specifically named virtual network and subnet, since it will create interfaces and public IP addresses on that network/subnet.

To create the specified virtual network, under “Resource groups”:

- select the resource group created above
- select “+Add”
- enter “Networking” in the search bar and hit enter
- in the resulting list, select “Virtual network”
- select “Create”
- set “Name” to “Acumos-OAM-vnet”
- set “Resource group” to “Use existing”
- from the drop-down list, select the resource group created above
- select “East US” (this location is currently required by the Acumos platform)
- set “Subnet” to “Acumos-OAM-vsubnet”
- select “Create”

Associate the NSG to the Subnet

To ensure the NSG rules created above are applied to the subnet you created, under “Resource groups”:

- select the resource group created above
- select the virtual network “Acumos-OAM-vnet”
- select “Subnets”
- select “Acumos-OAM-vnet”
- select “Network security group”
- select the NSG “E6E-NSG”
- select “Save”

Deploying Your Model

1. Locate the Model Detail Page for the model of interest
2. Click on the **Deploy to Cloud** drop-down arrow and select Microsoft Azure

The resulting dialog will require the parameters listed under *Configuring Your Azure Account* in this guide.

1. **Application ID** The ID for application during registrations in Azure Active Directory
2. **TenantID** The ID of the AAD (Azure Active Directory) in which application is created
3. **Secret key** Client Secret key for a web application registered with Azure Active Directory
4. **Subscription Key** Subscription grants access to Azure services and to the Azure Platform Management Portal
5. **Resource Group** Resource groups provide a way to monitor, control access, provision and manage billing for collections of assets that are required to run an application, or used by a client or company department
6. **Acr Name** Same as ApplicationID
7. **Storage Account** An Azure storage account provides a unique namespace to store and access Azure Storage data objects. All objects in a storage account are billed together as a group

The image shows two side-by-side screenshots of the 'Deploying to Microsoft Azure' dialog box. The left screenshot shows the 'Deploy' button and the 'Data Broker' button. The right screenshot shows the 'Deploy' button and the 'Data Broker' button, with the 'Data Broker' button highlighted.

The dialog box contains the following fields:

- Application Id *
- Tenant Id *
- Secret key *
- Subscription Key *
- Resource Group *
- Acr Name *
- Storage Account *

Below these fields is a 'Deploy' button. Under the 'BROKER' section, there is a 'Databroker Type' dropdown menu, a 'Zip' dropdown menu, and a 'URL' field. Below these are 'Position Mappings' and 'Field Mappings' sections, each with two rows of input fields. At the bottom are 'Cancel' and 'Data Broker' buttons.

Click **Deploy**. The Acumos platform will create these resources under your Azure subscription:

- a NIC
- a public IP address
- a disk
- a VM

At the current time, there is no explicit notification that deployment was complete and successful. You can verify deployment success as described in the following section.

Accessing and Verifying the Deployment

The Acumos platform currently creates a single user account on the deployed VM, with these credentials:

- username: dockerUser
- password: 12NewPA\$w0rd!

Cleaning up Azure Resources

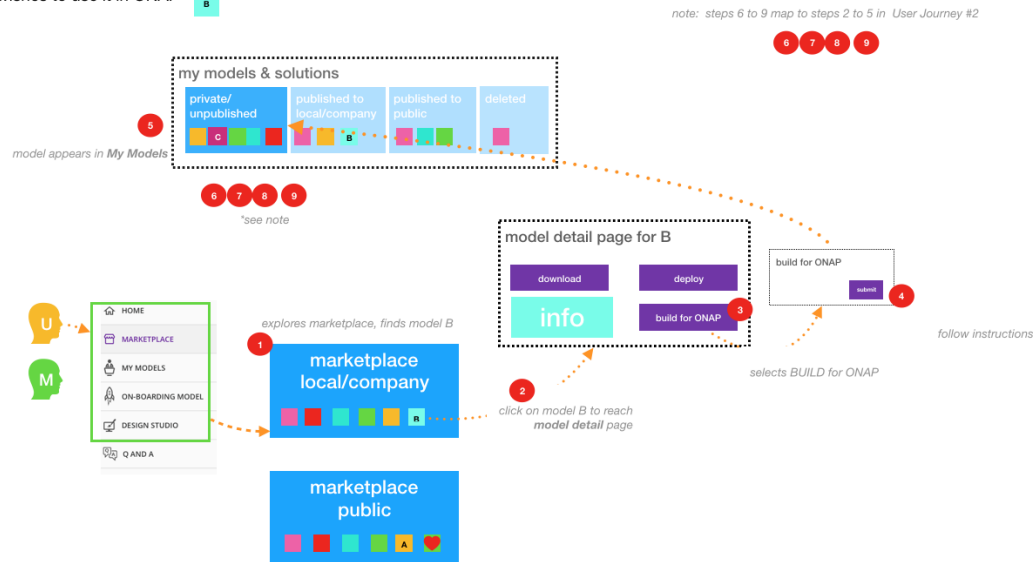
The Acumos platform does not automatically remove resources that it creates under your Azure account. When you are done testing with the model in the launched VM, if you do not want to keep these resources active, you can delete them through the Azure “All resources” list.

Converting a Model to be Used in ONAP

An overview of the user journey to discover and convert a model for use in ONAP is shown below:

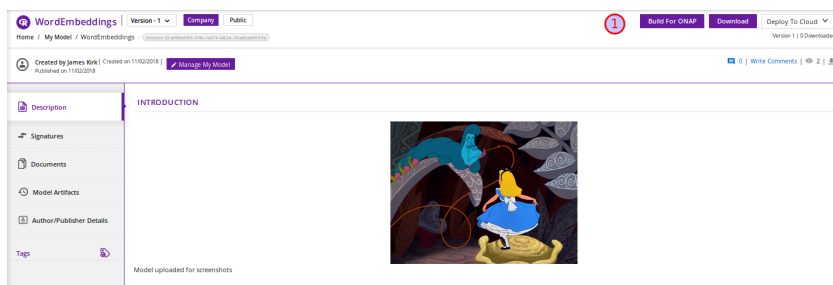
journey #8: build for ONAP

USER user journey:
User finds Model B in the marketplace
& wishes to use it in ONAP



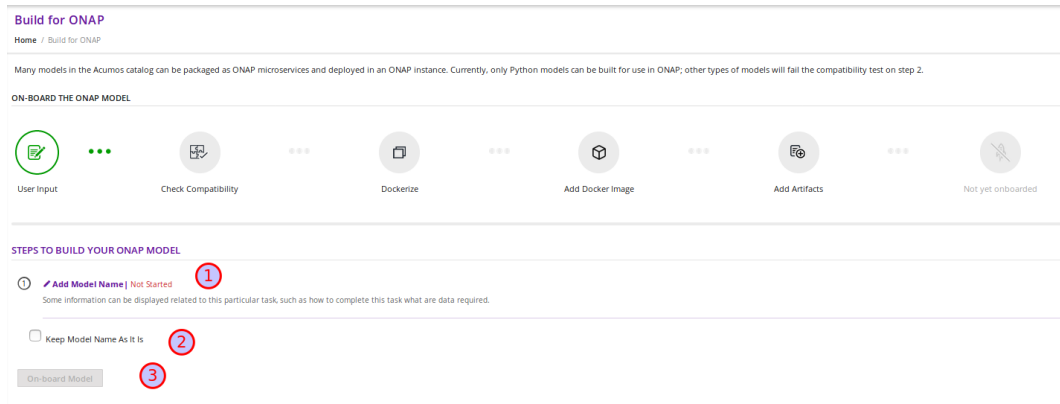
The conversion of a model for ONAP can be accomplished by the following steps:

If the model is compatible with ONAP, **Build for ONAP** button will be displayed on the **Model Details** page. Select **Build for ONAP** - this takes you to a new page.



1. Select **Add Model Name** to give the converted model a different name
2. OR you can select the **Keep Model Name As It Is** checkbox to keep the model's current name; note: if you select to keep the model's name, both **Add Model Name** and **Keep Model Name As It Is** are disabled
3. Once you have performed steps 1 or 2, select **On-Board Model**. This will initiate a series of back end process steps, illuminated as they run by the cascade from left to right of the icons on the top of the page. Once

the process is completed you can access your model in the **MY UNPUBLISHED MODELS** section of your personal catalog



2.1.4 The Portal - For Modelers

Overview

The Acumos Portal is designed to enable Modelers to easily on-board, document, and package their AI models into reusable microservices.

A Modeler may test out the Acumos features in a personal private/unpublished section of the Marketplace. Additionally, a Modeler may publish the models to the Company Marketplace or to the Public Marketplace for wider distribution.

Modelers are typically subject-matter experts in their fields, so Acumos models come from a wide range of domains and applications.

Models may be written in a number of popular programming languages or toolkits, including TensorFlow and R.

All of the models that a user has on-boarded can be viewed from the [My Models](#) page. Depending on their history, the models may exist in one for four sections: MY UNPUBLISHED MODELS, MY MODELS: PUBLISHED TO COMPANY MARKETPLACE, MY MODELS: PUBLISHED TO PUBLIC MARKETPLACE, and MY DELETED MODELS .

Models published to Company are visible only to account holders on your local Acumos instance. This can be thought of as “inside the instance firewall” – typically viewable by close collaborators. Models published to Public are available to outside Acumos instances. The set of peers that may have access to Public models is determined by your local Administrator.

Private/Unpublished models are visible only to the Modeler. However, a Modeler does have the option to share a model with a specific user who has an account on the same Acumos instance.

Model On-Boarding Overview

Acumos accommodates the use of a wide range of tools and technologies in the development of machine learning models, including support for both open source and proprietary toolkits. Models can be easily onboarded and wrapped into containerized microservices which are interoperable with many other components. On-boarding provides an ingestion interface for various types of models to enter the Acumos Machine Learning (ML) platform. Examples of models include well-defined objects such as scikit-learn estimators, TensorFlow weights, and arbitrary R functions.

The solution for accommodating a myriad of different model types is to provide a custom wrapping library for each runtime. The client library encapsulates the complexity surrounding the serialization and deserialization of models.

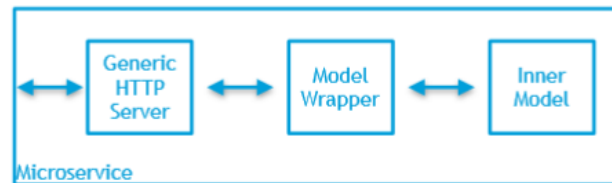
Additionally, the client library creates a common native interface, a wrapper, for invoking the inner model. In order for Acumos to be able to reason about models uniformly, there is a common model interface description that details what the available model methods are and what they look like. Acumos instantiates ML models as microservices and safely composes them together.

Architecture

High-level flow:



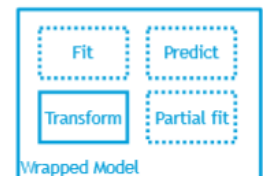
Example model wrapper



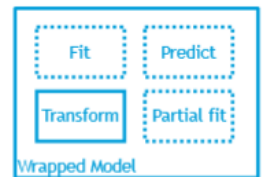
Generated microservice using wrapper

In the illustrations below, custom transformation functions which consume and produce a native DataFrame are converted to standardized native models. The transforms are then composed together in Acumos as microservices. This illustration begs the question of how the DataFrame can be represented abstractly in order to validate this workflow.

```
import pandas as pd
def py_func(df: pd.DataFrame) -> pd.DataFrame:
    ...
```



```
import com.foobar.DataFrame;
public DataFrame javaFunc(DataFrame df) {
    ...
}
```



	name	age	state	num children	num pets
0	john	33	swiss	2	0
1	mary	78	dc	2	4
2	peter	22	california	0	0
3	jeff	19	texas	1	5
4	bill	45	washington	2	0
5	lisa	33	dc	1	0

Example DataFrame



Methods and Semantics

Acumos is a machine learning platform, thus we need to provide certain “methods” in our wrapped models that Acumos can invoke in order to support various workflows. In a machine learning setting, these methods might look like:

- `fit(message) -> model state`
 - Does a full “batch” fit, replacing previous internal model parameters
 - Returns a “model state” object that provides a standard serialization method
- `partial_fit(message) -> model state`
 - Does a partial fit, updating internal model parameters
 - Returns a “model state” object that provides a standard serialization method
- `transform(message) -> message`
 - Returns an object that provides a standard serialization method

On-Boarding Client Libraries

The Acumos on-boarding process generates everything needed to create an executable microservice for your model and add it to the catalog. Acumos uses Protobuf as a language-agnostic data format to provide a common description of the model data inputs and outputs.

Acumos supports on-boarding Python, Java, and R models. The appropriate client library does the first step of the on-boarding process. This includes:

1. Introspection to assess the toolkit library versions and determine file types
2. Creation of a JSON description of the system
3. Creation of the protobuf file
4. File push to the Acumos on-boarding server

On-Boarding H2o.ai and Generic Java Models

The Acumos Java Client Library command line utility is used to on-board H2o.ai and Generic Java models. This library creates artifacts from an H2o or Generic Java model and pushes the artifacts to the on-boarding server for the H2o Model runner to be able to use them.

High-Level Flow

1. The Modeler creates a model in H2o and exports it in the MOJO model format (.zip file) using any interface (eg.Python, Flow, R) provided by H2o. For Generic Java, the Modeler creates a model and exports it in the .jar format.
2. The Modeler runs the JavaClient jar, which creates a Protobuf (default.proto) file for the Model, creates the required metadata.json file and an artifact called modelpackage.zip.
3. Depending on the choice of the Modeler, she can manually upload these generated artifacts to the Acumos Marketplace via its Web interface. This is Web-based on-boarding. We will see how to do this in this article.
4. Or the Java client library itself, on-boards the model onto the on-boarding server if the modeler provides the on-boarding server URL. This is CLI-based on-boarding.

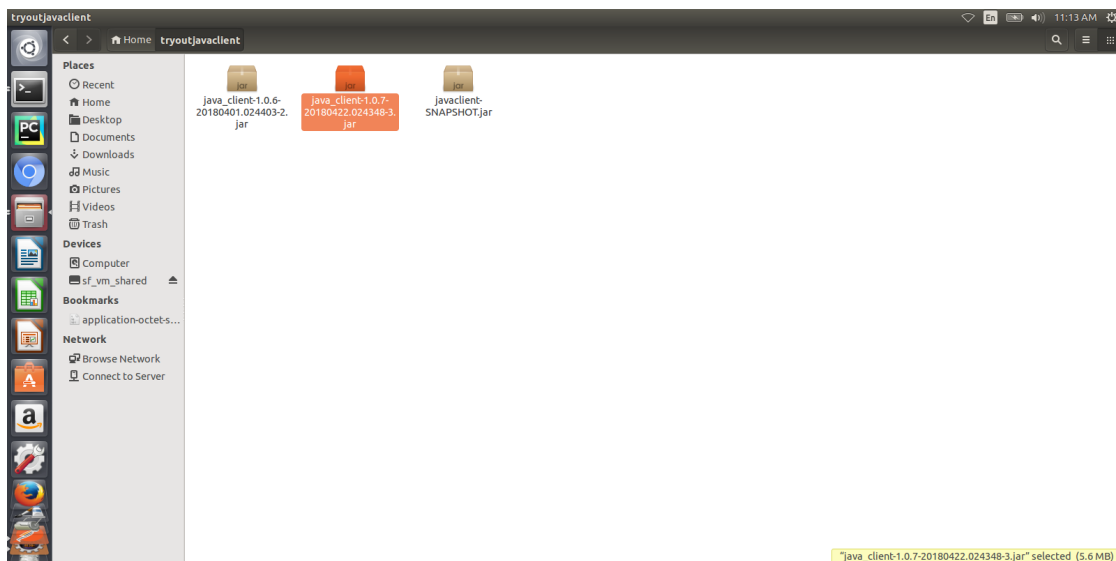
The Model Runner provides a wrapper around the ML model, packages it as a containerized microservice and exposes a predict method as a REST endpoint. When the model is onboarded and deployed, this method (REST endpoint) can then be called by other external applications to request predictions off of the model.

Prerequisites

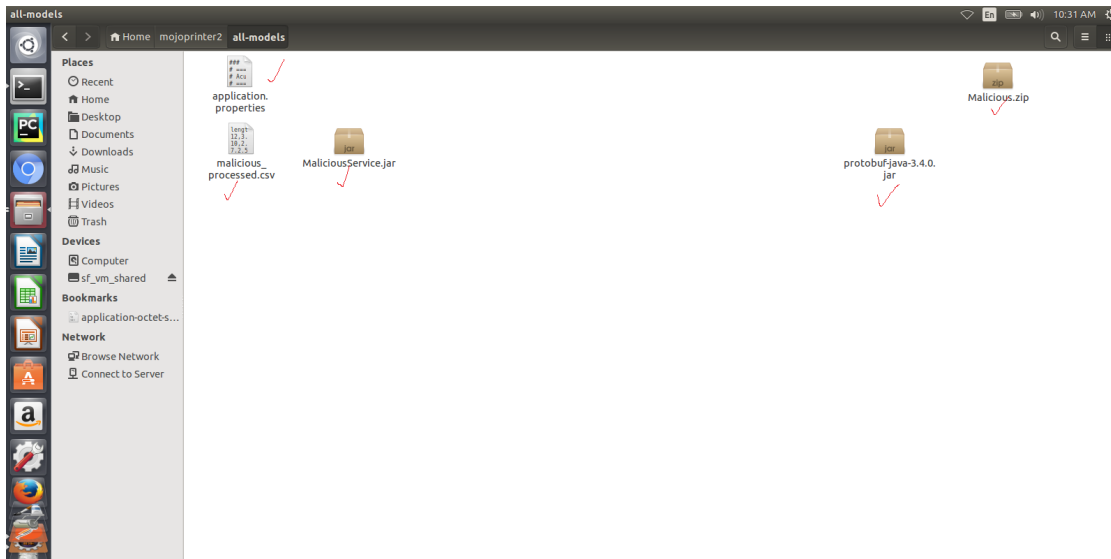
- Java 1.8
- The following Released components:
 - [Java Client v1.11.0](#) (java_client-1.11.0.jar)
 - [Generic Model Runner v2.2.3](#) (h2o-genericjava-modelrunner-2.2.3.jar)

Preparing to On-Board your H2o or a Generic Java Model

1. Place Java Client jar in one folder locally. This is the folder from which you intend to run the jar. After the jar runs, the created artifacts will also be available in this folder. You will use some of these artifacts if you are doing Web-based onboarding. We will see this later. Note: the versions of the libraries in the screenshots may be outdated.



2. Prepare a supporting folder with the following contents. Items of this folder will be used as input for the java client jar.



It will contain:

1. Models - In case of H2o, your model will be a MOJO zip file. In case of Generic Java, the model will be a jar file.
2. Model runner or Service jar - For H2O rename downloaded h2o-genericjava-modelrunner.jar as per the first section to H2OModelService.jar or to GenericModelService.jar for Java model and Place it in this folder.
3. CSV file used for training the model - Place the csv file (with header having the same column names used for training but without the quotes (“ ”)) you used for training the model here. This is used for autogenerating the .proto file. If you don't have the .proto file, you will have to supply the .proto file yourself in the supporting folder. Make sure you name it default.proto.
4. default.proto - This is only needed If you don't have sample csv data for training, then you will have to provide the proto file yourself. In this case, Java Client cannot autogenerate the .proto file. You will have to supply the .proto file yourself in the supporting folder. Make sure you name it default.proto Also make sure, the default.proto file for the model is in the following format. You need to appropriately replace the data and datatypes under DataFrameRow and Prediction according to your model.

```
syntax = "proto3";
option java_package = "com.google.protobuf";
option java_outer_classname = "DatasetProto";

message DataFrameRow {
  string sepal_len = 1;
  string sepal_wid = 2;
  string petal_len = 3;
  string petal_wid = 4;
}
message DataFrame {
  repeated DataFrameRow rows = 1;
}
message Prediction {
  repeated string prediction= 1;
}

service Model {
  rpc transform (DataFrame) returns (Prediction);
}
```

5. application.properties file - Mention the port number on which the service exposed by the model will finally run on.

```
server.contextPath=/modelrunner
# IF WORKING WITH MODEL CONNECTOR AND COMPOSITE SOLUTION, THE #server.
↪contextPath will be /
# NOTE: THIS WILL TAKE AWAY SWAGGER
# This is the port number you want to run the service on. User may select.
↪a convenient port.
server.port=8336

spring.http.multipart.max-file-size=100MB
spring.http.multipart.max-request-size=100MB

# Linux version

# if model_type is Generic Java, then default_model will be /models/model.
↪jar
# if model_type is H2o, then the default_model will be /models/Model.zip

#default_model=/models/model.jar
default_model=/models/Model.zip

default_protobuf=/models/default.proto

logging.file = ./logs/modelrunner.log

# The value of model_type can be H or G
# if model is Generic java model, then model_type is G.
# if model is H2o model, then model_type is H. And the /predict method.
↪will use H2O model; otherwise, it will use generic Model
# if model_type is not present, then the default is H

#model_type=G
model_type=H
model_config=/models/modelConfig.properties

# Linux some properties are specific to java generic models

# The plugin_root path has to be outside of ModelRunner root or the code.
↪won't work
# Default proto java file, classes and jar
# DatasetProto.java will be in $plugin_root\src
# DatasetProto$.classes will be in $plugin_root\classes
# pbuff.jar will be in $plugin_root\classes

plugin_root=/tmp/plugins
```

6. modelConfig.properties - Add this file only in case of Generic Java model onboarding. This file contains the modelMethod and modelClassName of the model.

```
modelClassName=org.acumos.ml.XModel
modelMethod=predict
```

Create your modeldump.zip file

Java Client jar is the executable client jar file.

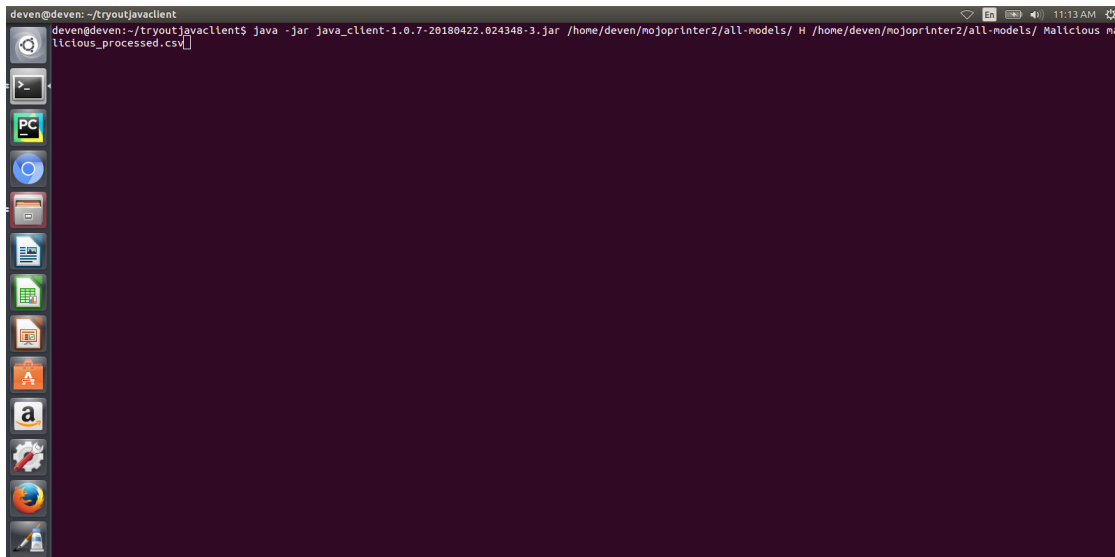
For Web-based onboarding of H2o models, the parameters to run the client jar are:

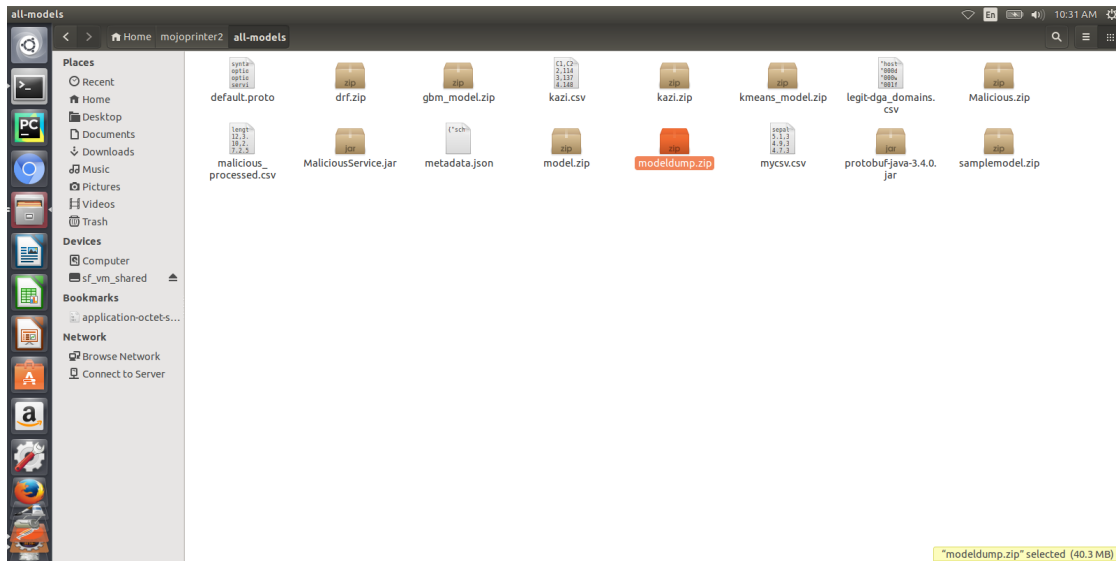
1. Current Folder path : Full folder path in which Java client jar is placed and run from
2. Model Type : H for H2o, G for Generic Java
3. Supporting folder path : Full Folder path of the supporting folder which contains items.
4. Name of the model : For h2o just the name of the model without the .zip extension. Make sure this matches name of the supplied MOJO model file exactly.
5. Input csv file : csv file that was used for training the model. Include the .csv extension in the csv file name. This will be used to autogenerate the default.proto file. This parameter will be empty if you yourself have supplied a default.proto for your model.

For CLI-based onboarding, the parameters to run the client jar are:

1. Onboarding server url.
2. Pass the authentication API url for onboarding - This API returns jwtToken for authenticated users. e.g <http://<hostname>:8090/onboarding-app/v2/auth>
3. Model Type : H for H2o, G for Generic Java.
4. Supporting folder path : Full Folder path of the supporting folder which contains items.
5. Name of the model : For h2o just the name of the model without the .zip extension. Make sure this matches name of the supplied MOJO model file exactly.
6. Username of the Portal MarketPlace account.
7. Password of the Portal MarketPlace account.
8. Input csv file : csv file that was used for training the model. Include the .csv extension in the csv file name. This will be used to autogenerate the default.proto file. This parameter will be empty if you yourself have supplied a default.proto for your model.

See example below for how to run the client jar and how the modeldump.zip artifact appears after its successful run:





Onboarding to the Acumos Portal

- If you used CLI-based onboarding, you don't need to perform the steps outlined just below. The Java client has done it for you. You will see a message on the terminal that states the model onboarded successfully.
- If you use Web-based onboarding, you must complete the following steps:
 1. After you run the client, you will see a modeldump.zip file generated in the same folder where we ran the Java Client for.
 2. Upload this file in the Web based interface (drag and drop). See *On-Boarding a Model Using the Portal UI*
 3. You will be able to see a success message in the Web interface. you will be able to see a success method in the Web interface.

The needed TOSCA artifacts and docker images are produced when the model is onboarded to the Portal. You and your teammates can now see, rate, review, comment, collaborate on your model in the Acumos marketplace. When requested and deployed by a user, your model runs as a dockerized microservice on the infrastructure of your choice and exposes a predict method as a REST endpoint. This method can be called by other external applications to request predictions off of your model.

Addendum : Creating a model in H2o

You must have H2o 3.14.0.2 installed on your machine. For instructions on how to install visit the H2o web site: <https://www.h2o.ai/download/>.

H2o provides different interfaces to create models and use H2o for eg. Python, Flow GUI, R, etc. As an example, below we show how to create a model using the Python innterface of H2o and also using the H2o Flow GUI. You can use the other interfaces too which have comparable functions to train a model and download the model in a MOJO format.

Here is a sample H2o iris program that shows how a model can be created and downloaded as a MOJO using the Python interface:

```
import h2o
import pandas as pd
import numpy as np
```

```
import matplotlib.pyplot as plt
import seaborn as sns

# for jupyter notebook plotting,
%matplotlib inline
sns.set_context("notebook")

h2o.init()

# Load data from CSV
iris = h2o.import_file('https://raw.githubusercontent.com/h2oai/h2o-3/master/h2o-r/
↳h2o-package/inst/extdata/iris_wheader.csv')

Iris data set description
-----
1. sepal length in cm
2. sepal width in cm
3. petal length in cm
4. petal width in cm
5. class:
    Iris Setosa
    Iris Versicolour
    Iris Virginica

iris.head()
iris.describe()
# training parameters
training_columns = ['sepal_len', 'sepal_wid', 'petal_len', 'petal_wid']
# response parameter
response_column = 'class'

# Split data into train and testing
train, test = iris.split_frame(ratios=[0.8])
train.describe()
test.describe()

from h2o.estimators import H2ORandomForestEstimator
model = H2ORandomForestEstimator(ntrees=50, max_depth=20, nfolds=10)

# Train model
model.train(x=training_columns, y=response_column, training_frame=train)

print (model)

# Model performance
performance = model.model_performance(test_data=test)
print (performance)

# Download the model in MOJO format. Also download the h2o-genmodel.jar file
modelfile = model.download_mojo(path="/home/deven/Desktop/", get_genmodel_jar=True)

predictions=model.predict(test)
predictions
```

Here is a sample H2o iris example program that shows how a model can be created and downloaded as a MOJO using the H2o Flow GUI.

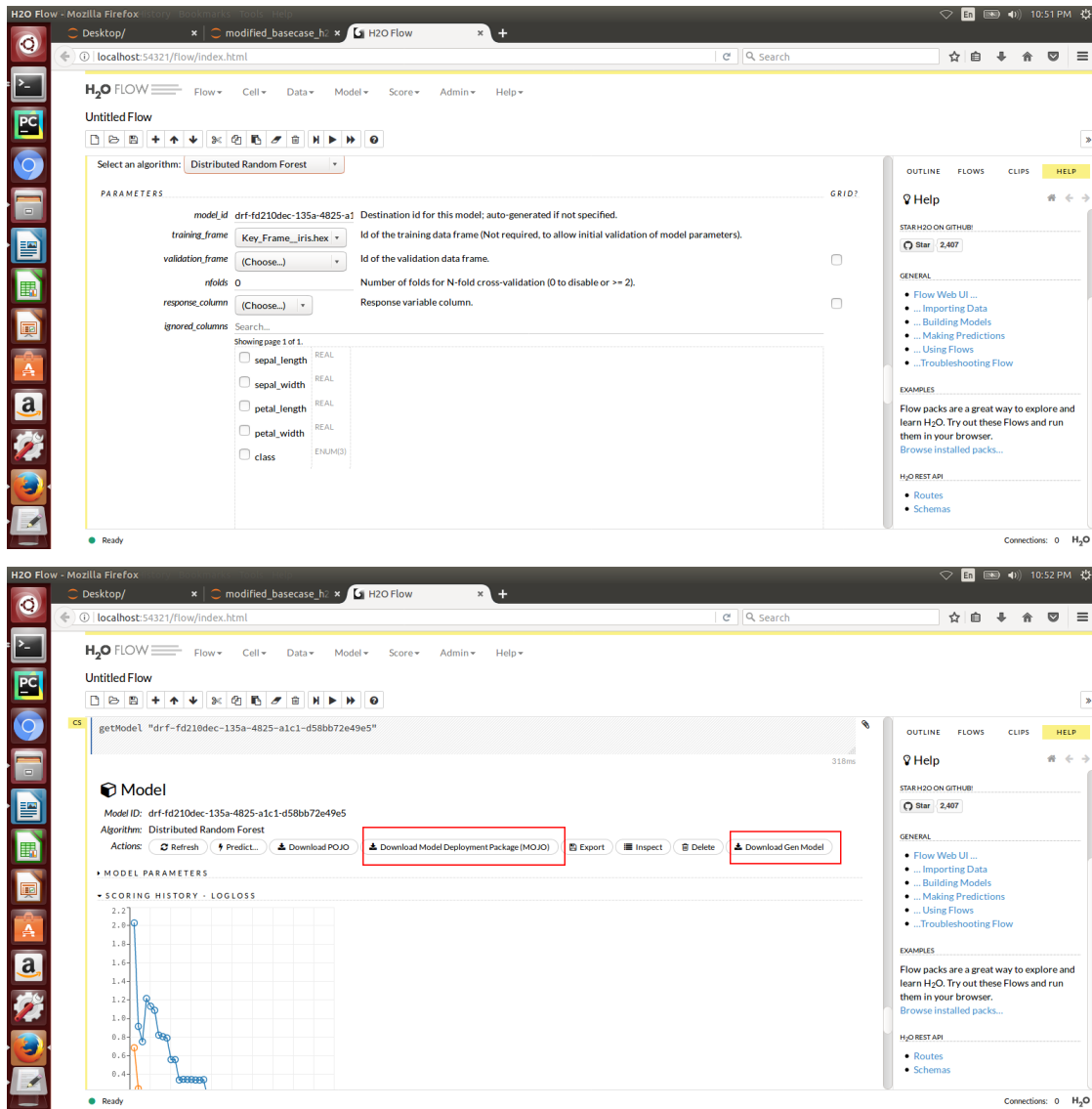
The first screenshot shows the H2O Flow web interface with the 'Data' menu open, highlighting the 'Import Files...' option. The interface includes a top navigation bar with 'Flow', 'Cell', 'Data', 'Model', 'Score', 'Admin', and 'Help' menus. The main area displays an 'Untitled Flow' with a toolbar and a list of routines on the left. The right sidebar contains a 'Help' section with links to 'STAR H2O ON GITHUB!', 'GENERAL', and 'EXAMPLES'.

The second screenshot shows the 'Import Files' routine configuration. The 'source_frames' field is set to '["iris.csv"]', and the 'destination_frame' is 'Key_Frame__iris.hex'. The 'parse_type' is 'CSV', and the 'separator' is '44'. The 'number_columns' is '5', and the 'single_quotes' is 'false'. The 'column_names' are '["sepal_length", "sepal_width", "petal_length", "petal_width", "class"]', and the 'column_types' are '["Numeric", "Numeric", "Numeric", "Numeric", "Enum"]'. The 'delete_on_done' is 'true', the 'check_header' is 'true', and the 'chunk_size' is '4194384'. The 'Job' section shows a 'Run Time' of '00:00:00.271' and a 'Remaining Time' of '00:00:00.0'. The 'Type' is 'Frame', and the 'Description' is 'Parse'. The 'Status' is 'DONE', and the 'Progress' is '100%'. The 'Done' button is visible.

The third screenshot shows the 'Key_Frame__iris.hex' summary page. The 'Actions' section includes 'View Data', 'Split...', 'Build Model...', 'Predict', 'Download', and 'Export'. The 'Rows' are '150', the 'Columns' are '5', and the 'Compressed Size' is '2KB'. The 'COLUMN SUMMARY' table is displayed below:

Label	type	Missing	Zeros	-Inf	Inf	min	max	mean	sigma	cardinality	Actions
sepal_length	real	0	0	0	0	4.3008	7.9999	5.8433	0.8281	-	-
sepal_width	real	0	0	0	0	2.0	4.4000	3.8540	0.4336	-	-
petal_length	real	0	0	0	0	1.0	6.3000	3.7587	1.7644	-	-
petal_width	real	0	0	0	0	0.1000	2.5000	1.1987	0.7632	-	-
class	enum	0	58	0	0	0	2.0	-	-	3	Convert to numeric

The 'CHUNK COMPRESSION SUMMARY' and 'FRAME DISTRIBUTION SUMMARY' sections are also visible at the bottom of the page.



On-Boarding a Python Model

Note: Python Client v0.7.0 was tested with the Acumos Athena platform release

The Acumos Python Client library is required for users who want to push their TensorFlow and scikit-learn models to the Acumos Portal. You will use this library for on-boarding all your models. The library creates meta-data by introspection, packages all the necessary information, and then uploads it to Acumos On-Boarding server. The Acumos Python Client library is packaged and available on PyPI. Please see the [PyPI](#) page for instructions and a tutorial.

To better know how to on-board a model using the Web interface, please have a look at the [Web Onboarding page](#)

On-Boarding an R Model

Note: R Client v0.2-7 was tested with the Acumos Athena platform release

Prerequisites

Before you begin:

1. You must have the following packages installed in your system : protobuf-compiler, protobuf-c-compiler, libprotobuf-c-dev, libprotobuf-dev, libprotoc-dev
2. You must have an Acumos account
3. You must have protobuf 3 installed on your system (version 2 will not work).

```
git clone https://github.com/google/protobuf.git protobuf
cd protobuf
./autogen.sh
./configure --prefix=`pwd`/../../`uname -m`-linux-gnu
make
make install
```

4. You must have R installed on you system. Please have a look at cran.r-project.org

Installing the Acumos R Client

Within R you need to install and load all dependent packages from CRAN first.

```
install.packages(c("Rcpp", "RCurl", "RUnit", "rmarkdown", "knitr", "pinp", "xml2"))
library(Rcpp, RCurl, RUnit, rmarkdown, knitr, pinp)
```

Then Install the Acumos R Client package and RProtobuf package thanks to the following command:

```
install.packages("RProtoBuf")
install.packages("acumos", , c("http://r.research.att.com", "http://rforge.net"))
```

Alternatively, to install from sources:

```
git clone git@github.com:s-u/acumos.git or git clone https://github.com/s-u/acumos.git
R CMD build acumos
R CMD INSTALL acumos_*.tar.gz
```

Using the Acumos R Client

Model bundle

To create the model bundle, use `acumos::compose()` with the functions to expose. If type specs are not defined, they default to `c(x="character")`. The model bundle (named `component.amc`) is in fact a zip file that contains three distinct files :

1. `meta.json` defining the component and their metadata,
2. `component.bin` the binary payload,
3. and `component.proto` with the protobuf specs.

Please consult R documentation page for details, i.e., use `?compose` in R or see the [Compose](#) page at RForge.

If you used R under windows you could meet an issue using the `acumos::compose()` function due to some problems between R under windows and zip. If RTools is not installed on your windows environment, the model bundle will not be created. So please follows the installation procedure of

[Rtools](#)

Authentication and upload

Once the model bundle is created, you can use the `push()` API to upload it in Acumos.

```
acumos::push("https://url", "file", "username:token")
```

url can be found in the ON-BOARDING MODEL page of your Acumos portal and looks like :
“hostname:port/onboarding-app/v2/models”

file : component.zip

username : your Acumos username

token : Authentication token available in the Acumos portal in your profile section

You can also authenticate yourself by using the `auth()` API:

```
acumos::auth("url", "username", "password")
```

url can be found in the ON-BOARDING MODEL page of your Acumos portal and looks like
“hostname:port/onboarding-app/v2/auth”

username : your Acumos username

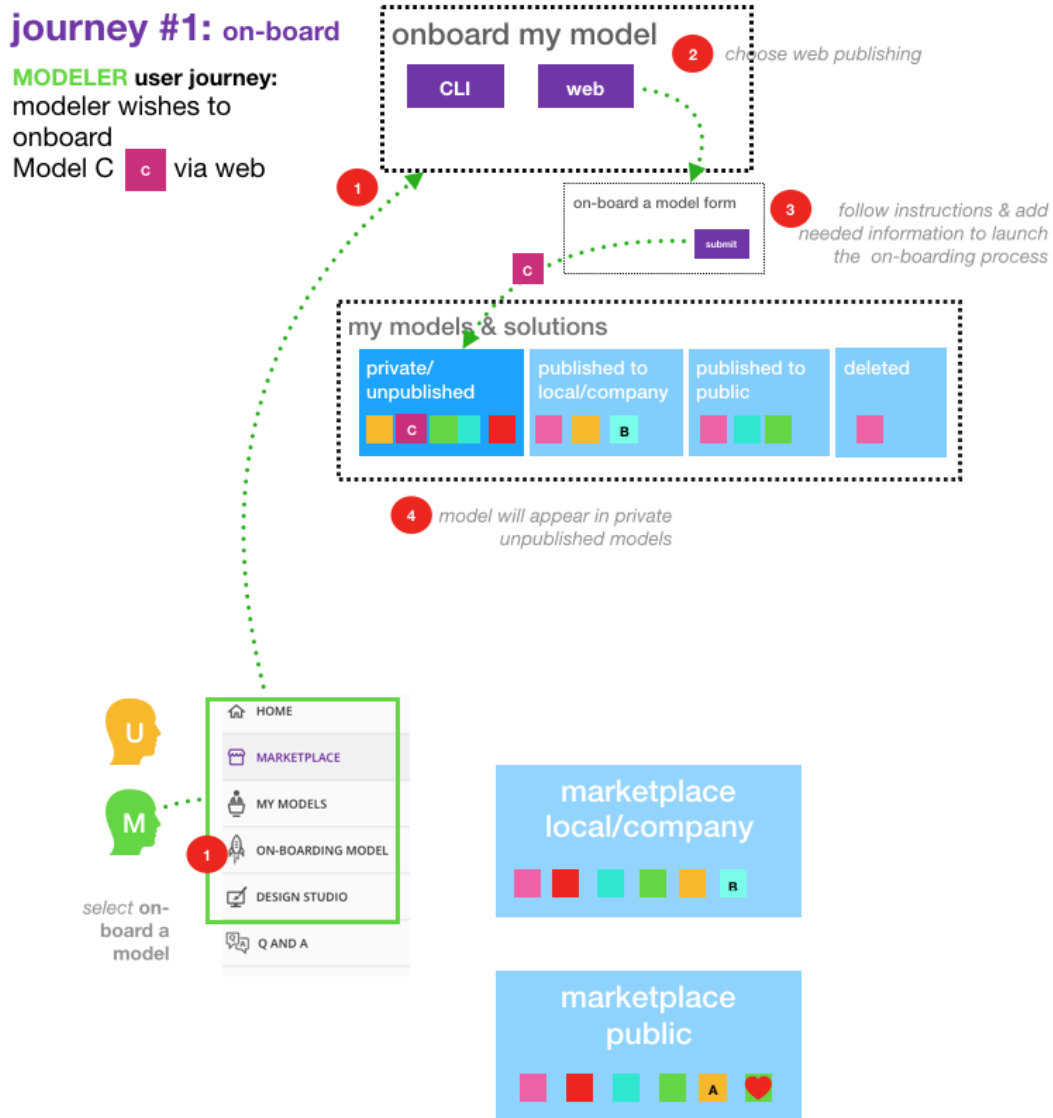
password : your Acumos password

In the Response, you will receive an authentication token to be used in the `acumos::push()` function :

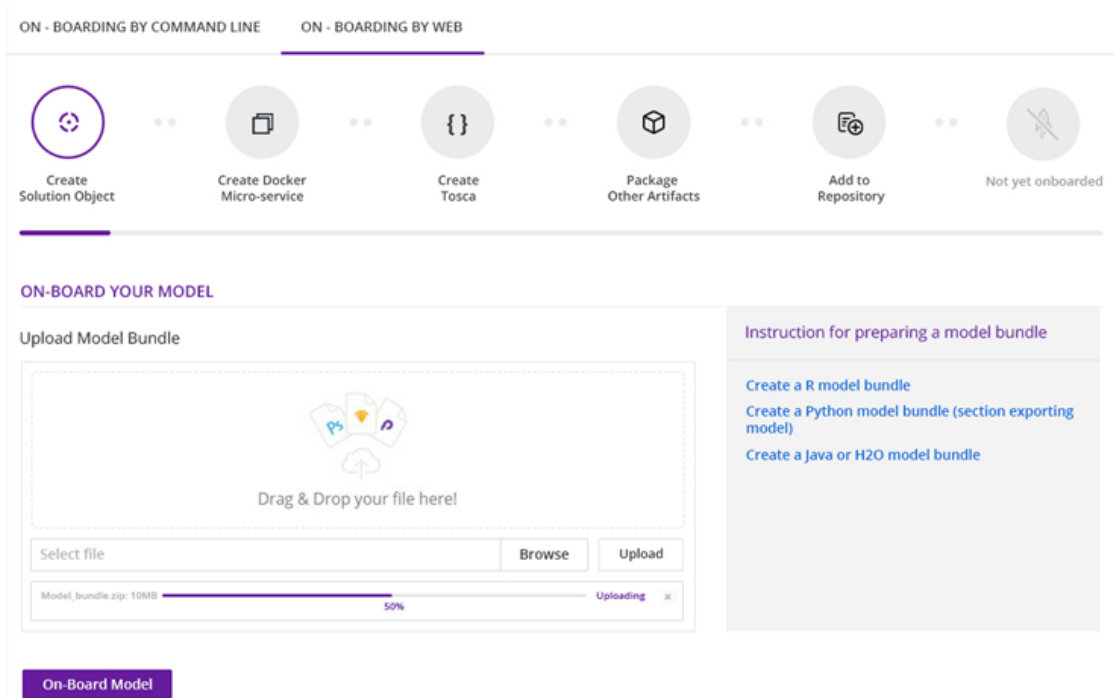
```
acumos::push("url", "file", "token")
```

On-Boarding a Model Using the Portal UI

A high-level summary of the on-boarding steps and overview of the workflow is shown below:

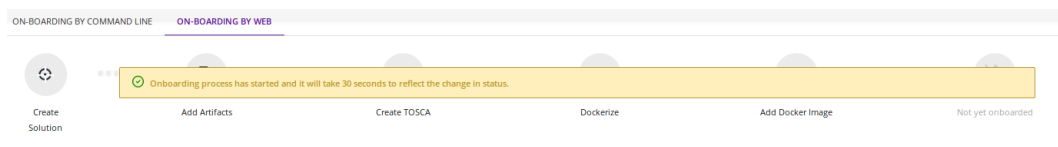


Prerequisites : You have to create a model bundle in your own environment before start to on-board. Acumos cannot transform your model in a microservice with only the model itself, it needs some other relevant information contained in the model bundle. The model bundle consists of `component.json` defining the component and its metadata, `component.bin` the binary payload and `component.proto` with the protobuf specs. You can retrieve all the information to create your model bundle in the “ON-BOARDING BY WEB” home page

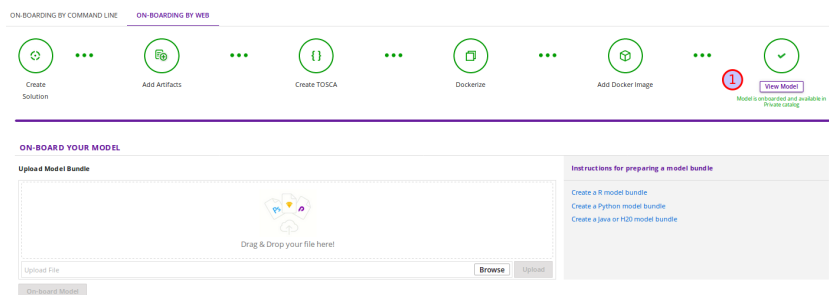


Once your model bundle has been created, follow these steps:

1. Select the “ON-BOARDING MODEL” tab in the outer left menu
2. Select “ON-BOARDING BY WEB”
3. Drag & Drop your model bundle or use the **Browse** button to select it
4. Click **Upload**
5. When uploading is finished, click on **On-Board Model**.



This will initiate a series of back-end process steps, illuminated as they run, by the cascade from left to right of the icons on the top of the page. These include creating the microservice that will run in a docker container, creating a TOSCA file for your model so it can be used in the Design Studio, and storing the artifacts and model.



On-boarding is finished when all steps turn green. Click the **View Model** button to see your model in the **MY UNPUBLISHED MODELS** section of the *My Models* page.

If one of the steps appears in red, on-boarding has failed. Check your notifications to determine why on-boarding failed - there should be a message with a link to download on-boarding log files. If you need help debugging, please reach out to the Acumos Community or Dev Discuss [mailing lists](#) or post on [StackOverflow](#).

Viewing Your Models

Users may view all the models they have uploaded by accessing the My Models page.

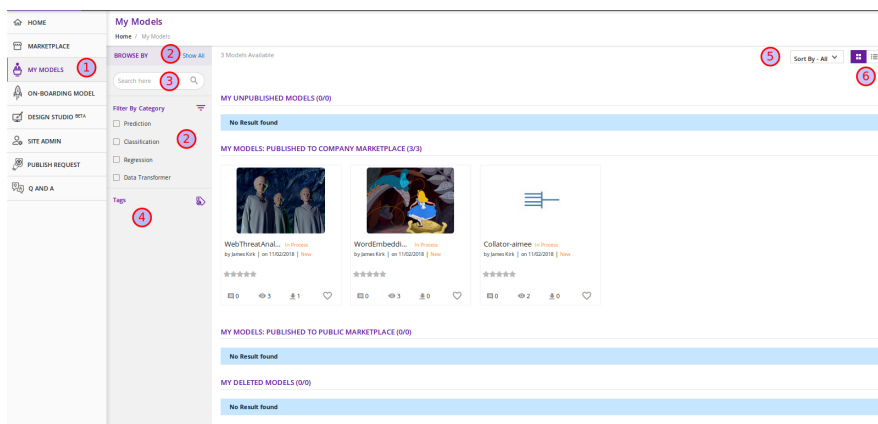
Models are organized by their visibility to others on your **My Models** page. They are sorted into the following sections:

MY UNPUBLISHED MODELS Initially, successfully on-boarded model will appear in my Models page in the UNPUBLISHED section. These are visible only to you and any collaborators of that model (shared). Partially on-boarded models (in process) are also displayed in this section but are shown greyed out until the on-boarding process is successfully completed.

MY MODELS: PUBLISHED TO COMPANY MARKETPLACE Models that have been published to Local, appear in the the LOCAL marketplace and are visible to anyone with an account on the local Acumos Instance.

MY MODELS: PUBLISHED TO PUBLIC MARKETPLACE Models that have been published to Public, appear in the the PUBLIC marketplace and may be viewed by users on Acumos instances that have a federated relationship with your local instance.

MY DELETED MODELS At this time, models are not truly deleted but rather moved into a “deleted” state.



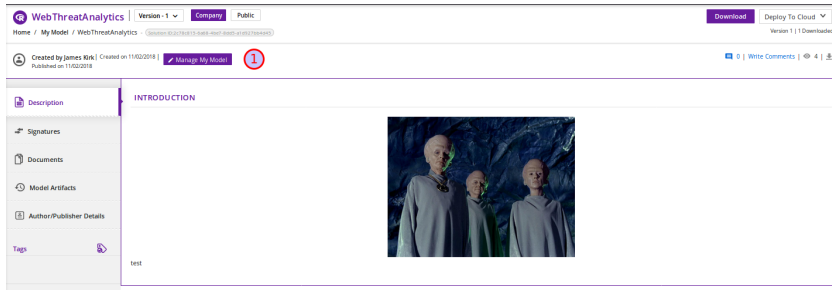
1. Access the **My Models** page from the outer left menu

2-6. This functionality works the same way it does on the main Marketplace page. Please see the [Overview](#) for details.

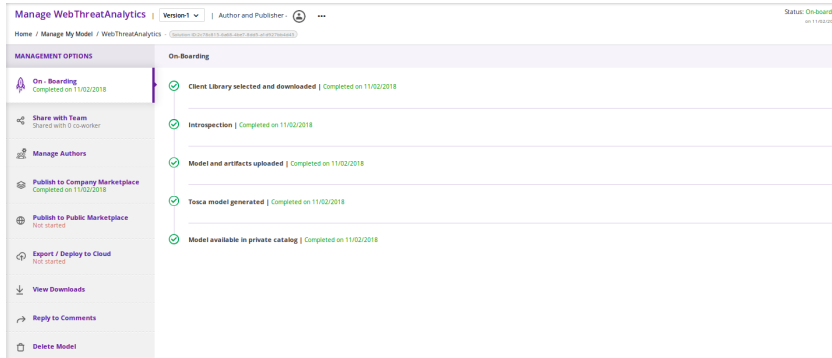
Clicking on any model image shows the **Model Detail** page for that model.

Managing a Model

The Model Detail page may not show very much information if the model has not been published. To add a description, documents and details, click on the **Manage My Model** button at the top.



A new page loads with **MANAGEMENT OPTIONS** on the left.

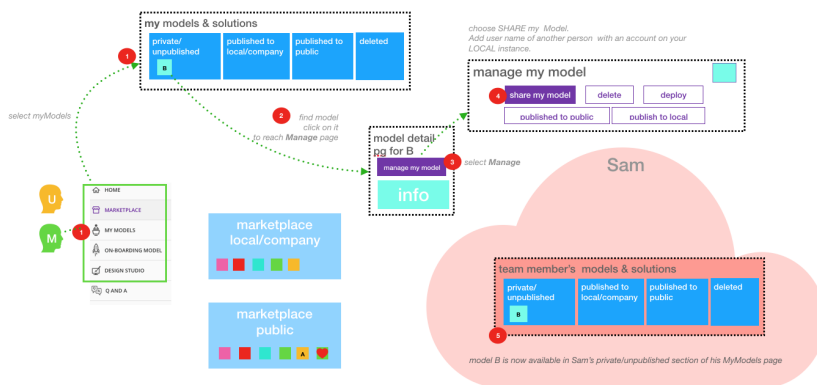


Sharing a Model with a Team

A Modeler can share a model with anyone who has an account on his/her local Acumos. When you share a model with a collaborator, you make that Modeler a co-owner of the model. This means they have all the same capabilities as the original owner. An overview is shown below.

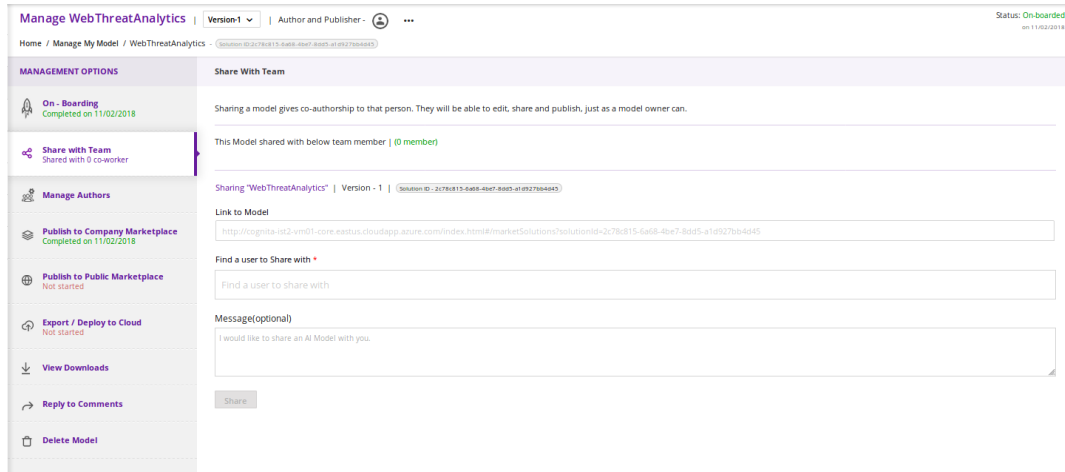
journey #2: share

MODELER user journey:
modeler wishes to share Model B with a team member, Sam.



The steps to share are as follows:

1. First, select the **Share with Team** tab under **MANAGEMENT OPTIONS**



- Next, where you see **Find a user to Share with**, type in the user ID of the person with whom you wish to share. You will need to get that user ID from them. The UI will show suggestions based on the characters you have typed. Once you have located the correct person, select the **Share** button

Share With Team

Sharing a model gives co-authorship to that |

This Model shared with below team member

Sharing "WebThreatAnalytics" | Version - 1

Link to Model

<http://cognita-ist2-vm01-core.eastus.clouda>

Find a user to Share with *

ken|



Akash Deep



Acumos Admin



Andrew Kristiansen



Emma Kristiansen




Ken Kristiansen

3. The **Share with Team** will update. You will see that your model is shared and you have added collaborators.

Share With Team

Sharing a model gives co-authorship to that person. They will

This Model shared with below team member | (1 member)



Ken Kristiansen

Sharing "webThreatAnalytics" | Version - 1 | Solution ID - 2c78c

Link to Model

<http://cognita-ist2-vm01-core.eastus.cloudapp.azure.com/in>

Find a user to Share with *

Find a user to share with

Message(optional)

WooHoo

Share

The collaborator will receive a notification that a new model has been shared with him/her.

Manage Authors

Model owners have the ability to add the details of additional authors.

The screenshot shows the 'Manage WebThreatAnalytics' interface. At the top, there's a header with 'Manage WebThreatAnalytics', 'Version 1', and 'Author and Publisher'. A status indicator on the right says 'Status: On-boarded on 11/02/2018'. Below the header, a breadcrumb trail reads 'Home / Manage My Model / WebThreatAnalytics'. The main content area is divided into two columns. The left column, titled 'MANAGEMENT OPTIONS', contains three items: 'On - Boarding' (Completed on 11/02/2018), 'Share with Team' (Shared with 1 co-worker), and 'Manage Authors' (selected). The right column, titled 'Manage Authors', has a sub-header 'EXISTING AUTHORS' and a text prompt 'Please add additional names of one or more author of this model.' Below this, there are two input fields: 'Name' (containing 'Ken Kristiansen') and 'Contact Info' (containing 'heisthemani@fakeaccount.net'). A note below the contact field says 'You can mention Email Address,URL, and Phone Number'. An 'Add author' button is at the bottom right of the right column.

After you fill in the required fields, click **Add author**.

Publishing a Model

Users may distribute their model by publishing it to either their Company Marketplace or to the Public Marketplace.

The presentation of the model may be different in each marketplace to meet the needs of the different communities. For example, a user may wish to provide company-specific details to their colleagues inside their Company instance. This may include proprietary information, documents or details that are only relevant to colleagues using the Company instance. Information published to Company is contained within the company firewall.

The Modeler may wish to present their model to the Public Marketplace in a more general way, so it can be discovered and adapted for use by others.

Acumos provides two separate publishing workflows to meet this need.

There is a facility to simply use the same information if the publication information is the same for both marketplaces. Also, publishing to either marketplace can be done in any order. There is no requirement to publish first to the Company marketplace. The same model can appear in both catalogs.

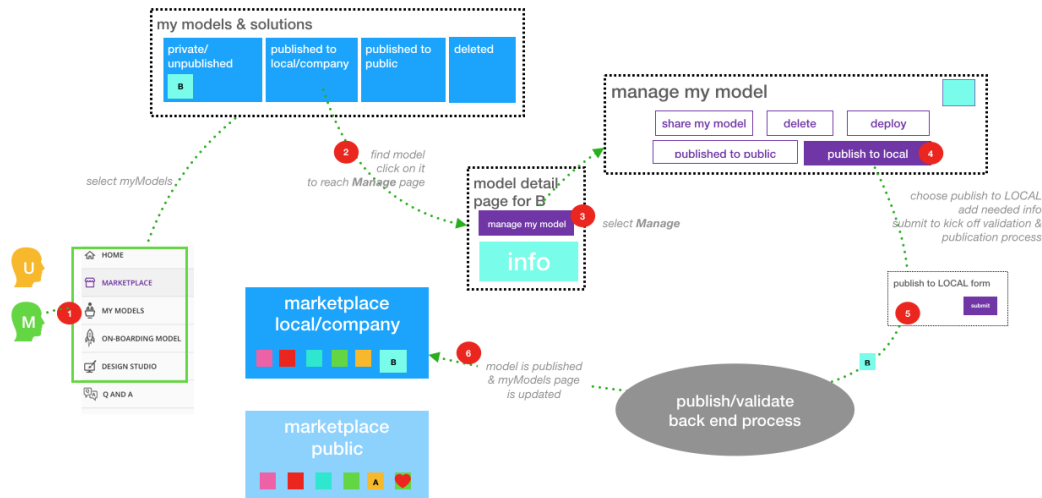
Publishing to the Company Marketplace

The publishing process is summarized here.

journey #3: publish a model to LOCAL

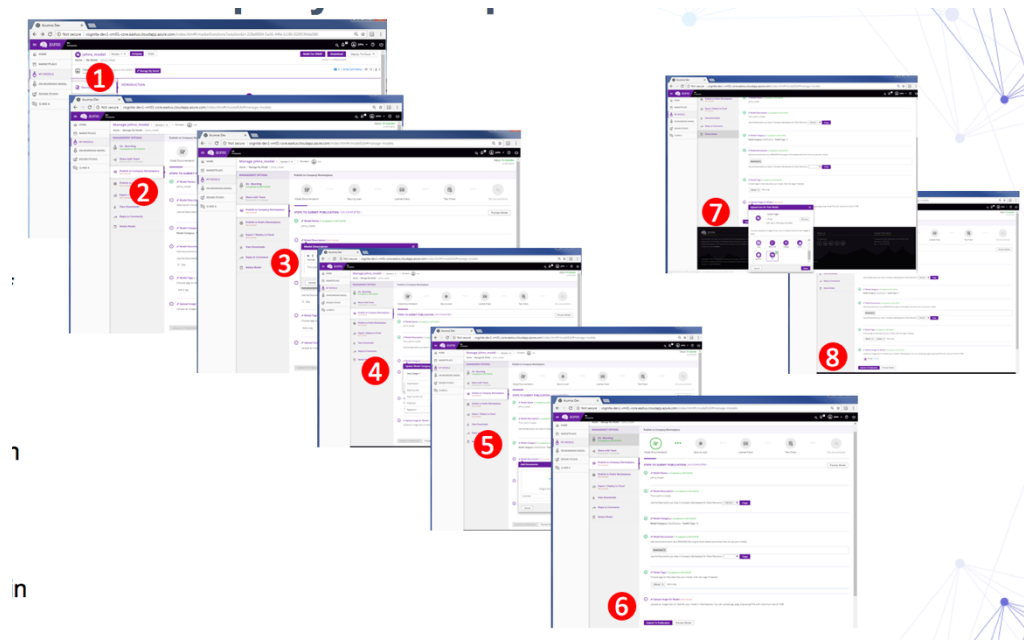
MODELER user journey:

Model B is owned by Modeler
It is in the private/unpublished models section of the **MyModels** page
He/she wished to publish it to his Local/Company Marketplace.



Specific steps:

1. From the **My Models** page, select the model of interest, open the **Model Detail Page** and click on **Manage My Model** at the top
2. Select Publish to Company Marketplace
3. Click on Model Description and describe your model in terms that your users will understand and wish to use it
4. Click on Model Category. Select a Category and Toolkit type from the dropdown box
5. Select **Model Documents** and add any useful documents, such as release notes or detailed instructions that will help your users
6. Click on **Model Tags**. Either select one of the system tags or add your own. Any tags you add will become available for other users to select as well.
7. You have completed the first step for publishing. Now click on **Submit for Publication**. This will launch a series of back end steps that will prepare your model for publication
8. The publishing workflow may consist of several steps configured by the Acumos Admin. Some instances may require manual review.
9. Once the publishing process is complete, all the workflow icons will be highlighted and the model will be available in the Company Marketplace

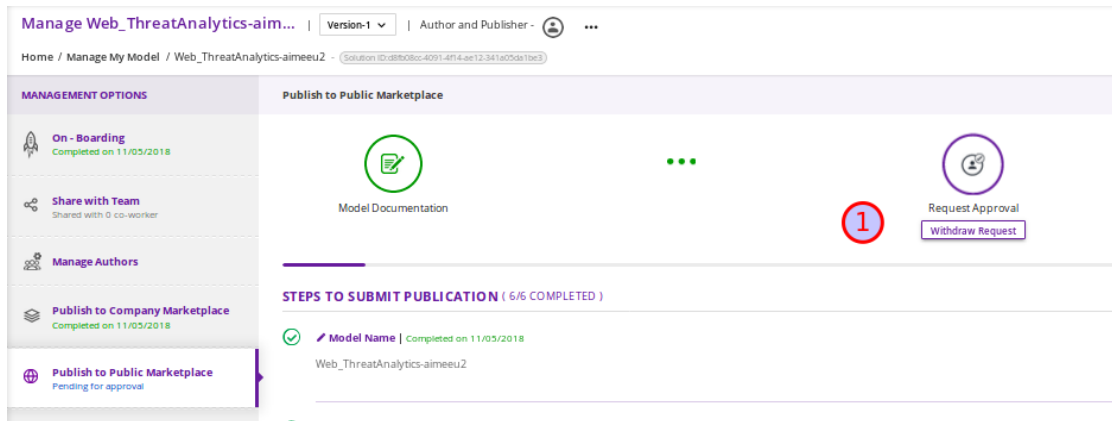


Publishing to the Public Marketplace

Publishing to the Public Marketplace follows the same steps as Publish to Company Marketplace except it requires the approval of a user who has the Publisher role.

The description and documentation may be different, as needed for a different audience. If you wish to use the same presentation for the Public model that you have previously published to Company, use the checkbox to select that.

A modeler can also withdraw a model from publication by clicking on **Withdraw Request** from the **Publish To Public Marketplace** screen.



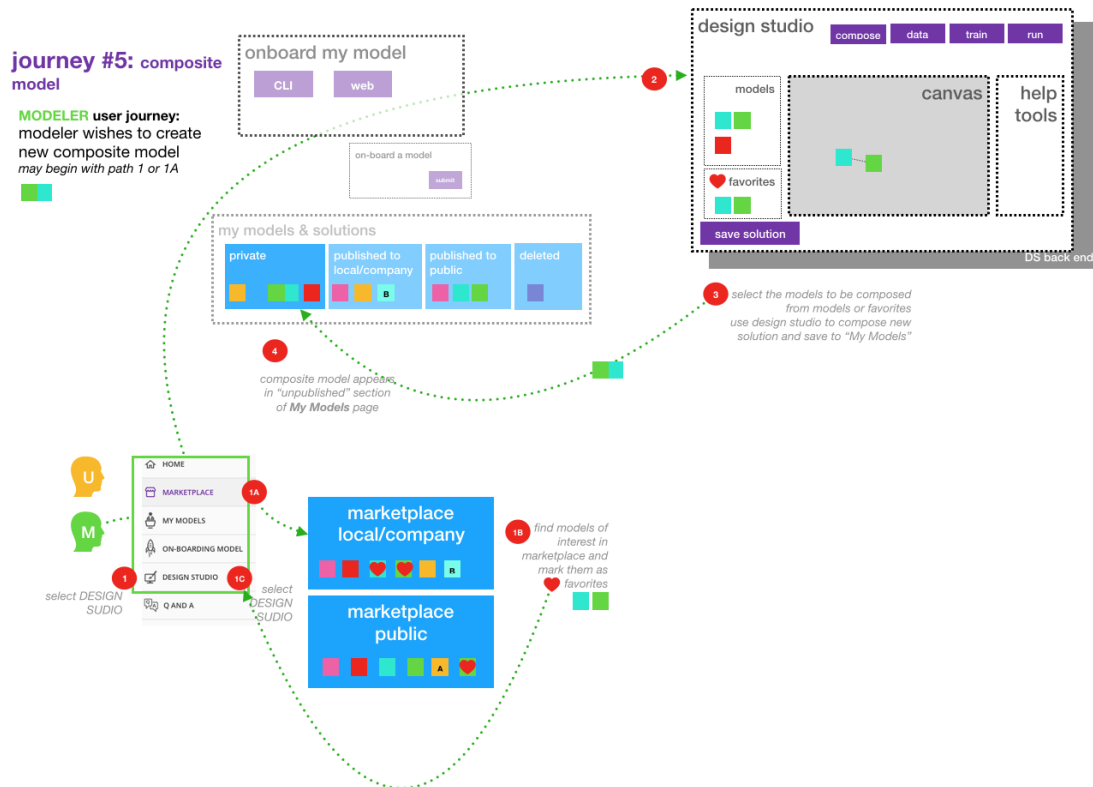
Once approved by a Publisher, the model will be published to the Public Marketplace.

2.1.5 The Design Studio: For Composers

Overview

The Design Studio is 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 runtime environment.

An overview of the user journey for the Design Studio is shown below.



Architecture

ML Models are the basic building blocks in the Design Studio. It is these models that are combined together by the designer to create complex ML application – aka composite solutions.

ML Models – Isolated and Standalone

ML Models are developed and contributed by ML subject matter experts. They may be written in any programming language and may have been developed in any toolkit – Scikit, Tensor Flow, R, H2O, etc.

The model developer may not necessarily be aware of the existence of other models. The models are usually standalone entities. They offer a standard contract – an interface definition to the external world. This contract specifies the details of the operation performed by the model, the input request (message) consumed by the model and the output response (message) produced by the model. In Acumos, this contract is specified in the Protobuf file.

ML Models – Ports, Requirements and Capabilities

Each ML Model may support one or more operations – corresponding to the functions, such as “prediction,” “classification,” etc., performed by the model. Each operation consumes an input message and produces an output message. The message signatures are specified in the Protobuf file.

Each operation is represented by two ports – an input port and an output port. An ML model may have more than two ports, if it provides (exposes) multiple operations (aka services).

1. Input Port - consumes the input message and provides the service, such as prediction or classification or regression to the caller/client. The input port represents the capability of the model. The client that need a service to be performed need to send a request to input or the capability port of the model.
2. Output Port – produces the output (response) message. Note that the output produced by an operation (say the Prediction message) need not necessarily be consumed by the caller/client, but in fact needs to be fed to another ML Model which provides another service, such as classification (of the Prediction message). So from a composition perspective, the output port represents a requirement that is satisfied by classification service.

Model Ingestion

How to Ingest ML Models in Design Studio

In order to ingest the onboarded ML Models into the Design Studio, the following steps must be performed:

1. The models along with their Protobuf files must be onboarded via the onboarding functionality, or a Protobuf file was generated when the model was onboarded
2. The Protobuf files should have both the service specification and the message specifications
3. The service specification of the Protobuf file should have the complete operation signature(s) listed in them – such as the:
 - (a) Type of the operation – rpc, etc
 - (b) Name of the operation
 - (c) Input message name
 - (d) Output message name
4. Each input and output messages should have their message signatures listed, and each field type should be basic Protobuf data type.

5. After the models have been successfully onboarded, the modeler must login to the Acumos Marketplace Portal in order to classify the uploaded model into one of model categories. Currently four categories are supported in Design Studio: Classification, Prediction, Regression and Other.
6. In order to classify the on boarded model into one of the four categories above, the modeler needs to:
 - (a) Go to the “My Models” in Market Place
 - (b) Click on the newly on boarded model
 - (c) Click on “Manage My Models”
 - (d) Click on “Publish to Company Marketplace”
 - (e) Click on “Model Category”
 - (f) Select the appropriate model category and the toolkit type
 - (g) Click Done
7. The model would now appear in the “Models” (left hand side) palette of the Design Studio under the appropriate category. The model is now available to be dragged and dropped in the Design Studio canvas.

Files Generated for Design Studio

Once the models have been onboarded, the Protobuf files associated with the model is used to generate Protbuf.json and TGIF.json files

Protobuf.json File

This is an intermediary file used to represent the Protobuf.proto file in JSON format. It is used for the generation of TGIF.json file.

TGIF.json File

The TGIF.json file represents an ML Model in the Design Studio. Every model should have a TGIF.json file associated with it to allow the model to be represented in the Design Studio, dragged and dropped in the Canvas and to allow the model to be composed with another model – based on composition rules (explained next).

The TGIF.json file contains these critical pieces of information:

1. **Self** – section: This section describes the name and version of the ML model which is displayed on the Design Studio Web UI.
2. **Services.provides** – section: This section provides a list of services offered by the ML Model. At present only the name of the operation and JSON representation of its input messages is included here. The information provided in Services.provides and Services.calls section is used for determining the composability of a pair of output and input ports of the ML Models.
3. **Services.calls** – section: This section provides a list of output messages of the services offered by the ML Model. As explained earlier, these output messages are consumed by the services provided by other ML Model(s). The name of the operation (same as provided in Services.provides) and JSON representation of its output messages is included here. The information provided in Services.provides and Services.calls section is used for determining the composability of a pair of output and input ports of the ML Models.
4. **Artifacts.Uri** – section: This section contains the location of the docker image of the ML Model. This information is used by the Blueprint file to retrieve the docker image of the model in order to deploy it in cloud.

Model Composition

The main function of the Design Studio is to compose the ML Models to produce a meaningful application.

Criteria for Model Composition

Currently the Design Studio implements a simple model composition strategy based on matching the output message of the output port of one ML Model to the input message of the input port of another ML Model.

In the Design Studio a pair of ports are compatible if the requirement of one port can be matched with the capability of another port. Or if the output of one model can be consumed by the input port of another model so as to get some service from the latter.

The matching criterion is based on comparing the Protobuf message signature of the output port to the message signature of the input port of another model.

A pair of output and input messages are compatible if all the following conditions are satisfied:

1. The number of tags in both their message signatures is the same
2. For each tag number, the fields on both the sides are of the same type
3. For each tag number, the fields on both the sides have the same role – repeated, optional, etc.

NOTE: the field names are not taken into consideration for determining compatibility.

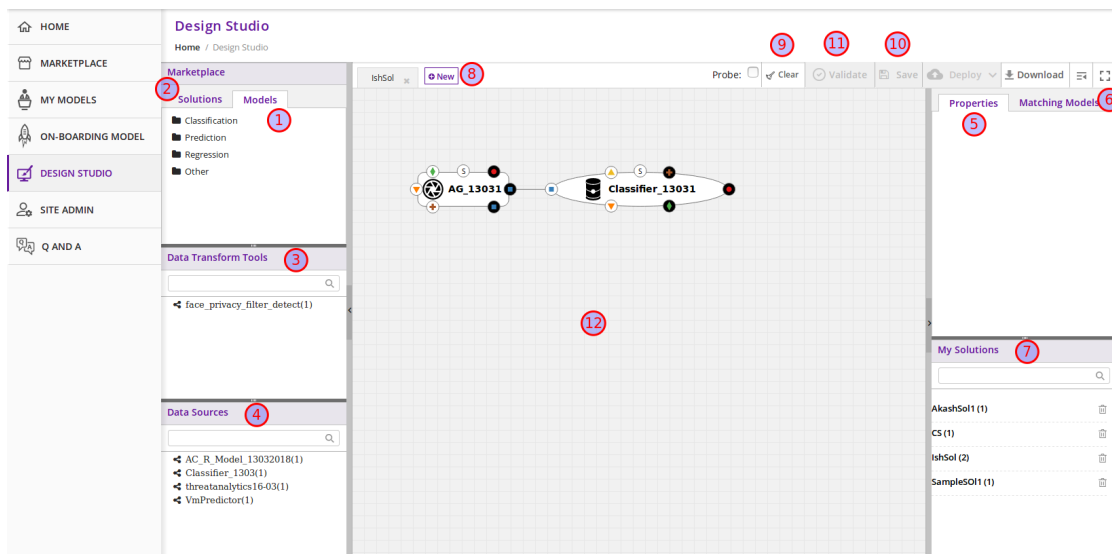
A pair of output and input ports are compatible, if all the following conditions are satisfied:

1. They both produce and consume identical number of messages
2. Each message on one side is compatible with another message on another side, as per the compatibility definition given above

NOTE: the message names are not taken into consideration for determining the compatibility.

User Interface Tour

The Design Studio UI, shown below, consists of a Design Canvas with a grid background in the center flanked on left and right hand side by the Models & Solutions Catalog Palette and the Properties & Matching Model Palette. At the bottom of the Design Canvas is the Validation Console. At the top are the New, Clear, Validate, and Save buttons.



1. **Models Tab:** Displays the catalog of the ML Models – the basic building blocks used for creating composite solutions. The models are currently classified under four categories – Classification, Prediction, Regression and Others
2. **Solutions Tab:** Displays the catalog of composite solutions (built out of basic building blocks) that have either “Public” visibility or belong to the logged in persons “Organization”
3. **Data Transformation Tools:** A set of useful data transformation utilities are displayed here. Currently there is a Data Mapper which performs mapping between some basic Protobuf data types, such as int32, string, float, double and bool. An Aggregator is another utility that is planned to be deployed there.
4. **Data Sources:** This section is meant to represent data sources which feed the ML Models. It could be any entity that produces data that is consumed by ML Models and Data Transformation Tools, such Data Lakes, Databases, Cell Towers, Network elements which produce data such as Routers, Switches, etc.
5. **Properties Tab:** Displays the properties of elements – such as ML Models and Messages inside the Ports. If an ML Model is selected by the user in the Design Canvas, it displays the name, type, owner, provider and tool kit type information. If a Message inside the ML port is selected by the user, it displays the Protobuf message signature – such as the fields of the message, their name, type, tag and role (repeated, optional etc.)
6. **Matching Models Tab:** If a requirement (output) port of an ML Model is selected in the Design Canvas, then this tab shows a list of all models that have matching capabilities (in their input ports). The user can then drag the desired model in the Design Canvas and connect the output port to the input port. If a capability (input) port of an ML Model is selected in the Design Canvas, then this tab shows a list of all models that have matching requirements (in their output ports).
7. **My Solutions:** Displays the catalog of composite solutions (built out of basic building blocks) that are marked “Private” to the logged in user. When the user clicks on an existing solution, that solution is displayed in the Design Canvas. The user can then make modification to the solution and save it as a separate solution by providing a new name or new version or both.
8. **New:** The user clicks this button to create a new composite solution.
9. **Clear:** The user clicks this button to clear an unsaved solution.
10. **Save:** The user clicks this button to save a new composite solution or save changes to an existing solution. The user is prompted to provide the name, version and a description of the solution. The user can make modification to the solution and save it as a separate solution by providing a new name or new version or both.
11. **Validate:** The user clicks this button to validate a composite solution created in the Design Canvas. Both the success and error messages are displayed in the Validation Console. If the solution is valid then a Blueprint.json file is created which is used to deploy the solution in the target cloud.
12. **Design Canvas:** This is where the users drags one or more ML Models – the basic building blocks to create a composite solution or if the user clicks on an existing solution in Solutions or My Solutions tab, it is displayed in the Design Canvas.

Ports of the Model

A model may have multiple ports. A Requirement (output) port is represented by a filled-in circle and a Capability (input) port is represented by an empty circle. The matching pair of ports are represented by identical icons inside their ports, such as diamonds, rectangles, triangles, + sign, etc.

Composition Based on Port Matching

The Design Canvas is the place where the user performs model composition based on the port matching criterion discussed earlier. The Design Canvas ensure that only matching ports are connected via a link. It does not allow non matching ports to be connected, thereby facilitating the design – time validation of the composite solution.

How to name the ML Model

A model name is automatically generated when a model is dragged from the “Models” catalog palette into the Design Canvas. The user can change the name by double clicking on the existing name and overwriting on it.

How to name the Link

Double click on the link – a text box appears, type the name of the link.

On Click of the Model

The model properties such as its name, owner, company, toolkit (Scikit, TensorFlow, R, etc.) are displayed in the Property box.

On Click of the Link

The link properties such as its name appears in the Property box.

On Hover over a Port

The name of the operation and name of either the input or the output message, depending on the port type, pops up in Design Canvas.

On Click of the Port

If the user clicks on an Output (Requirement) port, then all ML Models that have the matching input (Capability) ports are displayed in the Matching Models tab. If the user clicks on an Input (Capability) port, then all ML Models that have the matching Output (Requirement) ports are displayed in the Matching Models tab

On Click of the Message

When the user does a mouse click on a port, then operation and message name(s) pop up. Now the user can click on the message and Protobuf message signature appears in the Property tab.

Validation Console

When the user requests the validation of the composite solution, the Validation Console pops up from the bottom of the Design Canvas. This is where all the success and error messages related to the validation gets displayed.

Data Brokers

A Data Broker retrieves the data from passive data sources and converts it into protobuf format. The Data Broker provides the data to the Models via the Model Connector. The Model Connector explicitly requests the Data Broker retrieve data from data sources, receives the data in response, and provides the data to the Models.

Data Brokers are displayed in Data Sources palette in the lower left corner of the Design Studio.

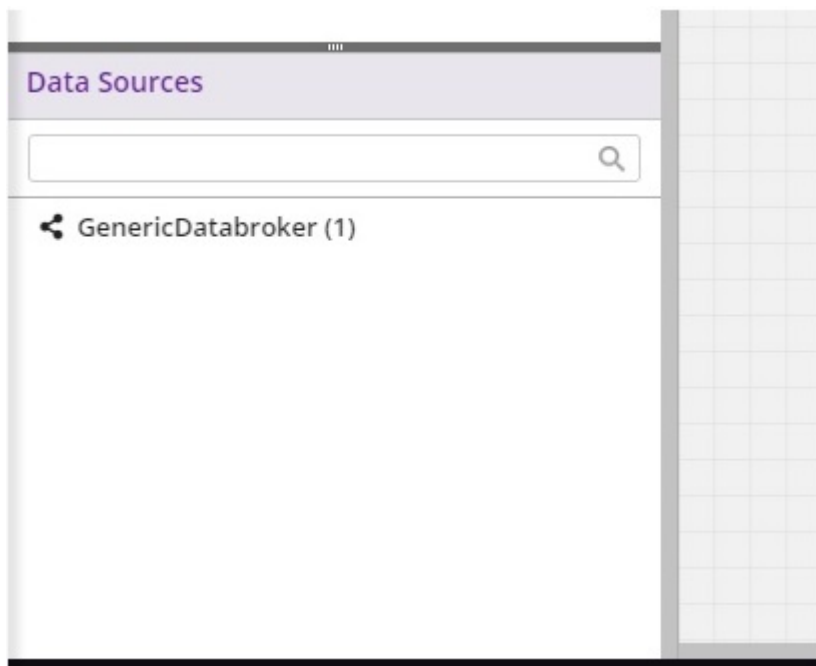
Acumos supports the following types of Data Brokers:

1. Database Data Broker (SQL DataBroker)
2. File system Data Broker (HDFS File System, UNIX, Hadoop, CSV, JSON)
3. Network Data Broker (Router, Switch, etc.)
4. Zip Archive Data Broker.

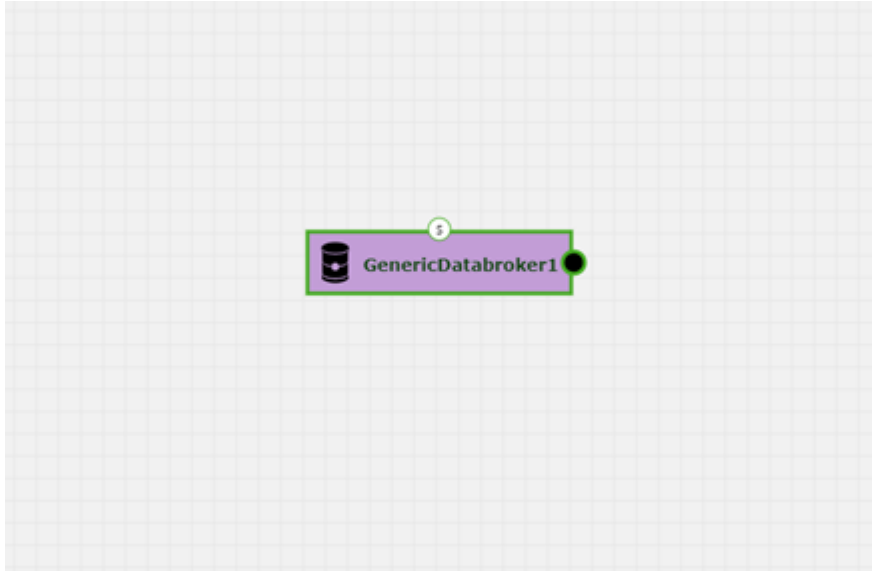
Note: The Data Broker functionality requires that specific a “Data Transformation & Toolkit” model be on-boarded in order for the functionality to be enabled. If you do not see Data Brokers in the **Data Transform Tools** palette, contact your Acumos Admin for further information.

Working With a Data Broker

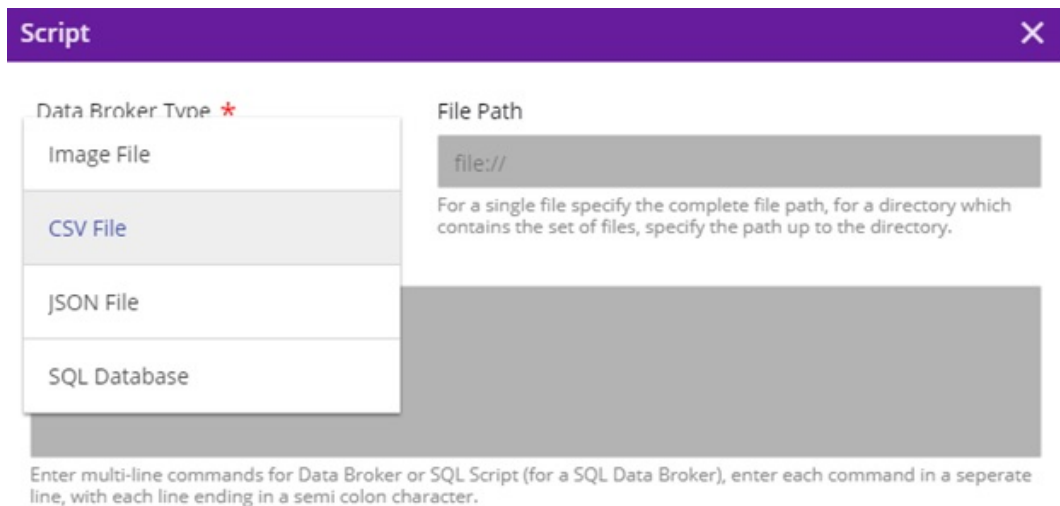
DataBrokers are displayed in the Data Sources palette.



1. Create/load a solution in Design Studio.
2. Select the GenericDataBroker (latest version) from the list of “Data Sources” and drag onto the canvas



3. Connect the GenericDataBroker output to the input of the first model
4. Select the “S” port at the top of the GenericDataBroker node, which will bring up the “Script” dialog popup
5. On click of “s” port of the Data Broker to allow input a free form text – a set of (one or more) Database SQL statements or a set of file system scripts. Note: these scripts are passed on to the Data Broker after they get deployed by the MP – Portal



6. Script Entry UI has the ability to edit (add, delete, modify, copy and paste) the script.
7. The Design Studio has the capability of storing the script and validating it from the UI after clicking on the Done button, which saves this into a back end CDUMP file. When reloading the solution on the design canvas, the saved script from the CDUMP file is loaded as well.

The current databroker supports csvDatabroker and sqlDatabroker. The user will need to enter following details:

CSV Databroker

The 'Script' dialog box has a purple header with a close button (X). It contains the following fields and controls:

- Data Broker Type ***: A dropdown menu with 'CSV File' selected.
- File Path**: A text input field containing 'file://'. Below it is a note: 'For a single file specify the complete file path, for a directory which contains the set of files, specify the path up to the directory.'
- Enter Script**: A large text area for entering multi-line commands. Below it is a note: 'Enter multi-line commands for Data Broker or SQL Script (for a SQL Data Broker), enter each command in a separate line, with each line ending in a semi colon character.'
- Upload Sample Data File ***: A file selection button labeled 'Choose File' next to the text 'No file chosen'.
- Select ***: Two radio buttons: 'First row contains field names' and 'First row contains Data'.
- Buttons**: 'Cancel' and 'Done' buttons at the bottom.

1. Data Broker Type = CSV
2. File Path = Will be populated during deployment
3. Enter Script = Will be populated during deployment
4. Choose File = select a sample CSV file with your test data from the local machine, which has the format e.g. for a model that takes two double values:

```
f1, f2
2.0, 4.0
```

5. Select "First row contains field names" or "First row contains data" based on the file uploaded
6. Click **Done**

SQL Databroker

Script

Data Broker Type *

SQL Database

JDBC URL

http://

For a single file specify the complete file path, for a directory which contains the set of files, specify the path up to the directory.

Enter Script

Enter multi-line commands for Data Broker or SQL Script (for a SQL Data Broker), enter each command in a separate line, with each line ending in a semi colon character.

Upload Sample Data File *

Choose File No file chosen

Database Name *

Select JDBC Datasource Class Name *

Select JDBC Class name

Cancel

Done

1. Data Broker Type = SQL
2. JDBC URL = Greyed out
3. Enter Script = Greyed out
4. Choose File = select a file with CREATE TABLE schema loaded in it, in order to parse the table contents for mapping. (Table Name and table field details are retrieved from the schema.)
5. Select the jdbc driver name from the dropdown which supports the file uploaded. (Currently, we only support mysql)
6. Enter the database name in which the table is present. It will be shown as below.

Script

Data Broker Type *
SQL Database

JDBC URL
http://
For a single file specify the complete file path, for a directory which contains the set of files, specify the path up to the directory.

Enter Script

Enter multi-line commands for Data Broker or SQL Script (for a SQL Data Broker), enter each command in a separate line, with each line ending in a semi colon character.

Upload Sample Data File *
Choose File testcreate.sql

Database Name *
cds

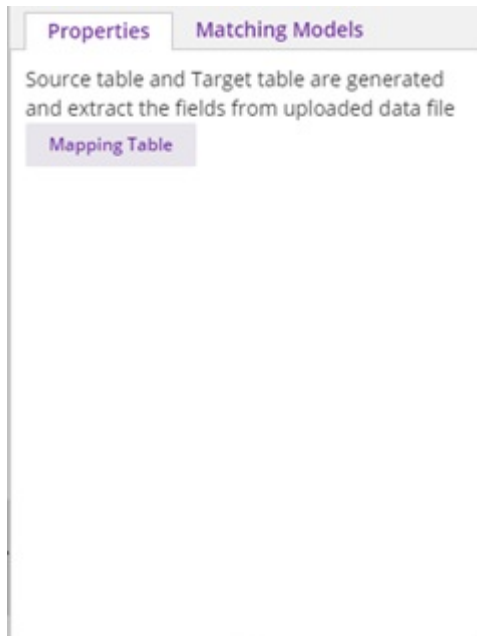
Select JDBC Datasource Class Name *
MySQL

Cancel Done

7. Click **Done**

Auto-Generating Source Table from Script

After the user clicks on “Done” button, if the schema file is valid, the file contents will be parsed, field names will be extracted, and the source table will get auto-populated. The source table can be viewed by clicking on the “Mapping Table” button present in the properties. If the schema file cannot be parsed, an error message will be displayed.

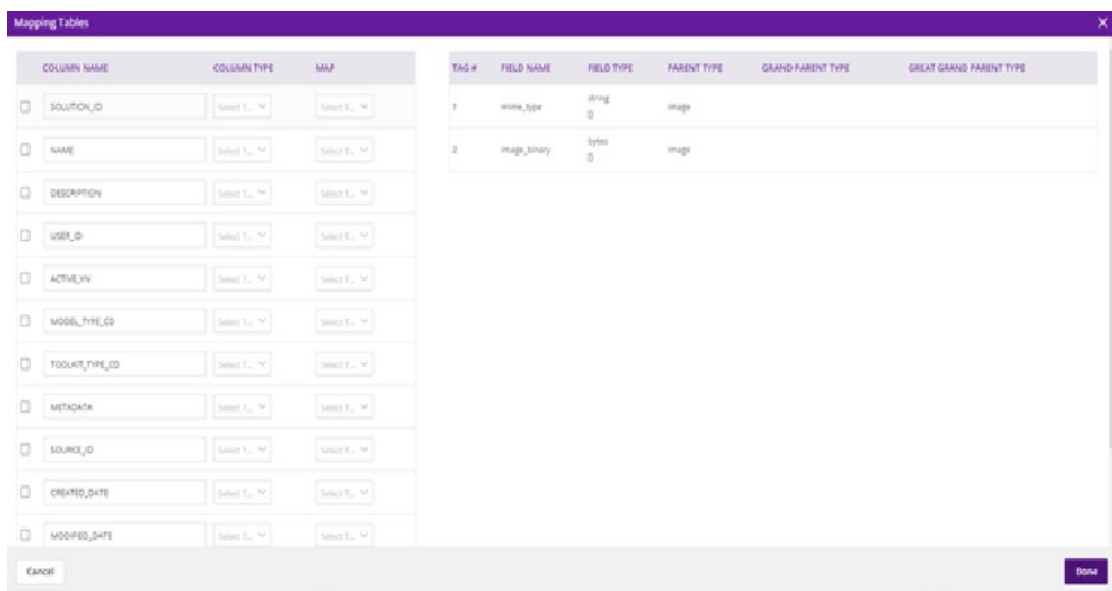


The Target table will be generated by using the single protobuf file of one of its input messages. After connecting the ML Model to the Data Broker output, click on the output port of the Data Broker. Then the property box will display the protobuf input message of the ML Model.

Auto-Generating Target Table from Protobuf File

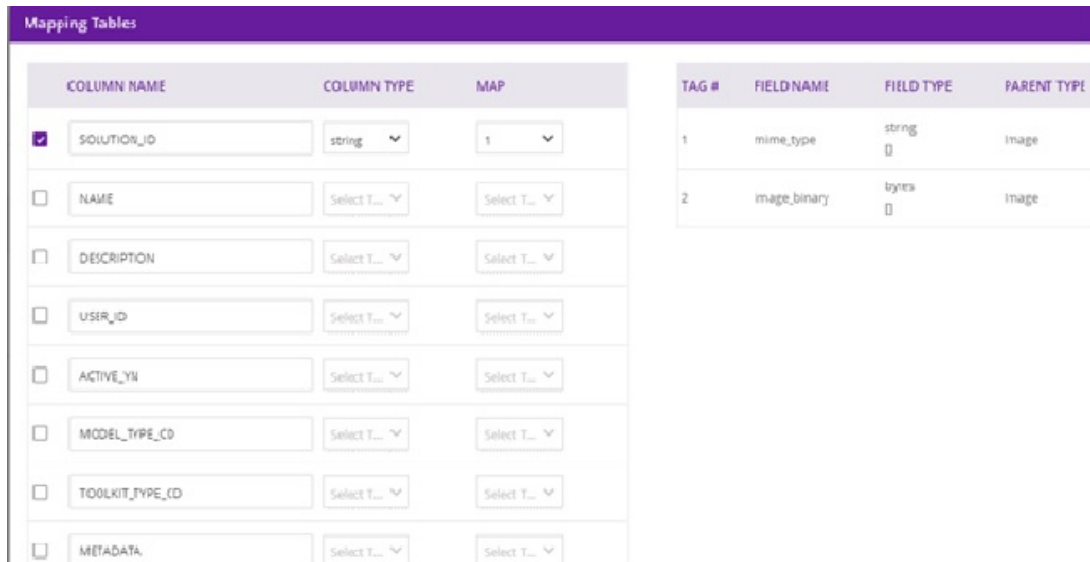
Once the output of Data Broker is connected to the input of ML model, the Data Broker acquires its message signature and generates the target table as per the protobuf specification. The Target table contains the N number of rows, where N is number of basic field types in the protobuf message (basic field name and basic field type).

On the right, under **Properties**, select **Mapping Table** and the resulting dialog should display the following:



Select each source field from the table, select the field type from the drop down, and the target tag to be mapped to the field. The target tags are captured from the protobuf specification of the model that is connected to Data Broker.

When you have mapped all fields, select **Done**.



COLUMN NAME	COLUMN TYPE	MAP
<input checked="" type="checkbox"/> SOLUTIONS_ID	string	1
<input type="checkbox"/> NAME	Select T...	Select T...
<input type="checkbox"/> DESCRIPTION	Select T...	Select T...
<input type="checkbox"/> USER_ID	Select T...	Select T...
<input type="checkbox"/> ACTIVE_YN	Select T...	Select T...
<input type="checkbox"/> MODEL_TYPE_CD	Select T...	Select T...
<input type="checkbox"/> TOOLKIT_TYPE_CD	Select T...	Select T...
<input type="checkbox"/> METADATA	Select T...	Select T...

TAG #	FIELDNAME	FIELD TYPE	PARENT TYPE
1	mime_type	string	image
2	image_binary	bytes	image

The Design Studio will save the Source-Table-to-Target-Table mappings in the CDUMP file when the **Save** button is clicked. The Design Studio will retrieve the saved mappings from the CDUMP file and display the them in the Property Box when reloading the solution in the design canvas.

Saving

Above the canvas, select the **Save** button and enter the details of the solution. This will be saved in “My Solutions” area.

Deploying

To generate a TOSCA blueprint for deployment to a cloud environment, select the **Validate** button. If validation is successful, the deploy button will be enabled. On click of any of the cloud platforms, you will be redirected to the **Manage my Model- Deploy to Cloud** page. At this point, this model should be usable with the Data Broker when deployed. See the [Deploying a Model](#) section for more information on deploying models.

Dockerizing the DataBroker

The Data Broker is implemented as a Java jar package. After the composite solution has been successfully validated in the Design Studio, the Composition Engine performs the following functions:

1. Retrieves the code of the Data Broker from a specific location in Nexus repository
2. Creates the Protobuf Wrapper for the Data Broker based on the output message that the Data Broker acquires at its output port when it is connected to an input port of an ML Model in the Design Studio. The input message is of Protobuf type string. This Wrapper converts:
 - (a) From Java to Protobuf types for the outgoing messages.
 - (b) From Protobuf to Java types for the incoming messages.
3. Creates the jar file of the Data Broker
4. Converts the jar to Microservices

5. Creates the Docker Image of the Data Broker Microservice from its jar file
6. Stores the Docker image of the Data Broker in the Docker repository (or Nexus repository)
7. Stores the location of the docker image in the TGIF.json of the Data Broker
8. Stores the location of the docker image in the Blueprint.json file (after successful validation)

Generic Data Mapper

The Generic Data Mapper service enables a user to connect two ML models A and B, where number of output fields of model 'A' and input fields of model 'B' are same. A user is able to connect the field of model A to the required field of model B. The Generic Data Mapper performs data type transformations between Protobuf data types. To use the Data Mapper successfully, a user should be well aware of the output value of each field of model A and the expected input value of each field of model B to get desired final output.

The Generic Data Mapper has the following functionality:

1. Maps data types between a pair of incompatible ports of the ML Models – map the data type of an output port to the data types of an input port
2. Any output port of a ML Model can be connected to a Data Mapper, and the Data Mapper can be connected to any input port of the ML Model
3. Composition Rule: From the Design Studio composition perspective a Data Mapper can accept any inputs and produce any outputs, depending on the ML models that are connected to its input and output side. So its requirements and capability will be indicated any
4. Performs transformation between basic Protobuf types only

A Generic Data Mapper can be found in Design Studio UI under the “**Data Transform Tools**” palette.

Note: The Generic Data Wrapper functionality requires that a specific “Data Transformation & Toolkit” model be onboarded in order for the functionality to be enabled. If you do not see a Generic Data Mapper in the **Data Transform Tools** palette, contact your Acumos Admin for further information.

Connecting Incompatible Ports

Coming soon.

Message Splitter and Collator Tools

The Splitter and Collator functionality supports the [Directed Acyclic Graph \(DAG\)](#) Topology in the Design Studio, supports both the split and join semantics, and provides the different collation semantics at join point.

Note: The Splitter and Collator require that specific “Data Transformation & Toolkit” models be onboarded in order for the functionality to be enabled. If you do not see a Collator and a Splitter in the **Data Transform Tools** palette, contact your Acumos Admin for further information.

The Splitter

The Splitter enables connecting one model to multiple models to support message splitting (broadcast and parameter splitting capability). The Splitter supports Copy-Based and Parameter-Based schemes.

The Splitter has the following features:

- Selectable splitting schemes
- Accept a single input message and produces multiple output messages of the same or different type, depending upon the splitting scheme
- The number of outputs is determined dynamically at run time
- The input port supports a single incoming link
- The output port can support one or more outgoing links in case the output message has a subset of input parameters
- Supports the addition and deletion of its input links and its output link
- Performs: 1) unmarshalling of input protobuf message into native format; 2) splitting of the unmarshalled message according to the splitting scheme selected by the user; and 3) marshalling the output messages back into Protobuf format before sending on its output port

The Collator

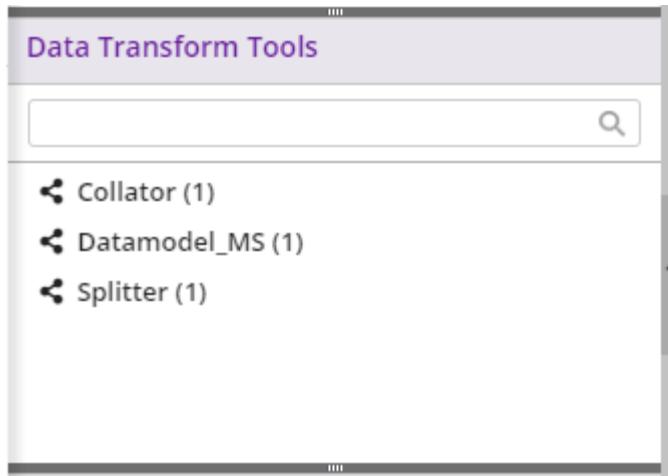
The Collator enables connecting multiple models and combining the input from the models into a single output message. The Collator supports Array-Based and Parameter-Based collation.

The Collator has the following features:

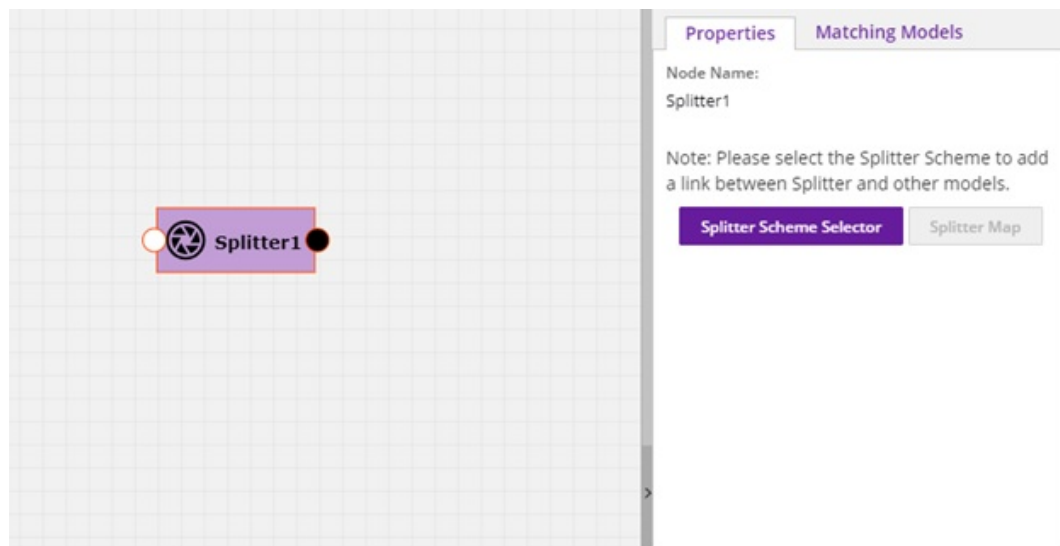
- Selectable collation schemes
- Accept a variable number of inputs (N) and produces a single output
- The number of inputs is determined dynamically at run time
- The input port can support one or more incoming links in case the output message has a subset of input parameters provided by one incoming link
- The output port will support only one outgoing link
- Collator will support the addition and deletion of its input links and its output link
- Performs: 1) unmarshalling of input protobuf messages into native format; 2) collation of unmarshalled messages according to the collation scheme selected by the user; and 3) marshalling the collated message back into Protobuf format before sending on its output port.

Using the Splitter

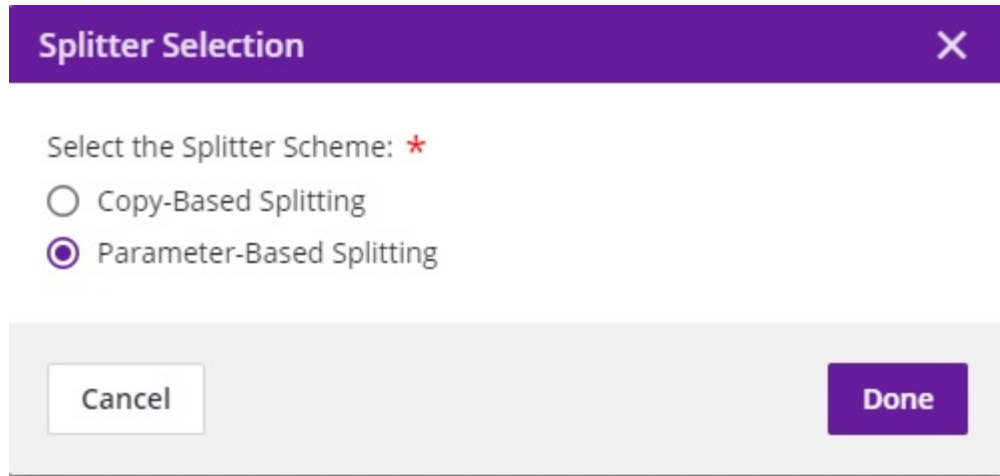
Splitters are located in the **Data Transform Tools** palette.



1. Create/load a solution
2. Select a Splitter from the list in the **Data Transform Tools** palette and drag onto the canvas
3. After dragging the Splitter on the canvas, it should be displayed as rectangular icon and the input (on left-hand side) and output (on right-hand side) ports should be disabled

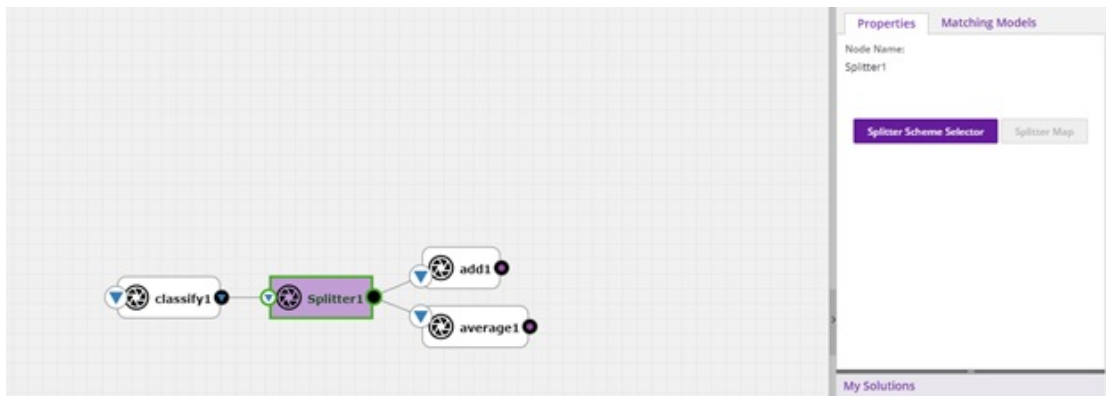


4. On the right hand side in properties box, there will be a **Splitter Scheme Selector** button; click on it to display the pop-up window for scheme selection



Copy-Based Splitting

The user will be able to establish a link to Splitter ports only after selecting the splitter scheme and Send (copy) the input message on all outgoing links. Make sure that both input and output message signatures are identical and both the input and output message data is same. The “Splitter Map” button (present in the properties panel) **will not be enabled** as it copies the complete message from the source model to all the target models connected to the Splitter. As it is a copy-based splitting, if either input or output port is connected to a ML model, it acquires the message signature from the ML model on both the input and output ports.



Connecting the Splitter Input Port First

If the input port of the Splitter is connected first to the output port of the (producing) Model, then:

- Splitter must display the message signature of the output port of the producing model on its input port.
- When the output ports are connected later on, Splitter must make sure that all output messages have the same message signature as the message signature of the input message, otherwise the connection should not be allowed.

Connecting the Splitter Output Port First

If the output port of the Splitter is connected first, then:

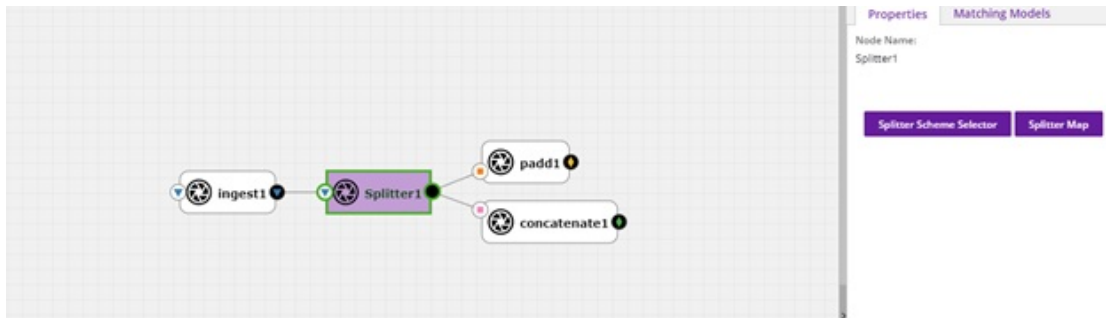
- The Splitter should allow the first outgoing link to be connected to its output port without any validation, and make a temporary copy of its message signature.
- For the second and subsequent links that are connected to the output port, the Splitter must make sure that their message signature is the same as that of the first message signature, otherwise the connection should not be allowed.
- When the input port is connected later on, the Splitter must make sure that its message signature is the same as that of output message signature on its output port, otherwise the connection should not be allowed.

Validation Rules

- May have one or more links connected at its output port (Note: The case of one link at the output port does not make sense but is allowed.)
- May have only one link connected at its input port
- Must have the same message signature for messages coming out of its output port into all the outgoing links
- The message signature at the input and output port of the Splitter must be the same
- The output of a Splitter cannot be connected to the input of a Collator

Parameter-Based Splitting

Split the input message based on its signature into (top – level) parameters and send different parts / parameters on different outgoing links. Make sure that Input and output message signatures are different and the collection (i.e., specific arrangement) of output message signatures represents the input message signature. The “Splitter Map” button **will be enabled**. The user must connect one model at the input and one or more models at the output port of the splitter. Once the input and output ports are connected, the source and target tables are auto populated that can be viewed when the user clicks on Splitter map button.



There is a mapping area in the Splitter Map pop-up that allows the user to copy a source field (parameter) to the target field(s) (parameter(s)). It is a drop down having all the source table tags. All the target-side fields must be mapped for a successful validation. At least one source field should be mapped to a target field. A source field may be mapped to multiple target fields.

For every mapping, there is an Error Validator that helps the user know if the mapping is valid or invalid (i.e. if the parameter types on both sides match).

SOURCE MESSAGE				MAPPING AREA		TARGET MESSAGES				
PARAMETER NAME	PARAMETER TYPE	PARAMETER ROLE	PARAMETER TAG #	SOURCE TAG MAPPING - COPY A SOURCE FIELD TO TARGET FIELD	ERROR VALIDATOR	PARAMETER TAG #	PARAMETER NAME	PARAMETER TYPE	PARAMETER ROLE	TARGET MODEL NAME
f1	double		1	3	✓	1	s	string		concatenate1
f2	double		2	1	✓	2	d	double		concatenate1
s	string		3	1	✓	1	f1	double		path1
				2	✓	2	f2	double		path1

If any of the mapping is invalid, then validation and blueprint generation will fail.

SOURCE MESSAGE				MAPPING AREA		TARGET MESSAGES				
PARAMETER NAME	PARAMETER TYPE	PARAMETER ROLE	PARAMETER TAG #	SOURCE TAG MAPPING - COPY A SOURCE FIELD TO TARGET FIELD	ERROR VALIDATOR	PARAMETER TAG #	PARAMETER NAME	PARAMETER TYPE	PARAMETER ROLE	TARGET MODEL NAME
f1	double		1	Select Tag	✗	1	s	string		concatenate1
f2	double		2	Select Tag	✗	2	d	double		concatenate1
s	string		3	Select Tag	✗	1	f1	double		path1
				Select Tag	✗	2	f2	double		path1

Connecting the Splitter Input Port First

If the input port of the Splitter is connected first to the output port of the (producing) Model, then:

- Splitter must display the message signature of the output port of the producing model on its input port
- When the output port is connected later on, Splitter's output port should remain ANY

Connecting the Splitter Output Port First

If the output port of the Splitter is connected first, then:

- The Splitter should allow all the outgoing link(s) to be connected to its output port without any validation
- When the input port is connected later on, the Splitter should allow only one incoming link to be connected to its input port without any message signature validation

The parameter-based splitter should split the input message into first-level parameters and Copy the required input parameters on each of its outgoing link based on the information in the Splitter Map. Arrange these parameters in a sequence based on the parameter ordering information in the message signature on that outgoing link. The Splitter should aggregate all parameters that needs to be sent to a single target in a single message. The Parameter-based Splitter should perform binary-to-native format conversion before collation and native to binary conversion after collation.

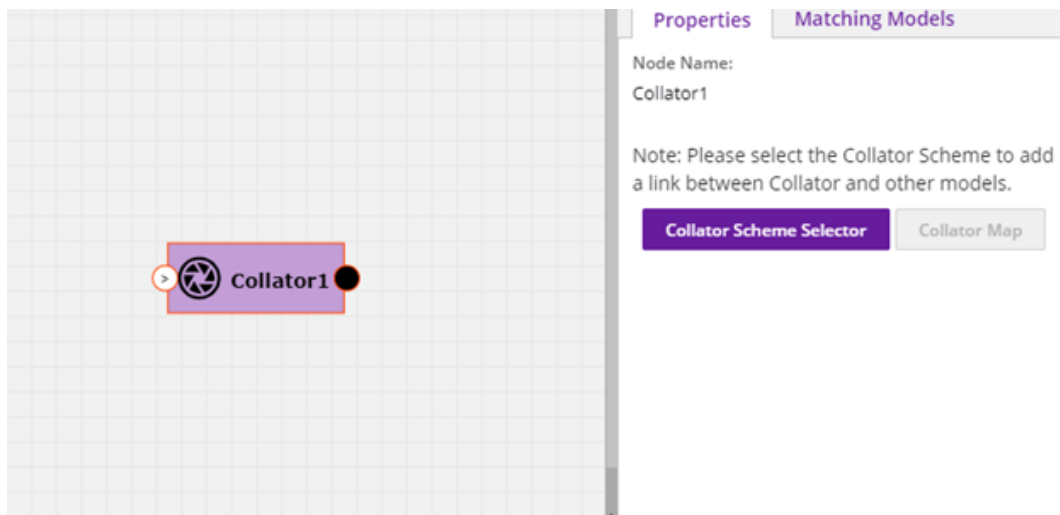
Validation Rules

- The Splitter allows a mapping between a pair of source and target parameters only if their message signatures match, otherwise an error should be indicated in the mapping area to allow the user to correct it.
- A parameter on the source side may be mapped to more than one parameter/tag on the target side as long as target parameters belong to different target models
- Two or more parameters from the source cannot be mapped to the same parameter/tag in the target message

- When no parameters from the source are mapped to the parameters on the target message, then the Splitter displays an error until the source model is deleted or at least one of the source side parameters is mapped to a target side parameter
- All parameters on the target side models must be mapped to their matching source side parameters, otherwise an error is shown in the mapping area until this condition is satisfied
- When both the source and target side parameters have been mapped correctly, no errors are displayed
- The Splitter input port may have only one incoming link
- The Splitter output port can have one or more outgoing links (a single outgoing link case is possible if this link provides all parameters required by the single target model.)

Using the Collator

1. Create/load a solution
2. Select a Collator from the list in the **Data Transform Tools** palette and drag onto the canvas
3. After dragging the Collator on the canvas, it is displayed as rectangular icon and the input (on left hand side) and output (on right hand side) ports should be disabled



4. On the right hand side in properties box, there will be a Collator Scheme Selector; click on it to display the pop-up window for scheme selection

Collator Selection
✕

Select the Collator Scheme: *

☒ Array-Based Collation

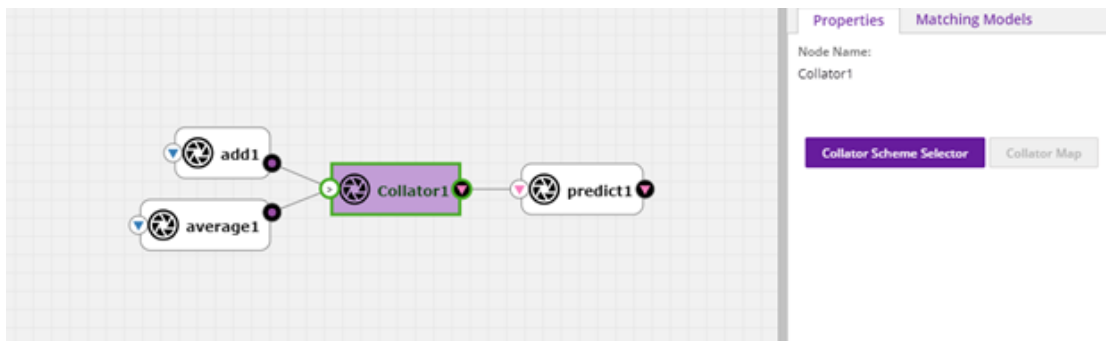
☐ Parameter-Based Collation

Cancel

Done

Array-Based Collation

Each incoming link provides complete message data, output the collection (an array) of all input message data. Each input message signature is the same, but message content (data) may be different and the output message signature is a collection (i.e., an array, or a repeated structure) of input message signatures. The Collator Map button **will not be enabled**. The output port of Collator only connects to a model which has a repeated complex message signature of the message at the input port (i.e., if the message signature at input port is “M”, the message signature of the output port is “repeated (M)”. **All** links connected to the input port must carry the same message signature “M”. That means the output message signature is an array of input message signature (on the input links) which are of same message type. If either of one of the input or the output port of the Collator is connected to an ML Model, then the input port acquire the message signature “M” and the output port acquires the message signature “repeated(M)”.



The Collator waits until all messages are received on all of its input ports, based on the incoming link information in the CDUMP file. When all the messages have been received, the Collator should convert the binary messages into native format and construct an array of the input messages. Collator should convert the array of input messages into a protobuf repeated message structure before delivering it on the output port.

Connecting the Collator Output Port First

If the output port of the Collator is connected first to the input port of the (consuming) Model, then:

- Collator displays the message signature of the input port of the consuming model on its output port; note that this will be a repeated Protobuf data type
- When the input ports are connected (later on), the Collator makes sure that all input messages have the same message signature as message signature of the output message except that input should not be a repeated type

Connecting the Input Port First

If the input port of the Collator is connected first, then:

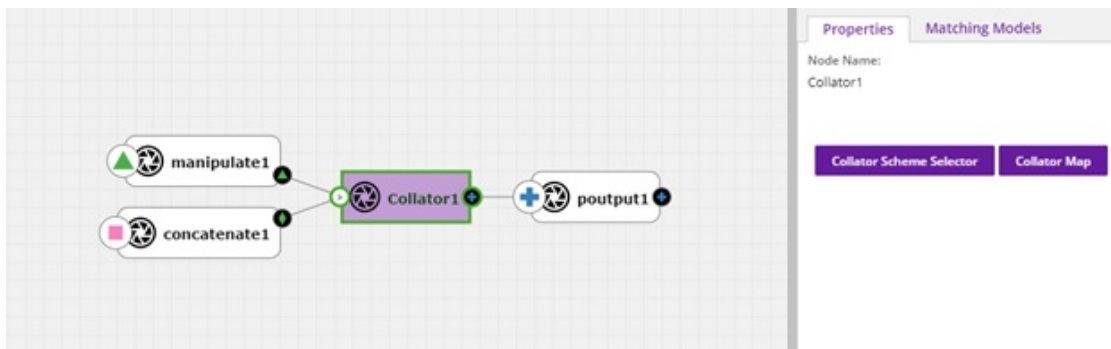
- The Collator allow the first incoming link to be connected to its input port without any validation and makes a temporary copy of its message signature.
- For the second and subsequent links that are connected to the input port, the Collator makes sure that the message signature is the same as that of the first message signature, otherwise the connection is not be allowed
- When the output port is connected later on, the Collator makes sure that its message signature is the same as that of repeated (input message signature), otherwise the connection is not be allowed

Validation Rules

- An array – based collator can have one or more links connected at its input port; note: in case of a single input link the user may want to convert a Model’s output message into an “array of message” structure before feeding it to the target model which only accepts an array structure
- The Collator can have only one link connected at its output port
- The array-based collator must have the same message signature for messages arriving at its input port from all the incoming links
- The output port of an array based collator must have a “repeated” structure of the message signature of its incoming links
- The output of a Collator cannot be connected to the input of a Splitter

Parameter-Based Collation

If a Parameter-based collation scheme is selected, the Collator Map button **will be enabled**. The user must connect one model at the output port and one or more models at the input port. Once the input and output ports are connected, the source and target tables are auto-populated and can be viewed by clicking on the Collator map button. As it is parameter-based collation, Collator output port acquires the message signature of the input port of the ML model connected to it and collator input port remains “ANY” which means any can be connected to it.



There is a mapping area in the Collator Map pop up, which allows the user to map (i.e., copy) a source field to a target field. It is a drop down having all the target table tags. All the Target side fields must be mapped for a successful validation. At least one field from each source should be mapped to a target field, otherwise a validation error is displayed. Multiple source fields cannot be mapped to the same target field. A source field cannot be mapped to more than one target field.

For every mapping, there is a error validator that helps the user know if the mapping is valid or invalid (i.e. if the parameter types on both sides match). If any of the mapping is invalid, then validation and blueprint generation will fail.

Collator Mapping Table

SOURCE MESSAGES					MAPPING AREA		TARGET MESSAGE			
DATA SOURCE NAME	PARAMETER NAME	PARAMETER TYPE	PARAMETER ROLE	PARAMETER TAG #	TARGET TAG MAPPING - MAP A SOURCE FIELD TO TARGET FIELD	ERROR VALIDATOR	PARAMETER TAG #	PARAMETER NAME	PARAMETER TYPE	PARAMETER ROLE
manipulate1	difference	double		1	Select Tag	✗	1	difference	double	
manipulate1	product	double		2	Select Tag	✗	2	product	double	
manipulate1	average	double		3	Select Tag	✗	3	average	double	
concatenate1	joined	string		1	Select Tag	✗	4	square	double	
							5	x	string	

Cancel Done

Validation Errors

Collator Mapping Table

SOURCE MESSAGES					MAPPING AREA		TARGET MESSAGE			
DATA SOURCE NAME	PARAMETER NAME	PARAMETER TYPE	PARAMETER ROLE	PARAMETER TAG #	TARGET TAG MAPPING - MAP A SOURCE FIELD TO TARGET FIELD	ERROR VALIDATOR	PARAMETER TAG #	PARAMETER NAME	PARAMETER TYPE	PARAMETER ROLE
manipulate1	difference	double		1	1	✓	1	difference	double	
manipulate1	product	double		2	2	✓	2	product	double	
manipulate1	average	double		3	3	✓	3	average	double	
concatenate1	joined	string		1	5	✓	4	square	double	
							5	x	string	

Cancel Done

No Validation Errors

Connecting the Collator Output Port First

- The output port of Collator should acquire the message signature of the input port of the Model, then collator's source table should be auto populated with details viz., the name of the source, parameter name, parameter type, its tag number and an initially empty mapping field in the collator map, based on the information contained in the protobuf file of the source
- Collator should analyse the output port message signature and split it into its component parts (i.e., into parameters which have tag numbers associated to them).

Connecting the Input Port First

- In this case the input port of Collator remains as ANY
- Collator's target table is auto populated with details the parameter name(s), parameter type(s) parameter tag number(s), and the mapping field should be populated with the list of output tag numbers, based on the information contained in the protobuf file of the target.

Validation Rules

- The collator will allow a mapping between a pair of source and target parameters only if their message signatures match, otherwise an error should be indicated in the mapping area to allow the user to correct it. (Alternatively show Pop Up when the mapping is invalid)
- A parameter on the source side cannot be mapped to more than one tag on the target side
- Two or more parameters from the source cannot be mapped to the same tag in the target message

- Multiple parameters from a single data source (i.e., Model) may map to different tags in the target message
- When no parameters from a source are mapped to the target message (figure – 4), then the Collator should show an error until that data source is deleted or one of the parameters is mapped
- Collator must make sure that at least one parameter from each source have been mapped to their corresponding target side tags, otherwise an error should be shown in the mapping area, until this condition is satisfied (i.e., that link is removed and therefore the corresponding unnecessary entries are removed)
- Collator must make sure that all target side parameters have been mapped, otherwise an error should be shown against those entries in the mapping
- When both the source and target side parameters have been mapped correctly, the error mark should be taken away
- The output port should have only one outgoing link
- The input port can have one or more links (a single link case is possible if this link provides more parameters than that required by collator's output port).

Saving

Above the canvas, select the **Save** button and enter the details of the solution. This will be saved in “My Solutions” area.

Deploying

To generate a TOSCA blueprint for deployment to a cloud environment, select the **Validate** button. If validation is successful, the deploy button will be enabled. On click of any of the cloud platforms, you will be redirected to the **Manage my Model- Deploy to Cloud** page. At this point, this model should be usable with the Data Broker when deployed. See the [Deploying a Model](#) section for more information on deploying models.

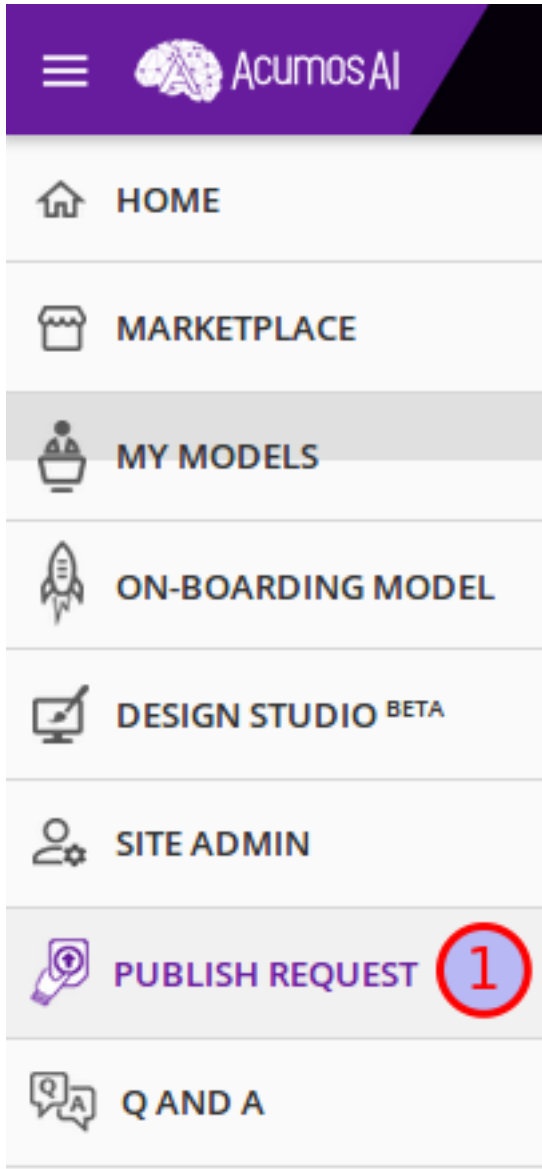
2.2 Portal and Marketplace Publisher Guide

2.2.1 Publishing Models

Users who have been assigned the Publisher role have the ability to:

1. Access a list of requests to publish to the Public Marketplace
2. Check the contents of the solution before publication
3. Approve or decline publication of the the solution

Publishers have a **Publish Request** left menu item.



Requests List

From the **Publish Request** page, a Publisher is able to view requests to publish models to the Public Marketplace, view model details by clicking on a request, approve a request, and decline a request.

HOME	Publish Request						
MARKETPLACE	Home / Publish Request						
MY MODELS	Current Request (0 active request)						
ON-BOARDING MODEL							
DESIGN STUDIO BETA							
SITE ADMIN							
PUBLISH REQUEST							
Q AND A							

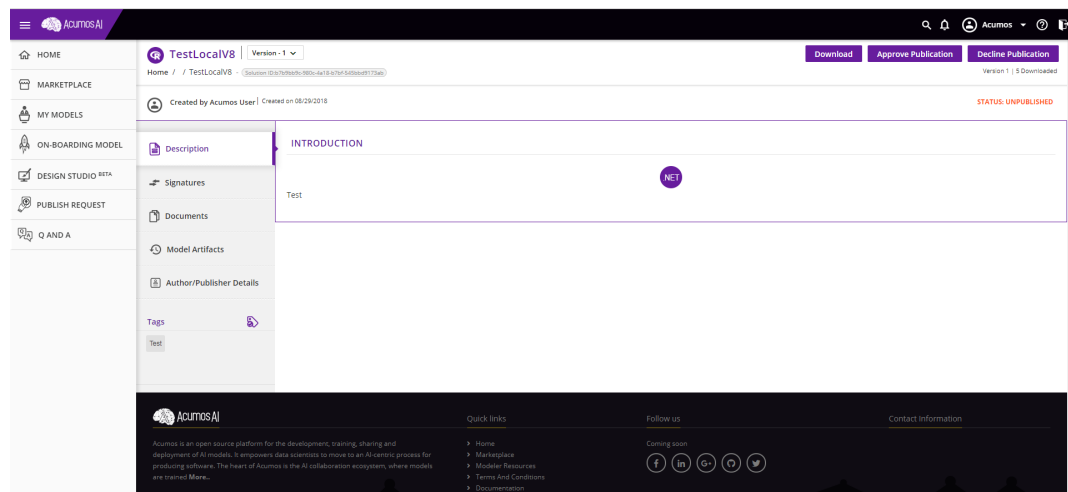
MODEL NAME	VERSION	REQUESTER	STATUS	REQUEST STATUS	COMMENTS	ACTION
Wks.ThreatAnalytics	48365dc-4971-471e-4e12...	Organization	Organization	Pending	View	ⓘ ✖
PUB_3110101_MEM	43293711-4355-4539-8033...	Sumit Berman	Public	Approved	View	ⓘ ⓧ
PUB_3110101_MEM	43293711-4355-4539-8033...	Sumit Berman	Public	Approved	View	ⓘ ⓧ
PUB_3110101_MEM	4487ec5b-457d-4a8f-5a71-3...	Sumit Berman	Public	Approved	View	ⓘ ⓧ
PUB_3110101_MEM	4487ec5b-457d-4a8f-5a71-3...	Sumit Berman	Public	Approved	View	ⓘ ⓧ
PUB_3110101_MEM	4487ec5b-457d-4a8f-5a71-3...	Sumit Berman	Public	Approved	View	ⓘ ⓧ
PUB_3110101_MEM	4487ec5b-457d-4a8f-5a71-3...	Sumit Berman	Public	Approved	View	ⓘ ⓧ
PUB_3110101_MEM	4487ec5b-457d-4a8f-5a71-3...	Sumit Berman	Public	Approved	View	ⓘ ⓧ
PUB_3110101_MEM	4487ec5b-457d-4a8f-5a71-3...	Sumit Berman	Public	Approved	View	ⓘ ⓧ
Empty 3	57a7756c-4386-4345-4a76...	Alexis Deep	Organization	Declined	View	ⓘ ✖
Test	82738023-5467-4649-4d71...	Alexis Deep	Public	Approved	View	ⓘ ⓧ
TestModel1010	58705620-4667-4671-4671...	Alexis Deep	Organization	Approved	View	ⓘ ⓧ

Showing 10 of 10 Publish Requests

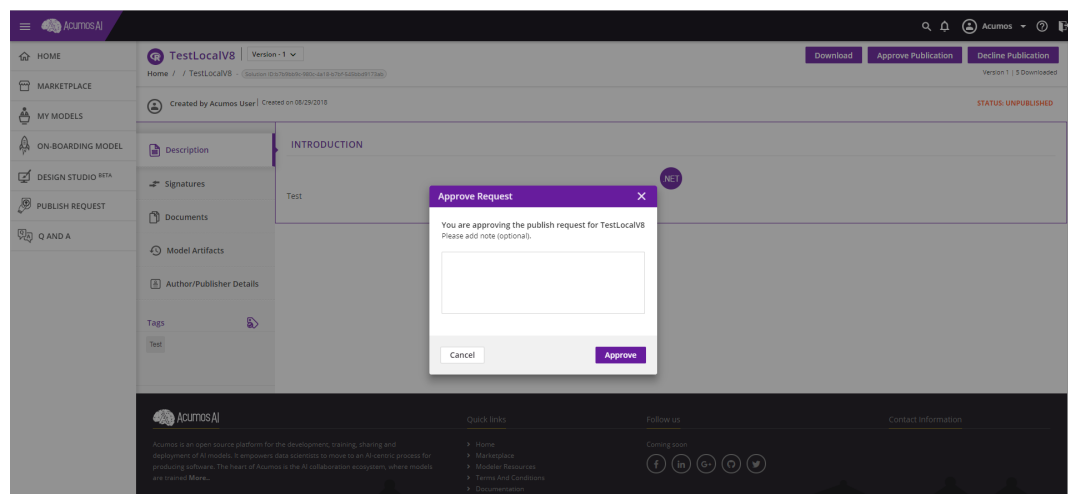
1. Enter filter criteria into the **Filter** text box in the upper right corner
2. Change the number of requests/page
3. Page navigation

Approving or Declining a Request

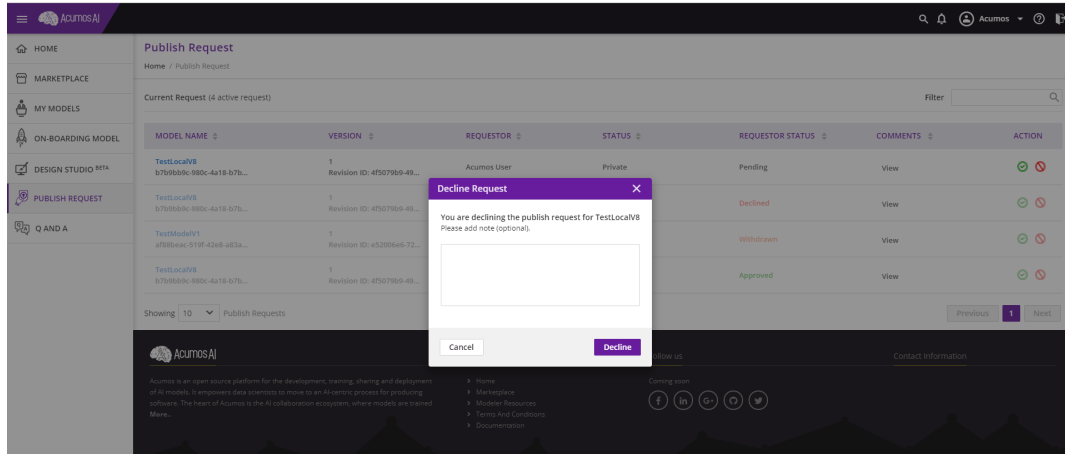
1. Click on Model Name to view the details of that specific model.



2. After reviewing the model details, approve or decline the request by clicking on the button **Approve Publication** or **Decline Publication** buttons on the top right corner of the screen. Note that a Publisher may not approve or decline his/her own model.



3. Provide the reason in the comments box of the popup. Then click on **Approve** or **Decline**
4. The Publisher can also approve/decline the request from the Current Requests list by clicking on the action buttons



2.3 Portal and Marketplace Admin Guide

2.3.1 Introduction

This user guide describes how to use the Admin Interface for Acumos portals.

What is Covered by the Admin Interface

The Admin Interface is a web-based tool for a site admin to:

1. Monitor the site activity
2. Manage users and change their roles.
3. Update and edit the site content that is managed by the CMS.
4. Configure the site.
5. Manage Federation relationships: configure peer gateways and set up subscriptions to that peer's public marketplace.
6. Manage Federation requests for specific models.
7. Configure workflows.

Admin Access to the Acumos Portal

When a new Acumos Portal is deployed, a default admin user will be created in the process of deployment by common-dataservice database setup scripts. A default username (“*TBD*”) and password (“*TBD*”) will be assigned to the admin and must be changed upon first login, as the password will be set to “expired” when created.

Users can be assigned the role of Admin via the Portal UI at “Site Admin” / “User Management”. Select the user and pick “admin” under the “Change Role To” dropdown. The user will need to logout and back in for the “Site Admin” tools to be visible.

The Site Admin Dashboard provides the toolset that admins use to manage the Acumos portal. The Site Admin Dashboard can be accessed by site admins only. Admins will have the “SITE ADMIN” option on the left of the portal UI when they login.

The following sections address each tool under the Site Admin Dashboard.

2.3.2 Site Monitoring

Site monitoring requires the installation of the ELK stack components. Please see the *Platform Operations, Administration, and Management (OA&M) User Guide* for installation instructions as well as how to access the Kibana dashboards from the **Monitoring** tab.

2.3.3 User Management

This tab lists all user accounts on the portal and enables:

- Searching for users and selecting a list of users by role
- Adding a user
- Bulk activation and deactivation of users
- Changing the role of a user to one or more of the system roles
- Creation of new roles that restrict use of various portal features

Note: A user may be **deactivated** but not **deleted**

System Roles

Role	Description	Permissions
MLP System User	Default role assigned to new accounts	On-board a model, use the Design Studio and Marketplace
Admin	Portal Administration	MLP System User, site administration
Publisher	Model Publishing	MLP System User, publish models to the Public Marketplace

Add New User

Selecting the “Add New User” button will present a dialog in which new user details can be provided, including:

- First Name (mandatory)
- Last Name (mandatory)
- User Name (mandatory): must be unique, and not already used for some other account
- Email (mandatory): must be a valid format email address, and not already used for some other account
- Password (mandatory): must contain at least eight characters, which should have at least one upper case and one lower case letter, numbers and symbols like, ! # @ \$ * &. If the password is determined to be “weak”, a stronger password must be selected or the “Add” button will not be selectable.
- Role (mandatory): one of the defined roles, by default “MLP System User”, “admin”

An option to send the new user an account creation email is provided. The email will be sent from the defined email address of the admin user that added the new user.

Add New User
X

First Name *

Last Name *

User Name *

Email *

Password *

Confirm Password *

☒ Send this password to the new user by email

Role *

Select

Cancel

Add

Password must contain at least eight characters, which should have at least one upper case and one lower case letter, numbers and symbols like, ! # @ \$ * &.

Add New Role

Selecting the “Add New Role” button will present a dialog in which roles can be defined, with the options:

- Role Name (mandatory): must be unique, i.e. not already used as a role name
- Role Name (mandatory): Assignment of one or more of the following permissions:
 - Access to the Design Studio
 - Access to the Marketplace
 - Access to Onboarding

TODO: provide further description of these permissions

Add Role

Role Name *

User Permissions *

☐ Design Studio
 ☐ Market Place
 ☐ On Boarding

Cancel

Add

2.3.4 Site Content

Admin users should verify that the 'Auth URL' and 'Push URL' values shown on the 'On-Boarding Model' page are correct for the Acumos platform. The values are defined in the component template for the Portal-BE component, and can be updated by modifying the template used to deploy the Portal-BE component, and then removing/re-deploying the Portal-BE component.

2.3.5 Site Configuration

This tab enables configuration of the following:

- Logos enabling the portal owner (typically a company or other organization) to brand the site: ***TODO: explain how these are presented***
 - Co-Branding Logo
 - Header Logo
 - Footer Logo
- siteInstanceName: Name for the portal site, used to differentiate the site when users first visit (before login), ala "Explore the <siteInstanceName> Marketplace", "Discover <siteInstanceName>", etc.
- ConnectionConfig: set parameters for networking ***(TODO: clarify how these are used)***
 - socketTimeout
 - connectionTimeout
- EnableOnboarding: option to enable or disable the onboarding feature for modelers
- validationText: If you have Text Check configured as part of your Publishing workflow, a keyword scan will be performed on a model's artifacts. That keyword scan will use the comma separated list of keywords entered in this field.

2.3.6 Federation

Federation enables peering of Acumos sites for the purpose of developing a broader public marketplace of models. Federation involves several main processes described in the following sections:

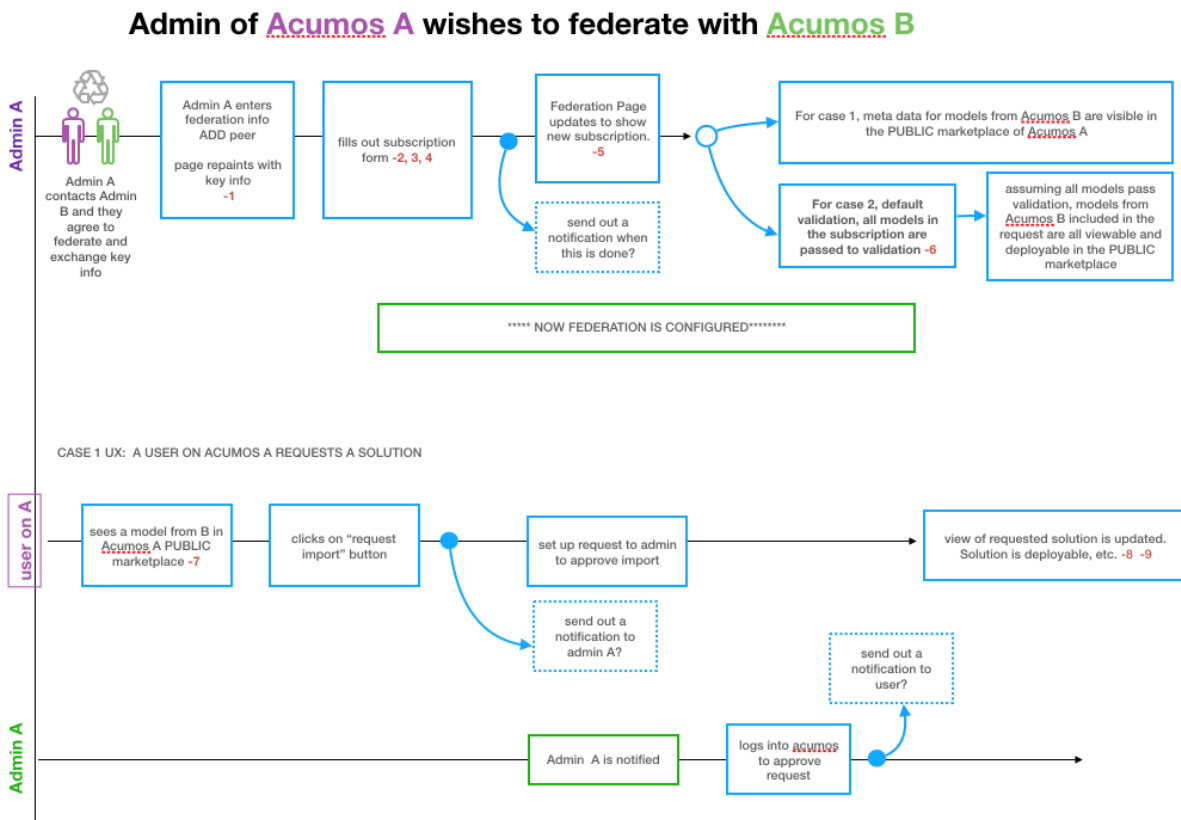
- Establishing connectivity between peers
- Creating peer relationships between portals
- Subscribing to models published in the public marketplace of peers

Following is an overview of the federation process.

Once connectivity has established between peers, and a peer relationship established in the admin UI (step 1 below), the local admin can setup a public marketplace subscription to some set of models from the peer (steps 2-4 below) and receive confirmation that the subscription is setup (step 5 below). An optional step at this point is the validation of the models received over the federation API, prior to insertion of their metadata (not the actual model artifacts) into the local portal's public marketplace.

A user of the local portal can then discover new models and request access to them, which may need to be approved by a local admin, per the local admin's customization of the related workflow. Once approved, the model artifacts are retrieved and stored in local repositories, and made available for the user to download, launch, etc.

The federation process is outlined below:



Configuring an Acumos Instance for Federation

Instructions are in the `../../submodules/federation/docs/config`.

Establishing connectivity between peers

Peer relationships are initiated and negotiated by peer companies/organizations through processes outside the scope of this guide, and may depend upon network provisioning also outside the scope of this guide. Once peering has been agreed to, the following steps are required as prerequisites to the use of the portal UI for setting up federation:

- Since portals and related public APIs are accessed only over HTTPS, each portal must have at least one SSL certificate to use for the following publicly exposed services and API endpoints, or one certificate for each:
 - Portal web service and onboarding API endpoint, both accessed through an HTTPS proxy setup as part of portal deployment. Further description of these aspects are provided in ***(TODO: link to portal deployment guide)***
 - The federation API endpoint
- Each peer shares their public certificate or certificate authority (CA) details for the federation API endpoint, and the other peer installs the CA certificate by:
 - On the host of the federation-gateway service, importing the CA cert into the SSL truststore as setup for federation-gateway.
 - restarting their federation-gateway service, and verifying basic connectivity to the peer gateway.

Creating peer relationships between portals

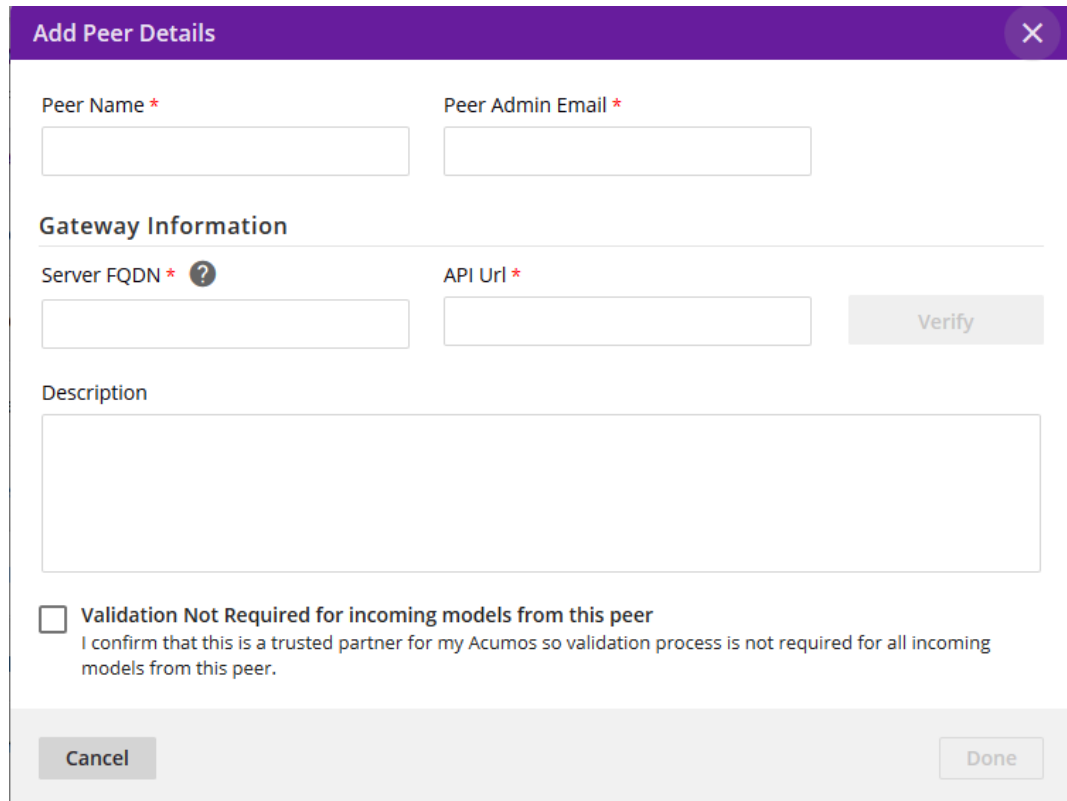
Enable Federation

The first step in creating peer relationships to enable federation overall, but clicking the slider on the upper-right of the Federation tab, labeled as “Federation”. By default, this shows as “Disabled”, and when clicked will change to “Enabled”. ***(TODO: clarify the effect that enabling or disabling federation has on the following processes)***

Add Peer

Selecting the “Add Peer” button will present a dialog in which a peer relationship can be defined and validated, with the options:

- Peer Name (mandatory): name to assign this peer in the peer list. ***(TODO: clarify any syntax/semantic restrictions/implications)***
- Peer Admin Email (mandatory): must be a valid format email address, and not already used for some other peer
- Server FQDN (mandatory): Domain name associated with a valid server certificate as signed by the peer CA as described above. The domain name must be resolvable via DNS, or by local host configuration (for an example of the latter, see the OnClick Deploy guide) ***(TODO: link to guide)***
- API Url (mandatory): URL for the peer federation API endpoint. Must be of the form “https:<FQDN or IP address>:<port configured for the federation API>”.
- Description (optional): Anything that helps describe this peer.
- Validation Not Required (optional): Enables subscribed models to be listed in the marketplace without pre-validation. ***(TODO: link to validation guide)***



The "Add Peer Details" dialog box is a purple-themed window with a close button (X) in the top right corner. It contains the following fields and controls:

- Peer Name ***: A text input field.
- Peer Admin Email ***: A text input field.
- Gateway Information**: A section header.
- Server FQDN ***: A text input field with a help icon (?) to its right.
- API Url ***: A text input field.
- Verify**: A button located to the right of the API Url field.
- Description**: A large text area for entering a description.
- ☐ **Validation Not Required for incoming models from this peer**: A checkbox with a label. Below it, a note reads: "I confirm that this is a trusted partner for my Acumos so validation process is not required for all incoming models from this peer."
- Cancel**: A button at the bottom left.
- Done**: A button at the bottom right.

Actions on peer list entries

Once a peer has been added, it is displayed in the list of peers. The list provides various information and actions for peer entries:

- Name, gateway info, contact: as entered in the “Add Peer” dialog
- Subscriptions: manage subscriptions (see section below)
- Status: “inactive” when the peer is added; “active” once the “Activate” icon (✓) is clicked. Goes back to “inactive” when the “Deactivate” icon (✗) is clicked. ***TODO: clarify effects of activation/deactivation***
- Status change icon: “Activate” icon (✓) when added or inactive, or “Deactivate” icon (✗) when active.
- Edit (✎): edit peer details (see section below)
- Delete (🗑): removes the peer entry
- Self: identifies whether the peer entry is for the local portal as selected when the peer was added. “Mark as Self” when added or currently not set to self. “Remove as Self” when set to self. Clicking the current setting will toggle to the other value.

NAME	GATEWAY INFO	CONTACT	SUBSCRIPTIONS	STATUS	ACTION	SELF
acumos-a	https://acumos-a.com:90...	admin@acumos-a.c...	View/Add	Inactive since 21.03.2018		Mark as Self

Edit Peer

Selecting the “Edit Peer” icon () will bring up the same dialog as “Add Peer” with the addition of the “Verify” button being selectable.

Selecting the “Verify” button will:

- ***TODO: clarify what the verify does, affects, and any subsequent actions once a verification process is successful/unsuccessful***

Subscriptions

Selecting “View/Add” in the “Subscriptions” column will bring up a dialog for management of subscriptions with the peer. When first added, the peer subscriptions list will be empty.

The dialog titled "View/Add Subscriptions for acumos-a" features a purple header with a close button. Below the header, there is a "Subscription List" tab and a "+ New Subscription" button. A "Filter" input field with a search icon is on the right. The main content area has a table with headers: "SUBSCRIPTIONS ID", "CREATED ON", "UPDATED ON", and "FREQUENCY OF UPDATE". The table body contains a single row with the text "No Result Found". At the bottom, there are "Cancel", "Full Access" (with a dropdown arrow), and "Done" buttons.

Add Subscription

Selecting “New Subscription” in the “View/Add Subscriptions” dialog will display search options for models in the peer catalog:


The dialog titled "View/Add Subscriptions for opnfv06" features a purple header with a close button. Below the header, there is a "Subscription List" tab and a "← Back To Subscription List" button. The main content area contains search options: a "Model ID" field with a "Filter" input and a search icon, followed by an "OR" radio button, a "Category" dropdown menu with "Select Category" as the placeholder, another "OR" radio button, a "Toolkit type" dropdown menu with "Select toolkit" as the placeholder, and finally an "All Models" checkbox. At the bottom, there are "Cancel", "Full Access" (with a dropdown arrow), and "Done" buttons.

In the “Model ID” field, to search for a specific model by ID (the ID displayed in a browser location field when you are browsing the model, e.g. “solutionId=079779dd-6962-4f7e-8655-fe6310242b81”), enter the ID (e.g. 079779dd-6962-4f7e-8655-fe6310242b81), and if that model is available in the remote marketplace it will appear in the dialog, e.g.:

View/Add Subscriptions for acumos_ist2

Subscription List [← Back To Subscription List](#)

Model ID: 079779dd-69 OR Category: Toolkit type: OR ☐ All Models

 TestmodelV1
by on 1520822809000
Model belongs to: AT&T

Select Frequency of update

Other options include:

- To search by Category (Classification, Data Sources, Data Transformer, Prediction, Regression), select the category from the “Category” drop-down. To narrow the search to a specific toolkit within that category, or to search only by toolkit, select the toolkit (Composite Solution, Design Studio, H2O, Probe, R, Scikit-Learn, TensorFlow, Data Broker, Training Client, ONAP).
- To search for all models, select the “All Models” box. ***TODO: explain why it may not be selectable***

If any matching models are available in the remote marketplace they will appear in the dialog, e.g. as below. From here you can select:

- select an automatic refresh of models matching the search criteria (Hourly, Daily, Monthly) from the “Select Frequency of Update” drop-down. Or select “Update on demand” for manual updates.
- “Full Access” or “Partial Access”: ***TODO: describe***
- Clear Catalog: ***TODO: describe***

View/Add Subscriptions for acumos_ist2

Subscription List [← Back To Subscription List](#)

Model ID: OR Category: Toolkit type: OR ☐ All Models

Data Transformer

Select Frequency of update

SOLUTION ID	SOLUTION NAME	CATEGORY	TOOLKIT
eb86a965-5163-4abf-ab14-59c3b12cf805	face_privacy_filter_detect		

To save the subscription as selected above, select the “Add To Subscription List” button. The subscription will be added to the list for this peer, e.g. as below.

View/Add Subscriptions for acumos_ist2

Subscription List + New Subscription Filter

	SUBSCRIPTIONS ID	CREATED ON	UPDATED ON	FREQUENCY OF UPDATE			
+	ID 495	03/13/2018	03/21/2018	Hourly	Preview	Trigger	
+	ID 496	03/13/2018	03/21/2018	Hourly	Preview	Trigger	
+	ID 497	03/13/2018	03/21/2018	Hourly	Preview	Trigger	

Cancel Full Access Done

To see the details for a subscription, select the “+” icon, which will expand the display with details and options, e.g. as below. From here you can:

- Select a new “Frequency of Update”
- Delete the subscription by selecting the trashcan icon (image18)
- Preview: ***TODO: describe the function***
- Trigger: ***TODO: describe the function***

View/Add Subscriptions for acumos_ist2



Subscription List + New Subscription Filter

	SUBSCRIPTIONS ID	CREATED ON	UPDATED ON	FREQUENCY OF UPDATE			
-	ID 495	03/13/2018	03/21/2018	Hourly	Preview	Trigger	
<ul style="list-style-type: none"> • Model Type: Classification • Toolkit Type: 							
+	ID 496	03/13/2018	03/21/2018	Hourly	Preview	Trigger	
+	ID 497	03/13/2018	03/21/2018	Hourly	Preview	Trigger	

Cancel Full Access Done

Managing Subscription Requests

The “Requests” tab enables an admin to manage subscription requests. *This tab is not fully implemented, but it will enable these options:*

- Approve, by selecting the check icon ()
- Deny, by selecting the ‘X’ icon ()

TODO: further explain what happens upon these actions

Site Admin
Home / Requests

DASHBOARD OPTIONS

- Monitoring
- User Management
- Site Content
- Site Configuration
- Federation
- Requests**
- Configure Workflows

REQUESTS

Current peers (5 active request)

REQUEST ID	REQUESTED DETAILS	REQUEST TYPE	SENDER	DATE	STATUS	ACTION
REQID 1	Requested Details1	Model Download	TechM 1	03/17/2018	Pending	
REQID 2	Requested Details2	Federation	Acumous 2	03/17/2018	Pending	
REQID 3	Requested Details3	Model Download	TechM 3	03/17/2018	Pending	
REQID 4	Requested Details4	Federation	Acumous 4	03/17/2018	Pending	
REQID 5	Requested Details5	Model Download	TechM 5	03/17/2018	Pending	
REQID 6	Requested Details6	Federation	Acumous 6	03/17/2018	Pending	

2.3.7 Requests

@TODO what is this tab for? this tab contains nothing but dummy data


2.3.8 Configure Workflows

A number of Acumos tasks, such as on-boarding and publishing require the user to complete a series of tasks and then initiate a back-end workflow to complete the overall task. The workflow can be described as a series of steps, all of which must succeed to complete the overall task.

The Admin of a Acumos system may include or omit steps from the back-end workflow to customize their local instance.

UI for Workflow Configuration

When “Configure Workflows” is selected, the current set of workflows and currently configured steps is displayed. When the Acumos portal is deployed, this will include the system default workflows for “On-boarding”, “Publishing to Local”, “Publishing To Public” and “Import Federated Model Work”, e.g. as below. Options from here:

- Deactivate or Assign any workflow
- Modify any workflow, by adding or deleting optional steps
 - Grayed-out steps are mandatory and cannot be deleted or modified
 - Optional steps have a “bar-in-circle” icon () which enables the step to be deleted, e.g. the Security Scan step as optional for the Onboarding work flow

WORKFLOW CONFIGURATION

On-boarding Work flow

1 Create Micro service
Onboarding | added on 01/22/2018

2 Package
Onboarding | added on 01/22/2018

3 Dockerize
Onboarding | added on 01/22/2018

4 Create TOSCA
Onboarding | added on 01/22/2018

5 Add to Repository
Onboarding | added on 01/22/2018

6 Security Scan
Onboarding | added on 01/22/2018

Deactivate Workflow

Assign Workflow

Publishing to Local Work Flow

1 Model Documentation
Mandatory for Publishing Model

2 Security Scan
Onboarding | added on 01/22/2018

3 License Check
Onboarding | added on 01/22/2018

4 Text Check
Onboarding | added on 01/22/2018

Deactivate Workflow

Assign Workflow

Publishing to Public Work Flow

1 Model Documentation
Mandatory for Publishing Model

2 Security Scan
Onboarding | added on 01/22/2018

3 License Check
Onboarding | added on 01/22/2018

Deactivate Workflow

Assign Workflow

Import Federated Model Work Flow

1 Steo 01
Federation | added on 01/22/2018

2 Step 02
Federation | added on 01/22/2018

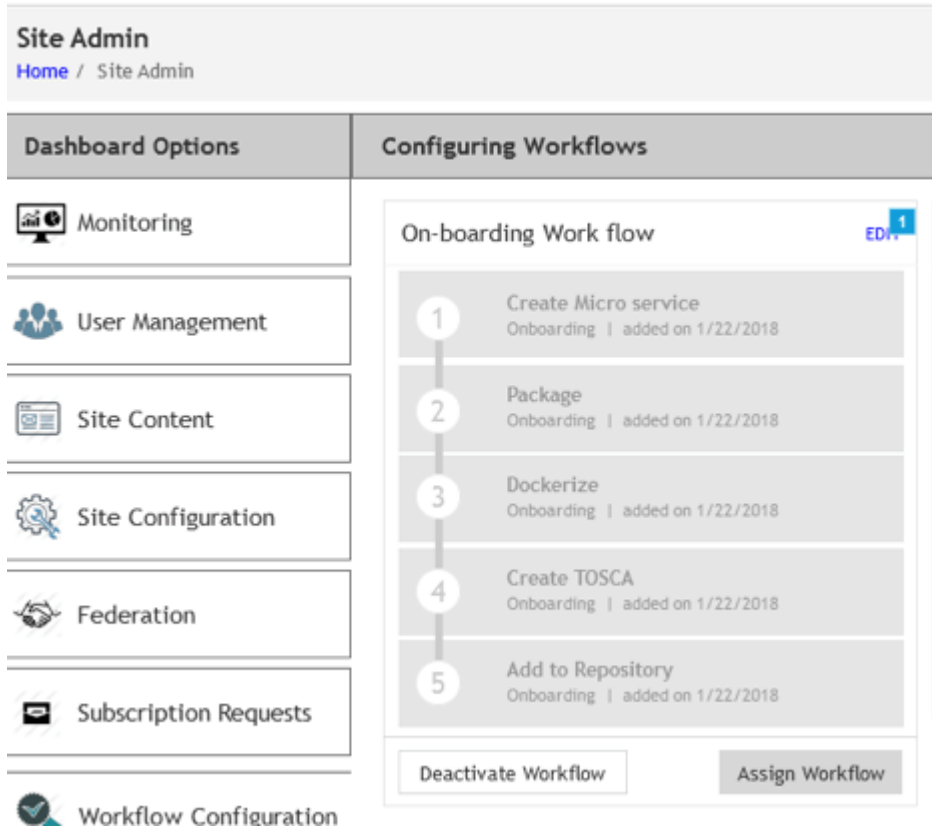
3 Step 03
Federation | added on 01/22/2018

4 Step 04
Federation | added on 01/22/2018

5 Step 05
Federation | added on 01/22/2018

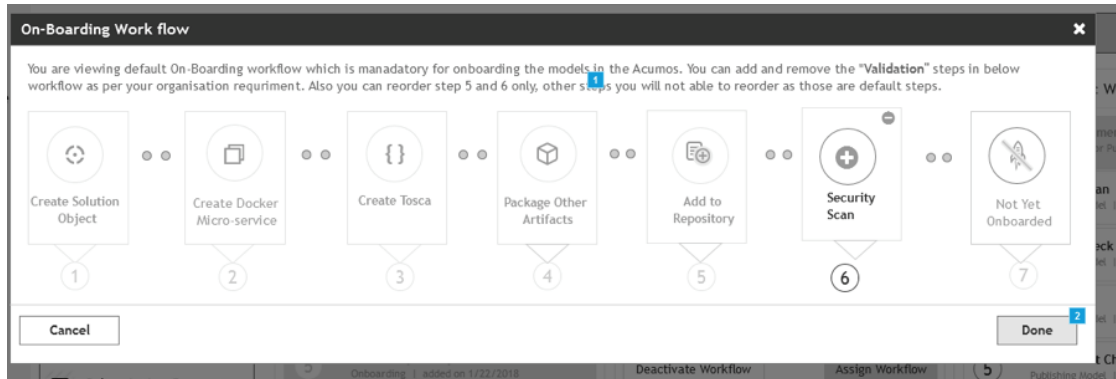
Example: Change workflow for On-boarding

Select the correct workflow and choose the EDIT button.

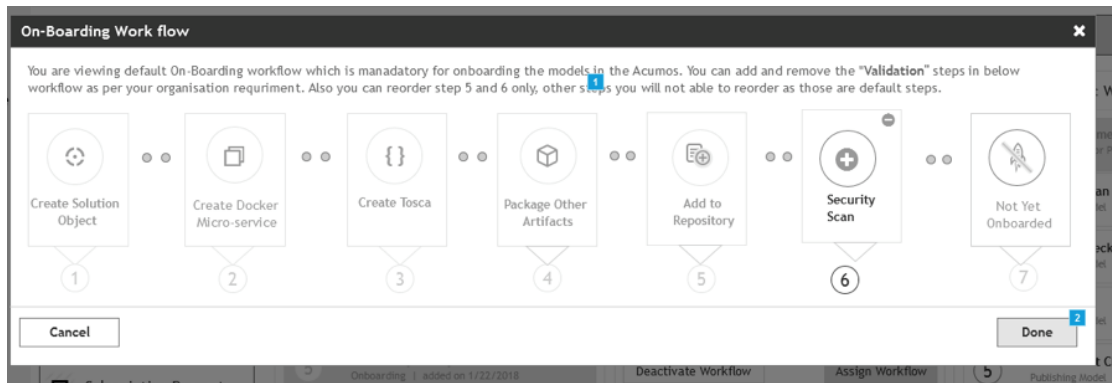


The detailed steps are shown, including the optional SECURITY step. If the SECURITY step is currently not included, and Admin wishes to add it, they click on the + symbol for that step.

Adding a Security Step:



Deleting a Security Step: Click on the “-“ button”.



Result: The new security step is shown in the workflow. To implement the change, the Admin must select **Assign Workflow** button.

Search anything you need

Site Admin

Dashboard Options

- Monitoring
- User Management
- Site Content
- Site Configuration
- Federation
- Subscription Requests
- Workflow Configuration

Configuring Workflows

Security Scan added in to On Boarding workflow successfully. Please click on Assign workflow button to apply the new workflow to your instance.

On-boarding Work flow

- 1 Create Micro service
- 2 Package
- 3 Dockerize
- 4 Create TOSCA
- 5 Add to Repository
- 6 Security Scan

Deactivate Workflow Assign Workflow

Publishing to Local Work flow

- 1 Model Documentation
- 2 Security Scan
- 3 Licence Check
- 4 Text Check

Deactivate Workflow Assign Workflow

Publishing to Public Work flow

- 1 Model Documentation
- 2 Security Scan
- 3 Licence Check
- 4 Text Check
- 5 Manual Text Check

Deactivate Workflow Assign Workflow

Import Federated Model Work flow

- 1 Step 01
- 2 Step 02
- 3 Step 03
- 4 Step 04
- 5 Step 05

Deactivate Workflow Assign Workflow

2.3.9 Addendum

On-Boarding Design Studio Toolkit Models

The Design Studio requires specific toolkit models to be on-boarded in order for Data Broker, Generic Data Mapper, Splitter, and Collator functionality to be enabled.

On-Boarding the Models

1. Download the following archived model bundles from the Design Studio [wiki](#) page:
 - Collator.zip
 - Splitter.zip

- DataBroker.zip
 - DataMapper.zip
2. Each zip file is a model that needs to be on-boarded; follow the [web on-boarding](#) instructions to upload the models.

Publishing the Models

All the models should be published to the Company marketplace. Each model needs to have Model Category and Toolkit Type set. See the *Publishing to the Company Marketplace* section of the [Managing a Model](#) page in the Portal and Marketplace User Guide for instructions.

Model	Model Category	Toolkit Type
Collator	Data Transformation	Collator
Data Broker	Data Sources	Data Broker
Generic Data Mapper	Data Transfer	
Splitter	Data Transformation	Splitter

3.1 One Click Deploy User Guide

3.1.1 Introduction

This user guide describes how to deploy Acumos platforms using the “One Click deploy” tools designed for developers or those who want a simple and automated way to deploy an Acumos platform. Currently deployment supports an all-in-one (AIO) target.

What is an AIO deploy?

The AIO deploy tools build an all-in-one instance of Acumos, with the database, Nexus repositories, and docker containers all running on a single virtual machine or physical host machine.

What’s included in the AIO tools

In system-integration repo folder AIO:

- `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. Post-install, you can get various passwords from this e.g. mysql root and user passwords if you want to do any manual database operations, e.g. see what tables/rows are created as part of Acumos deployment.
- `clean.sh`: script you can run as “`bash clean.sh`” to remove the Acumos install, to try it again etc.
- `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.
- `openssl.cnf`: OpenSSL configuration file used in self-signed certificate generation.

- `peer-test.sh`: Automated deployment of two AIO platforms, with federation and demo model onboarding. Used to test federation use cases.
- `create-peer.sh`: Automated setup of a peer relationship between two Acumos AIO deployments. Used by `peer-test.sh`.
- `create-user.sh`: Automated user provisioning and role assignment. Used by `peer-test.sh` to create users for model onboarding, and portal admins for testing federation actions on the Acumos platform.
- `bootstrap-models.sh`: Model package onboarding via curl. Optionally called by `peer-test.sh`.

In system-integration repo folder `AIO/docker/acumos`:

- docker-compose yaml files for all system components to be deployed or otherwise acted upon via `docker-compose.sh`.

In system-integration repo folder `AIO/kubernetes`:

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

3.1.2 Release Scope

Current Release (Athena)

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

- single-node (AIO) deployment of the Acumos platform under docker or kubernetes
- deployment with a new Acumos database, or redeployment with a current database and components compatible with that database version
- Component services under docker/kubernetes as named below (deployed as distinct container-based services), or installed directly on the AIO host:
 - core components of the Acumos platform
 - * Portal Marketplace: `portal-fe-service`, `portal-be-service`
 - * Hippo CMS: `cms-service`
 - * Solution Onboarding: `onboarding-service`
 - * Design Studio Composition Engine: `dsce-service`
 - * Federation Gateway: `federation-service`
 - * Azure Client: `azure-client-service`
 - * Common Data Service: `cds-service`
 - * Filebeat: `filebeat-service`
 - external/dependency components
 - * docker engine/API: `docker-service` (under kubernetes), or docker running on the AIO host for docker-based deployment
 - * MariaDB: `mariadb` running on the AIO host
 - * Kong proxy: `kong-service`
 - * Nexus: `nexus-service`

The Athena release will include these capabilities in development:

- Deployment in a multi-node configuration under kubernetes
- Deployment of or integration with a backend Shared-Data-Service (SDS) for Persistent Volume Claims (PVC) under kubernetes
- Deployment with upgrade migration of an existing Acumos database
- Deployment with integration to pre-existing external components: MariaDB, Nexus, proxy, ELK stack
- Additional platform core components
 - Metricbeat
 - OpenStack Client
 - Microservice Generation
 - Security Verification
 - Kubernetes Client
 - Docker Proxy
- Additional external/dependency components
 - ELK stack

Future Releases

Future releases may include these new features:

- Deployment in AIO or multi-node configuration on public clouds

3.1.3 Step-by-Step Guide

Prerequisites

This guide assumes:

- you are deploying either:
 - a single AIO instance of the Acumos platform, via “oneclick_deploy.sh”
 - two AIO instances of the Acumos platform as peers, via “peer-test.sh”
- each Acumos host is an Ubuntu 16.04.x desktop/server, with at least 16GB of RAM (recommended). The Acumos platform as deployed by this script on bare metal consumes currently about 6GB of RAM, so may be deployable in hosts with less than 16GB RAM.
- you are deploying the AIO platform(s) to host(s):
 - that have a hostname resolvable by DNS or through the hosts file of whatever machine you use to interact the Acumos web portal (referred to here as the “portal”) and platform APIs such as onboarding and federation.
 - that have access to the internet, either directly or through a proxy
 - to which you have full access to the target host, i.e. all ports are accessible
 - to which you have shell access (for a single AIO instance) or key-based SSH access (for peer-test deployment)
- Note the target host(s) can be another physical host, or a VM running on your workstation

Install Process

The notes below provide an overview of the installation process. See [Verified Features](#) below for a summary of what's been verified to work in the test environments where this has been used.

- Open a shell session (bash recommended) on the host on which (for single AIO deployment) or from which (for peer-test deployment) you want to install Acumos, and clone the system-integration repo:

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

- In the system-integration/AIO folder
 - Customize the acumos-env.sh script per your environment's needs, e.g. specify any proxy settings required, or select specific component ports other than the default, etc

- * If you are redeploying/restarting the platform, you can preserve the current database and any models you have onboarded, by setting the ACUMOS_CDS_PREVIOUS_VERSION environment variable in acumos-env.sh to the same value as the ACUMOS_CDS_VERSION variable, as shown below:

```
export ACUMOS_CDS_PREVIOUS_VERSION=1.16
export ACUMOS_CDS_VERSION=1.16
```

- * The script will preserve an existing database and all the related credentials (MariaDB, Nexus, CDS, ...) during the deployment, if the ACUMOS_CDS_PREVIOUS_VERSION variable is set. This will also be supported for database upgrade in a coming version (the capability is developed, but not fully tested).
- If you are deploying a single AIO instance, run the following command, selecting docker or kubernetes as the target environment. Further instructions for running the script are included at the top of the script.

```
bash oneclick_deploy.sh <docker|k8s>
```

- If you are deploying two Acumos AIO instances as peers, run the following command (NOTE: “under the hood”, this uses oneclick_deploy.sh):

```
bash peer-test.sh <host1> <user1> <under1> <host2> <user2> <under2> [models]
```

- For the above commands specify:
 - * “docker” to install all components other than mariadb and the docker-engine under docker-ce
 - * “k8s” to install all components other than mariadb under kubernetes
 - * “<host1>”/”<user1>” as hostname and user account to install under for the first peer, and “<host2>”/”<user2>” similarly for the second peer
 - * optionally, for “[models]” specify a folder with Acumos models to be onboarded under a “test” user account (an admin user, automatically created by the peer-test.sh script)
- The deployment will take 5-20 minutes depending upon whether you have run this command before and thus docker has already downloaded the Acumos docker images. That will speed up subsequent re-deploys.
- When deployment is complete, you should see a message similar to this, stating the URL for the Portal:

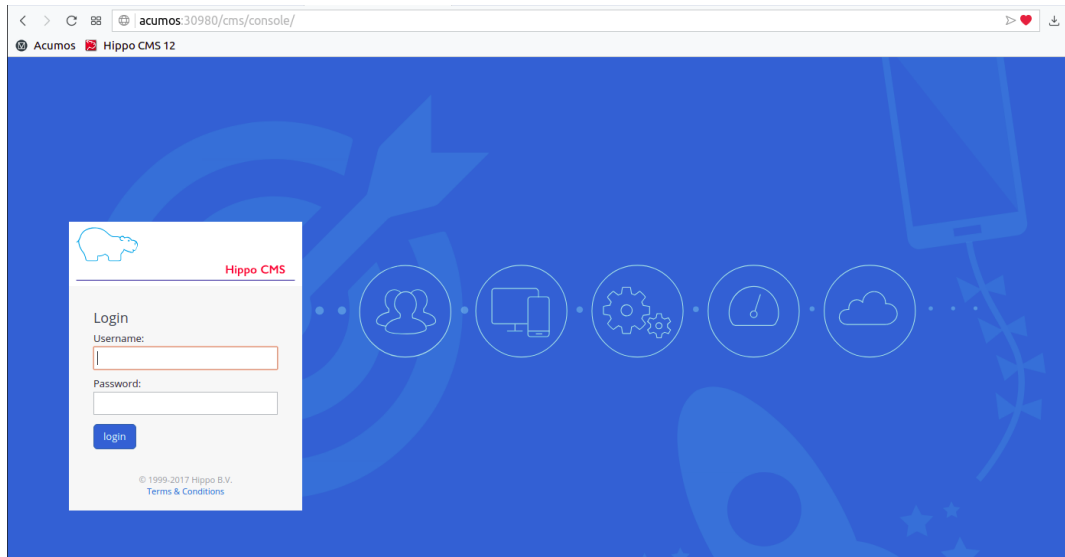
```

$ [f:1534342247507 == AnOVA11]
$ log "Deploy is complete. You can access the portal at https://acumos:30443 (assuming you have added that hostname to your hosts file)"
$ set +x
main:535 (Wed Aug 15 09:10:47 CDT 2018) Deploy is complete. You can access the portal at https://acumos:30443 (assuming you have added that hostname to your hosts file)

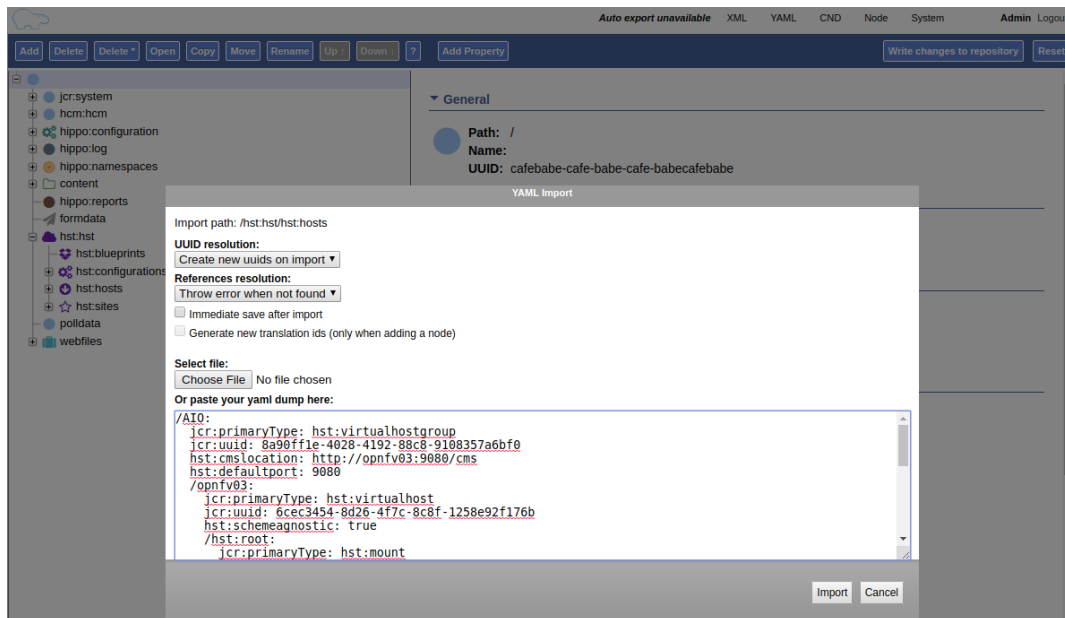
```

- To enable all Portal content, you will need to complete one manual setup action for the Hippo CMS. Note this action is not required to use the Portal, just to ensure that all Portal-displayed info is presented correctly. Follow these steps on each AIO host (replacing “<hostname>” with the applicable name for the host):

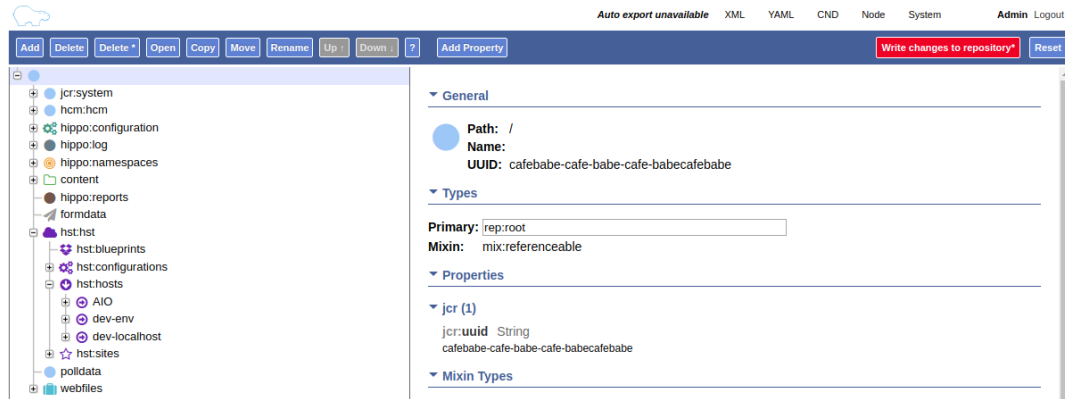
- Login to the Hippo CMS console as “admin/admin”, at http://<hostname>:<ACUMOS_CMS_PORT>/cms/console, where ACUMOS_CMS_PORT is per `acumos-env.sh`; for the default, the address is `acumos:30980/cms/console`



- On the host where you installed the AIO Platform, login to the account you used when installing, and copy the contents of file `aio-cms-host.yaml`
- On the CMS UI at the left, click the + at `hst:hst` and then right-click `hst:hosts`, and select “Yaml Import”. In the resulting dialog, paste the copied contents of file `aio-cms-host.yaml`



- When the dialog closes, you should be able to see a new node “AIO” under `hst:hosts`. You can now your changes by pressing the `Write changes to repository` button in the upper right.



- Update your local workstation’s hosts file so the portal domain name “<hostname>” will resolve on your workstation. Add a line: <ip address of your AIO host> <hostname>. Note: on Ubuntu, the hosts file is at /etc/hosts. The example below is from an Ubuntu laptop with the AIO instance running in a Virtual Box environment.



- Create an admin user: the oneclick_deploy.sh script **does not** create a default user. However, you can use the create-user.sh script to create an “Admin” user for the platform. The create-user.sh script is located in the same directory as the oneclick-deploy.sh script. Usage instructions are included at the top of the create-user.sh script. Below is an example of how to create an admin user:

```
$ bash create-user.sh admin Admin123 Admin Admin admin@admin.net Admin
...(lots of output)
$ User creation is complete
```

- You should now be able to browse to <https://<hostname>>, and
- register new user accounts, etc
- if you deployed a peer-test set of Acumos portals, log into the “test” user account with password per peer-test.sh (see line with “bash create-user.sh”)
- If you get a browser warning, just accept the self-signed cert and proceed.

Updating Configuration and Components

As described in *Install Process* and *Stopping, Restarting, and Reinstalling*, you can redeploy the whole platform without losing current data (e.g. users and models), by changing the values in acumos-env.sh (as updated by an earlier install process) as needed, leaving the rest as-is, and re-executing the deployment command you used for the previous deployment.

However, this process is not guaranteed to be fail-proof, and if you are concerned about the ability to recover database items that may be lost, it is recommended that you first backup the databases or export data from them. Some tools have been developed for this, e.g.

- `dump-model.sh`: this tool is intended to enable export of all artifacts related to one or more models by solution/revision

The following types of redeployment are regularly tested as part of the AIO toolset development:

- updating the configuration

- values in `acumos-env.sh`, or values in the component templates etc, can be modified and re-applied by redeploying the components. Note however that some values may not work with previous data, as the related components are not redeployed/reconfigured. For example, the following values should not be changed without a clean redeploy:
 - * domain name of the Acumos platform
 - `ACUMOS_DOMAIN`
 - * CDS settings
 - `ACUMOS_CDS_PASSWORD`
 - * Nexus settings
 - `ACUMOS_NEXUS_ADMIN_USERNAME`
 - `ACUMOS_NEXUS_ADMIN_PASSWORD`
 - `ACUMOS_RO_USER`
 - `ACUMOS_RO_USER_PASSWORD`
 - `ACUMOS_RW_USER`
 - `ACUMOS_RW_USER_PASSWORD`
 - * server certificate credentials
 - `ACUMOS_KEYPASS`
- upgrading a specific component or set of components
 - components can be upgraded, e.g. for testing or to move to a new [release assembly](#). However, ensure that you have addressed any component template changes, as described by the release notes for the new component versions.
- upgrading the CDS database version
 - CDS version changes sometimes result in a new version of the CDS database schema. Version upgrades are supported by the AIO toolset, given that there is an available mysql upgrade script in the common-dataservice repo. Scripts are provided for an incremental update only; see the [CDS github mirror](#) for examples of the available scripts.

Stopping, Restarting, and Reinstalling

If you deployed under docker, you can stop all the Acumos components (e.g. to suspend/shutdown your host) without losing their databases via the command:

```
sudo bash docker-compose.sh stop
```

Restart the services later using the following command (note it may take a few minutes for all to be active):

```
sudo bash docker-compose.sh restart
```

If you deployed under kubernetes, you can also restart the whole platform, by the following command, as long as the generated values in `acumos-env.sh` (e.g. passwords for MariaDB, CDS, Nexus, ...) have not been changed:

```
bash oneclick_deploy.sh k8s
```

If you deployed under kubernetes, you can also restart a specific component by the name of the deployment. As in the example below, you can use the `kubectl` command to get the deployment names. Note that:

- the deployment templates as updated by oneclick-deploy.sh (substituting variables as needed) are in the sub-folder deploy/kubernetes/deployment
- the elasticsearch, logstash, and kibana deployments are all defined in file elk-deployment.yaml, so when recreating any of these, refer to that file in the `kubectl create -f` command

```
$ kubectl get deployments -n acumos
NAME                DESIRED    CURRENT    UP-TO-DATE    AVAILABLE    AGE
azure-client        1          1          1             1            5d
cds                  1          1          1             1            5d
cms                  1          1          1             1            5d
docker              1          1          1             1            5d
dsce                 1          1          1             1            5d
elasticsearch        1          1          1             1            5d
federation           1          1          1             1            5d
filebeat             1          1          1             1            5d
kibana               1          1          1             1            5d
kong                 1          1          1             1            5d
kubernetes-client    1          1          1             1            3m
logstash             1          1          1             1            5d
metricbeat           1          1          1             1            5d
msg                  1          1          1             1            5d
nexus                1          1          1             1            5d
onboarding           1          1          1             1            5d
portal-be            1          1          1             1            5d
portal-fe            1          1          1             1            5d

$ kubectl delete deployment -n acumos kubernetes-client
deployment.extensions "kubernetes-client" deleted
$ kubectl create -f deploy/kubernetes/deployment/kubernetes-client-deployment.yaml
deployment.apps "kubernetes-client" created
```

You can clean the installation (including all data) via:

```
bash clean.sh
```

Verified Features

new user registration

The following Acumos platform workflows and related features have been verified as working so far. This list will be updated as more workflows are verified.

The following features are verified as part of the process of deployment or post-deployment through the referenced test scripts:

- high-level deployment scenarios under which specific tests are executed
 - Deploy with all-new components
 - * leave `ACUMOS_CDS_PREVIOUS_VERSION` as the default (blank) and execute deployment
 - Redeploy with pre-existing mariadb, nexus, etc
 - * set `ACUMOS_CDS_PREVIOUS_VERSION` to the same value as `ACUMOS_CDS_VERSION` in `acumos-env.sh` and execute deployment
 - Redeploy with upgraded database

- * in `acumos-env.sh`, set `ACUMOS_CDS_PREVIOUS_VERSION` to the value of `ACUMOS_CDS_VERSION` as used in the last deployment, and increment `ACUMOS_CDS_VERSION` to the next version of the CDS, and execute deployment
- new user registration: `create-user.sh`
 - Create user, via Portal API `/api/users/register`
 - Finding role by name, via CDS API `/ccds/role`
 - Create role by name, via CDS API `/ccds/role`
 - Assign role to user, via `/ccds/user`
 - Get role for user, via CDS API `/ccds/user/$userId/role/$roleId`
 - Get user account details, via CDS API `/ccds/user/$userId`
- ‘self’ peer creation: `oneclick-deploy.sh`
 - Create ‘self’ peer, via CDS API `/ccds/peer`
- remote peer creation: `create-peer.sh`
 - Get `userId` of user, via CDS API `/ccds/user`
 - Create peer, via CDS API `/ccds/peer`
 - Apply new truststore entry by restarting the Federation service
 - Subscribe to all solution types at peer, via CDS API `/ccds/peer/sub`
 - get list of solutions, via Federation API `/solutions`
- model onboarding via command line: `bootstrap-models.sh` and `onboard-model.sh`
 - User authentication and JWT token retrieval, via Onboarding API `/onboarding-app/v2/auth`
 - Model onboarding, via Onboarding API `/onboarding-app/v2/models`
 - Onboarding of normal models and “Datasource” type models

The following manual tests are regularly verified as part of AIO testing:

- user login
- user signup
- model onboarding via web
- model sharing with another user
- model publication to company marketplace
- model publication to public marketplace
- federated peer relationship creation via portal
- federated subscription to public marketplace models
- verification of subscribed model presence in public marketplace
- creation of composite solution
- addition of probe to composite solution
- setting Datasource model Category “Data Sources” and Toolkit “Data Broker”
- creation of composite solution with Datasource
- model deployment in private kubernetes (“deploy to local”)

- simple model
- composite model
- composite model with Probe
- composite model with Probe and Data Broker

Notes on Verified Features

User registration and login

A test script to automate user account creation and role assignment has been included in this repo. See [create-user.sh](#) for info and usage. For an example of this script in use, see [Federation](#).

Model onboarding via command line

Currently this is verified by posting a model package to the onboarding API, as toolkit clients will do when installed. Two scripts are used for this:

- [bootstrap-models.sh](#)
 - onboard all models in a folder; models are in subfolders and include the three essential artifacts, as generated by an onboarding client, or downloaded earlier from an Acumos portal
 - * model.zip
 - * metadata.json
 - * a .proto file, either model.proto (normal models) or default.proto (Datasource type models)
- [onboard-model.sh](#)
 - onboard a specific model (a folder with the files as describe above)

Federation

oneclick_deploy.sh will automatically create a “self” peer as required by the federation-gateway.

If you want to deploy two Acumos AIO instances to test federation, see these scripts for info and usage:

- peer-test.sh: installs and peers two Acumos AIO instances, on two hosts, and optionally uploads model packages via curl.
- create-peer.sh: used by peer-test.sh. You can call this script directly to add a peer to an existing Acumos platform.

You can also manually create a federated peer:

- If you have not created an admin user, run create-user.sh as above to create one.
- Login to the portal as the admin user
- Under the “SITE ADMIN” page, select “Add Peer”, enter these values, and select “Done”:
 - Peer Name: FQDN of the peer
 - Server FQDN: DNS-resolvable FQDN of the peer
 - API Url: [http://<FQDN of the peer>:<federation-gateway port from acumos-env.sh>](#)

- Peer Admin Email: any valid email address
- Verify that the peer relationship was setup via executing these commands on the AIO host
- `source acumos-env.sh`
- `curl -vk -cert certs/acumos.crt -key certs/acumos.key <API Url as above>`
- You should see details of the HTTPS connection followed by

```
{ "error": null, "message": "available public solution for given filter",
  "content": [...] }
```

- This indicates that the request for “solutions” was accepted. “...” will either be “” (no solutions) or a JSON blob with the solution details.

Features Pending Verification

- model onboarding via web
- model private sharing with user
- model launch
- design studio

3.1.4 Logs Location

Logs are easily accessible on the AIO host in the `/var/acumos` directory.

```
$ ls /var/acumos/logs
$ acumos-azure-client ccds ds-compositionengine federation-gateway
↪kubernetes-client microservice-generation on-boarding portal-be portal-
↪fe

$ ls /var/acumos/logs/portal-be
$ access.log audit.log debug.log error.log
```

These host folders are mapped to persistent volumes exposed to the components.

3.1.5 Additional Notes

The scripts etc in this repo install Acumos with a default set of values for key environment variables. See `acumos-env.sh` for these defaults. You should be able to modify any explicit value (not variables) defined there, but some additional steps may be needed for the installed platform to work with the updated values. For example:

- To use a non-default domain name for the acumos AIO server (default: `acumos`), change `ACUMOS_DOMAIN` in `acumos-env.sh`, and use the chosen domain name in the “Install Process” above, in place of “`acumos`”.
- You can install multiple Acumos platforms (e.g. to test federation), just be sure to give each a unique domain name as above.
- The latest verified Acumos platform docker images are specified in `acumos-env.sh`. This script will be updated as new versions are released to the staging or release registries of the Acumos.org nexus server.

3.2 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.

3.2.1 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 describes how to use the Acumos Elastic Stack (formerly known as the ELK Stack).

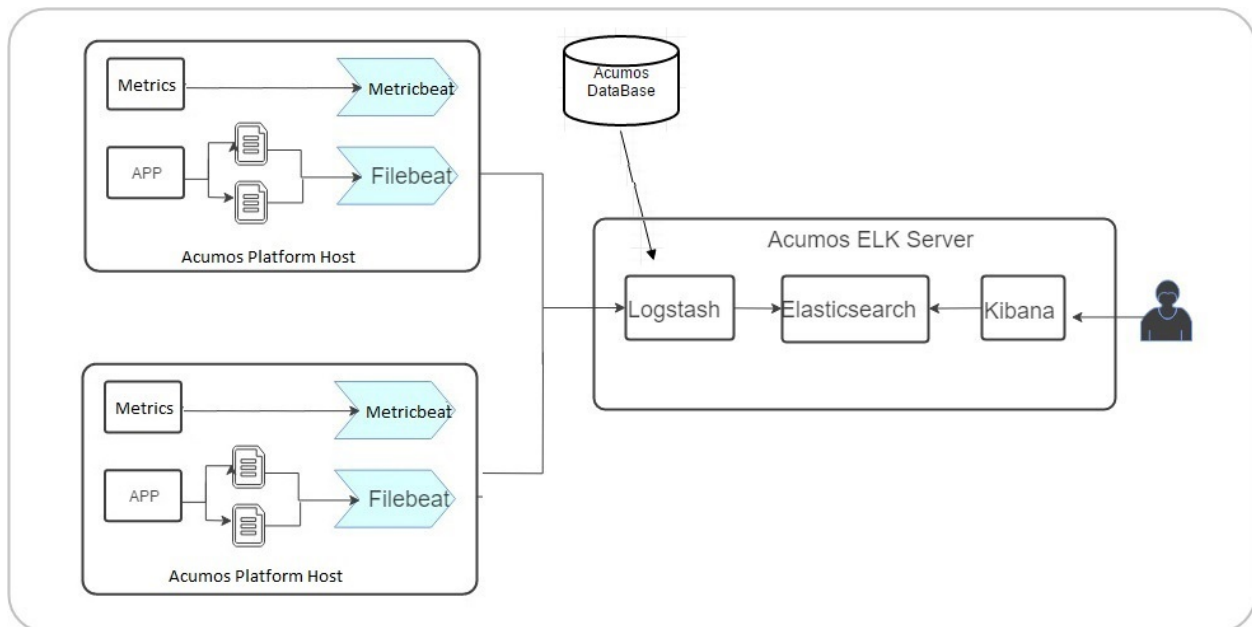
Target Users

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 if no volumes created earlier.If acumos-esdata volume already exist on host machine then skip this step.

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

3. The acumos-elk-env.sh file is the environment file for ELK stack. Update variables ELASTIC-SEARCH_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

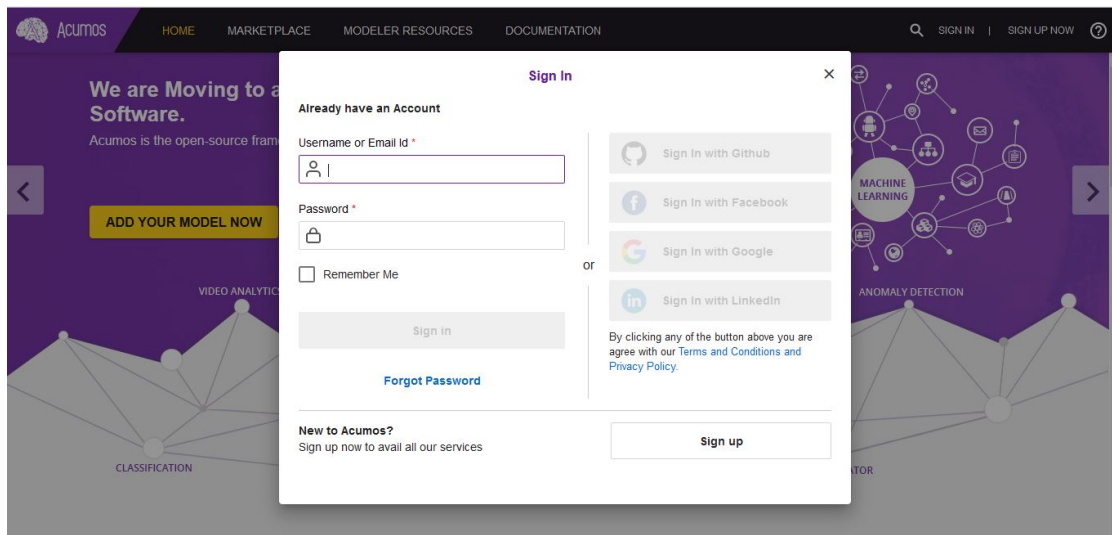
```

Elastic Stack UI Tour

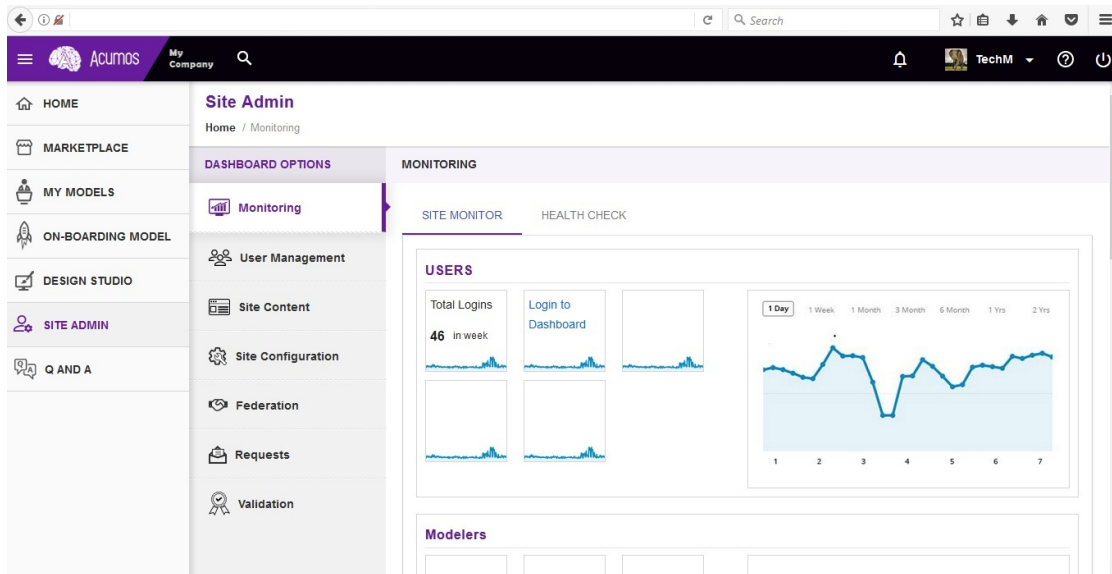
According to the [Kibana website](#), Kibana is an open source analytics and visualization platform designed to work with Elasticsearch. You use Kibana to search, view, and interact with data stored in Elasticsearch indices. You can easily perform advanced data analysis and visualize your data in a variety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

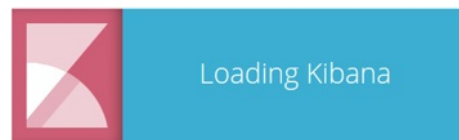
Site admins have access to Elastic Stack's Kibana Dashboard. Login to the dashboard:



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



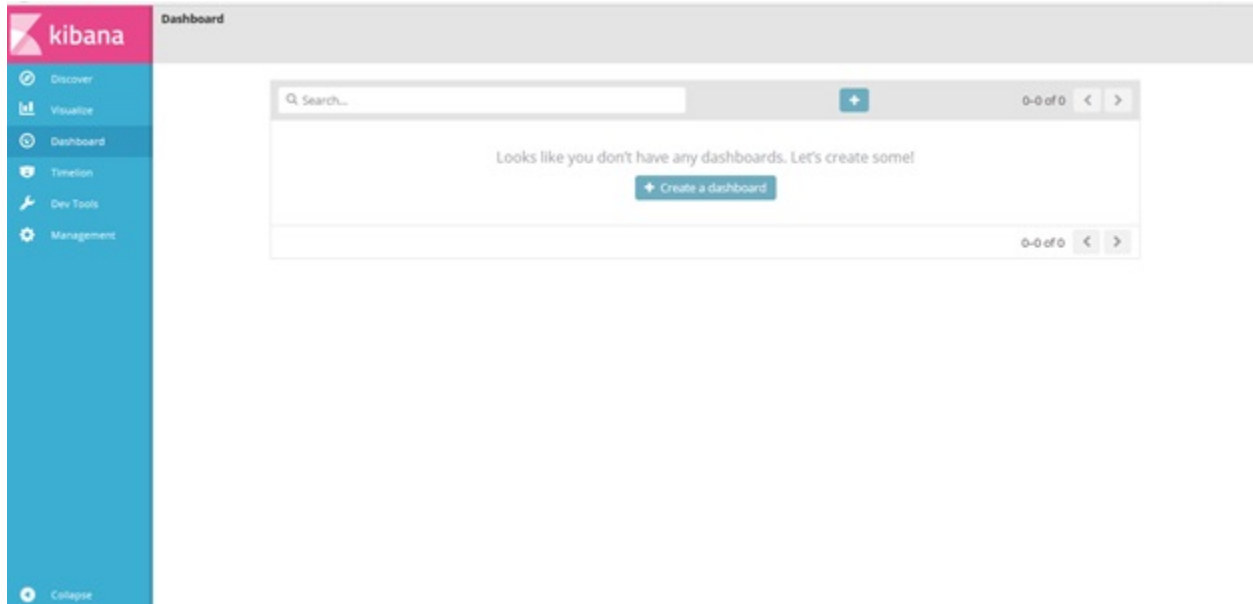
Redirects to Loading Kibana visualization platform



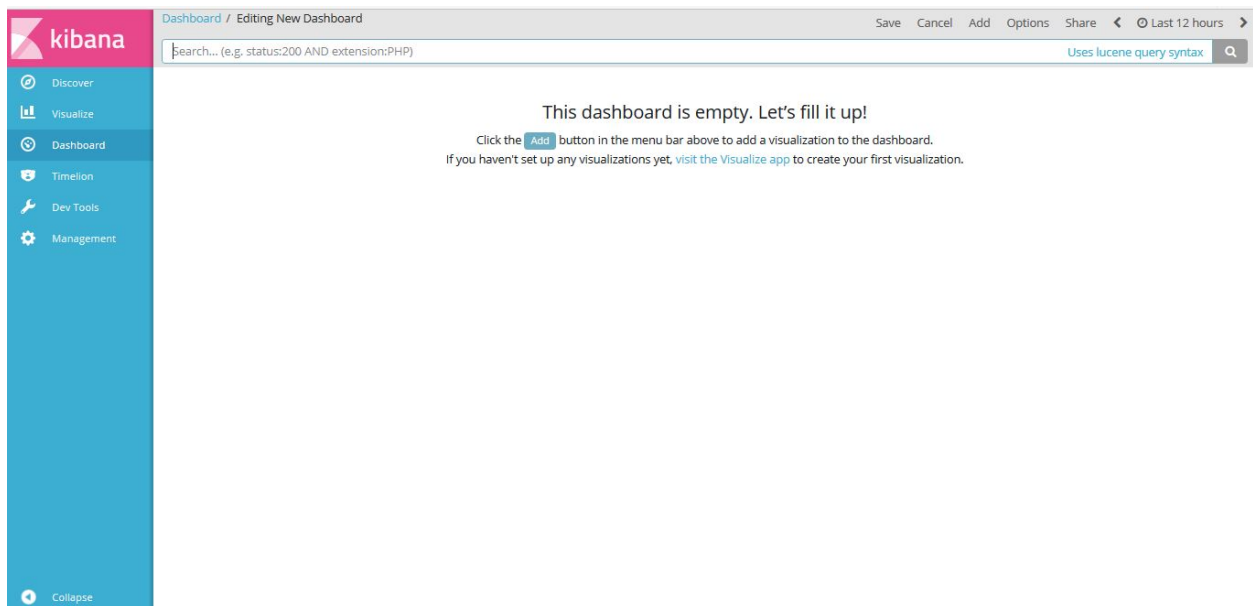
Acumos Kibana Dashboard Creation

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

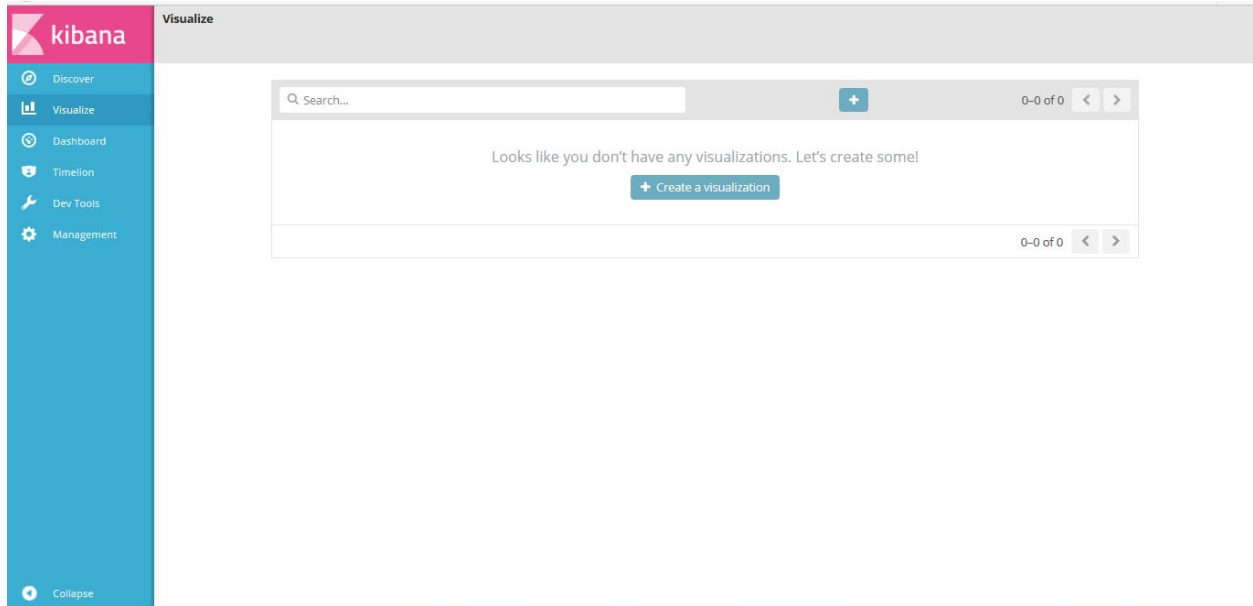
To create dashboard click on Create a dashboard or On plus sign show in the search bar.



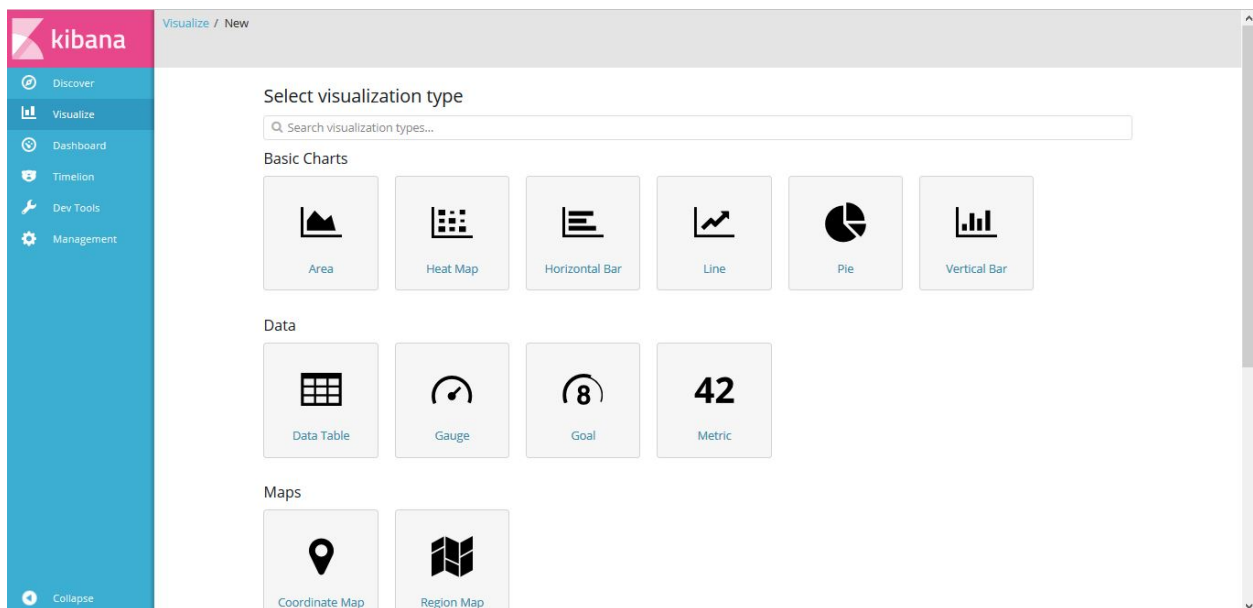
click on Visit the Visualize app



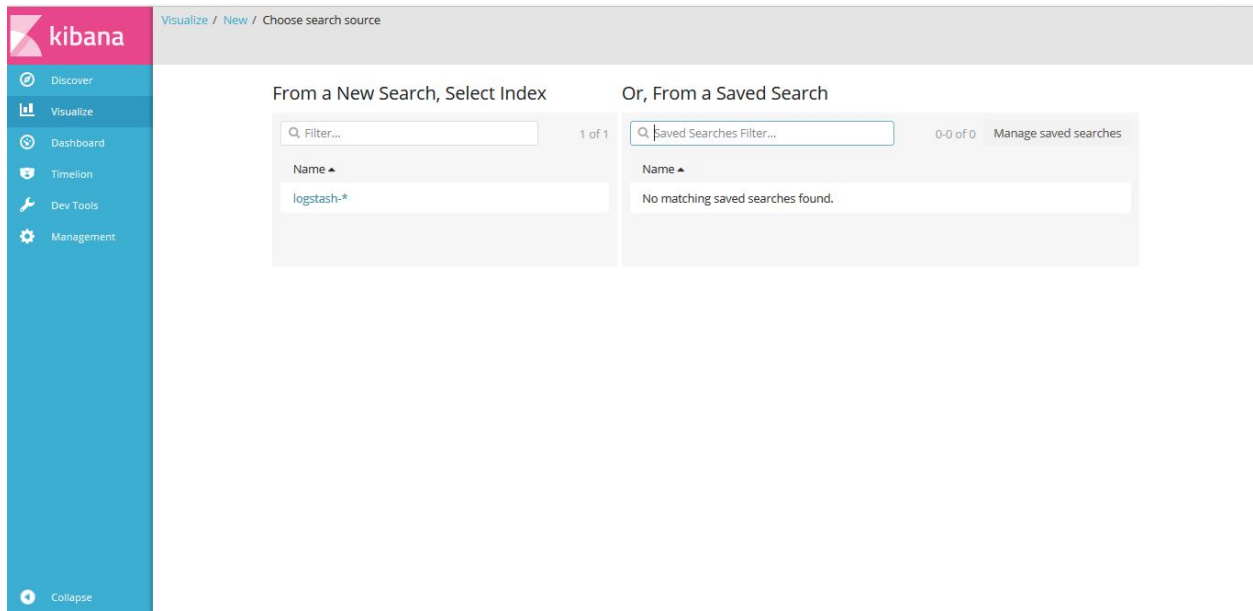
click on “Create a visualization” or “+”(i.e Plus sign) show in the search bar.



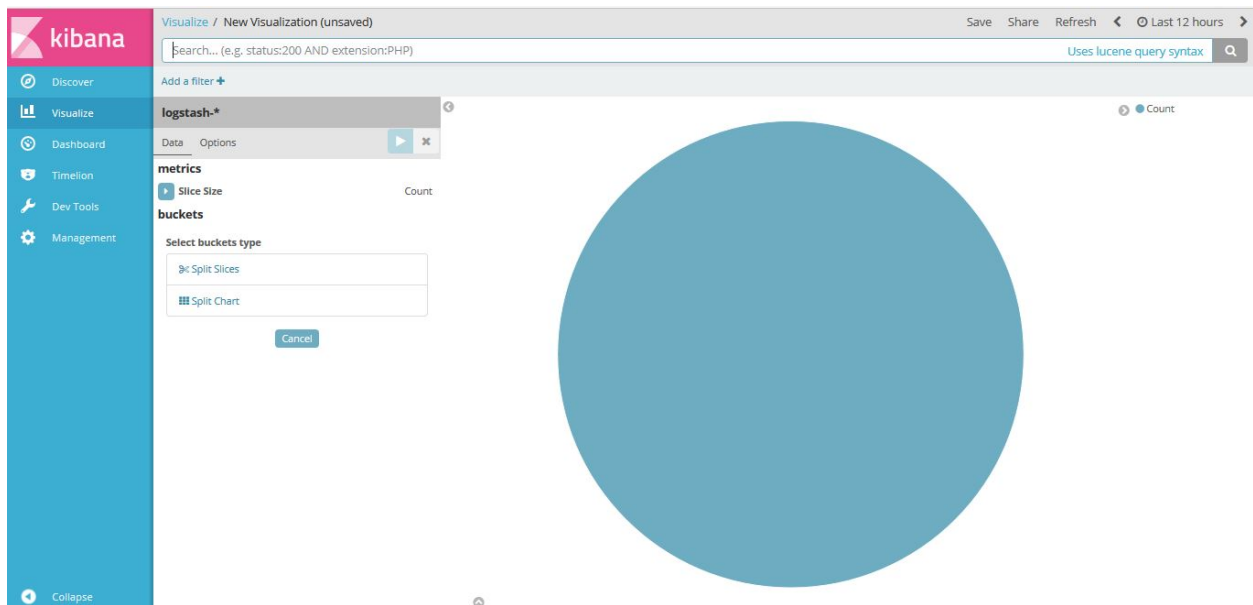
Select visualization type. For example click on “Pie”.



Choose search source as `logstash-*`

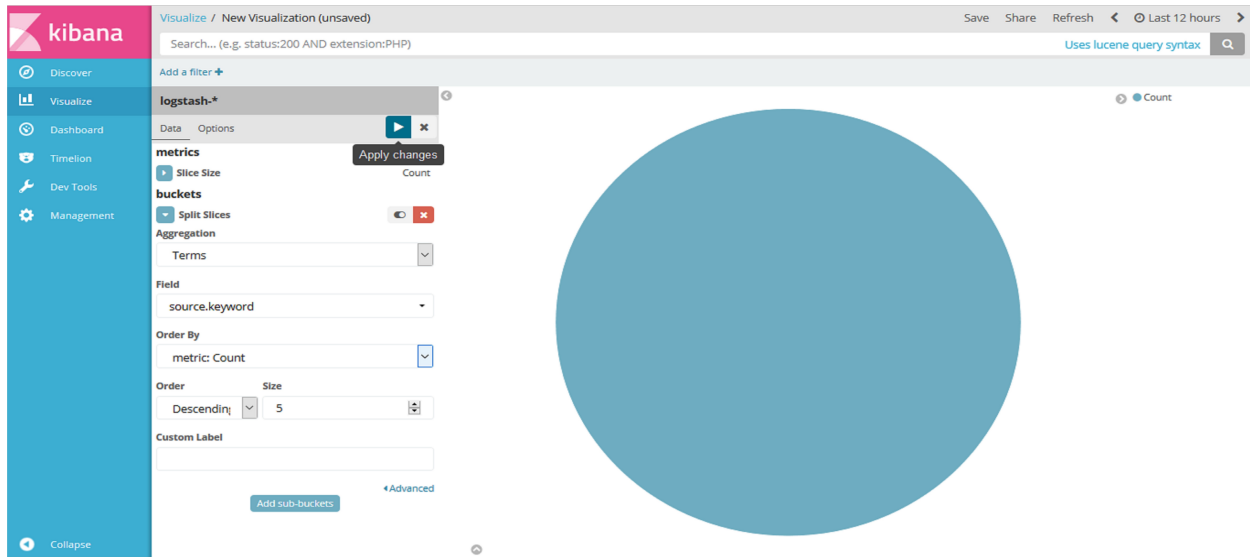


Click on Split Slices

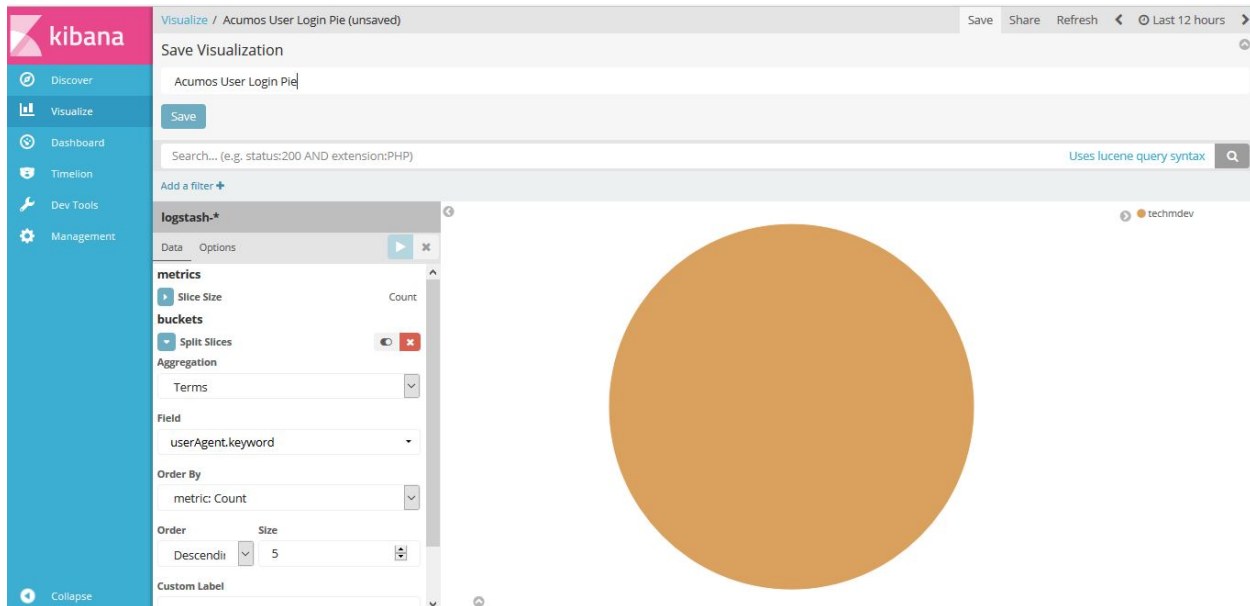


Select Aggregation as “Terms” and Field as “userAgent.keyword”, Click on “Apply changes”

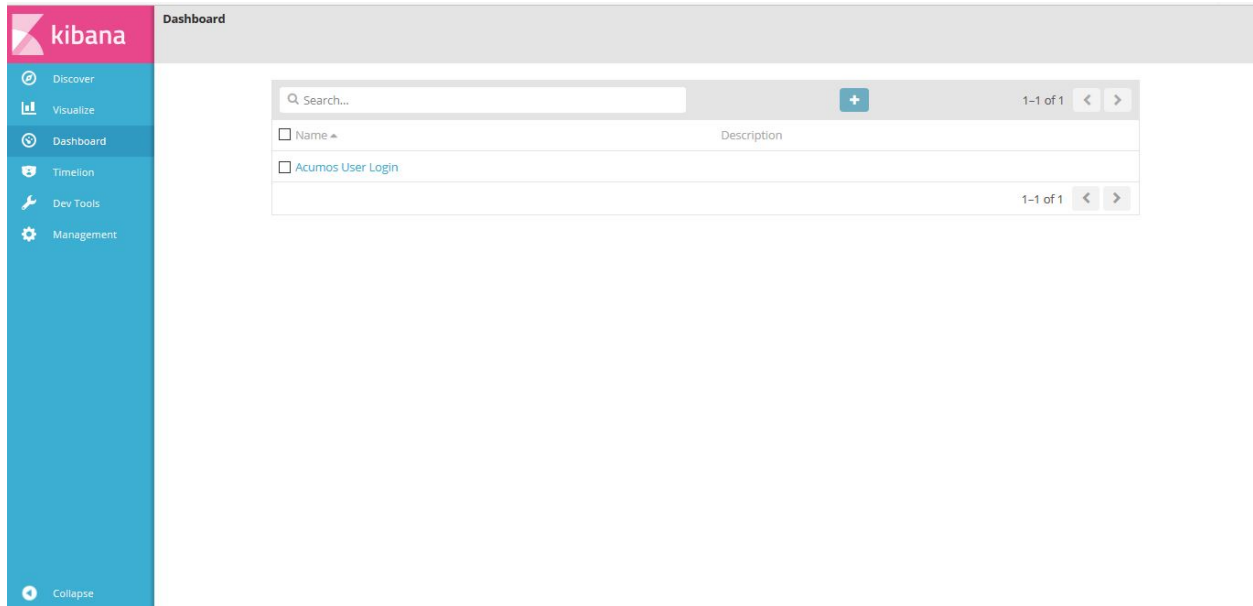
Note: Elasticsearch aggregations are to extract and process your data.



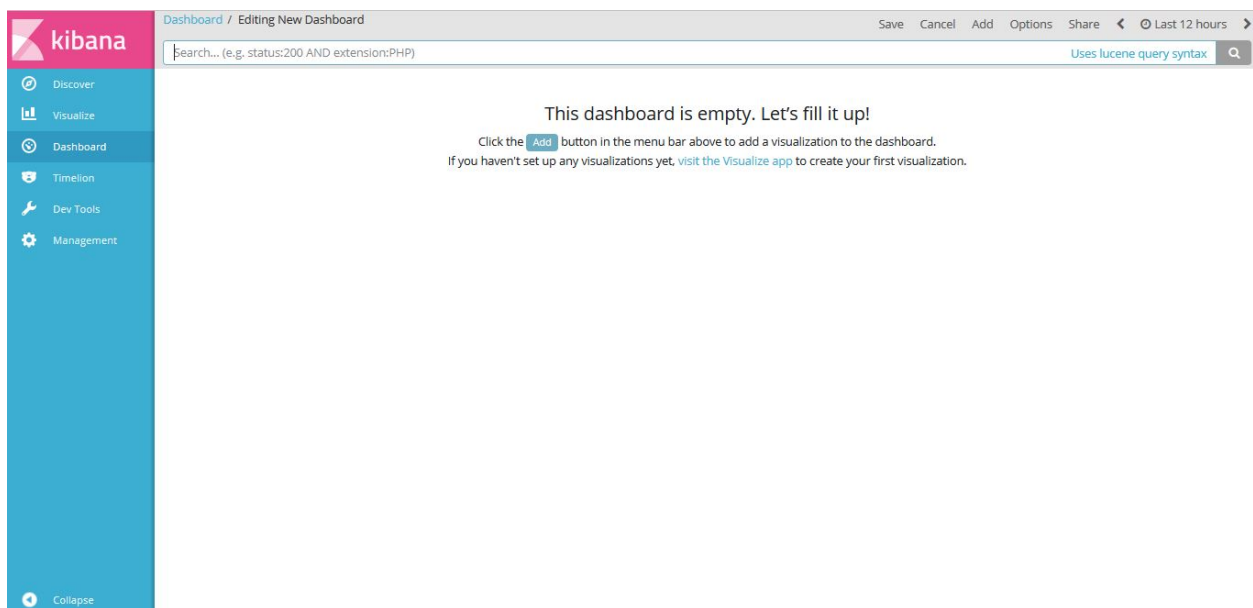
To save this chart click on “Save”, Enter a name appropriate name. For example “Acumos User Login”.



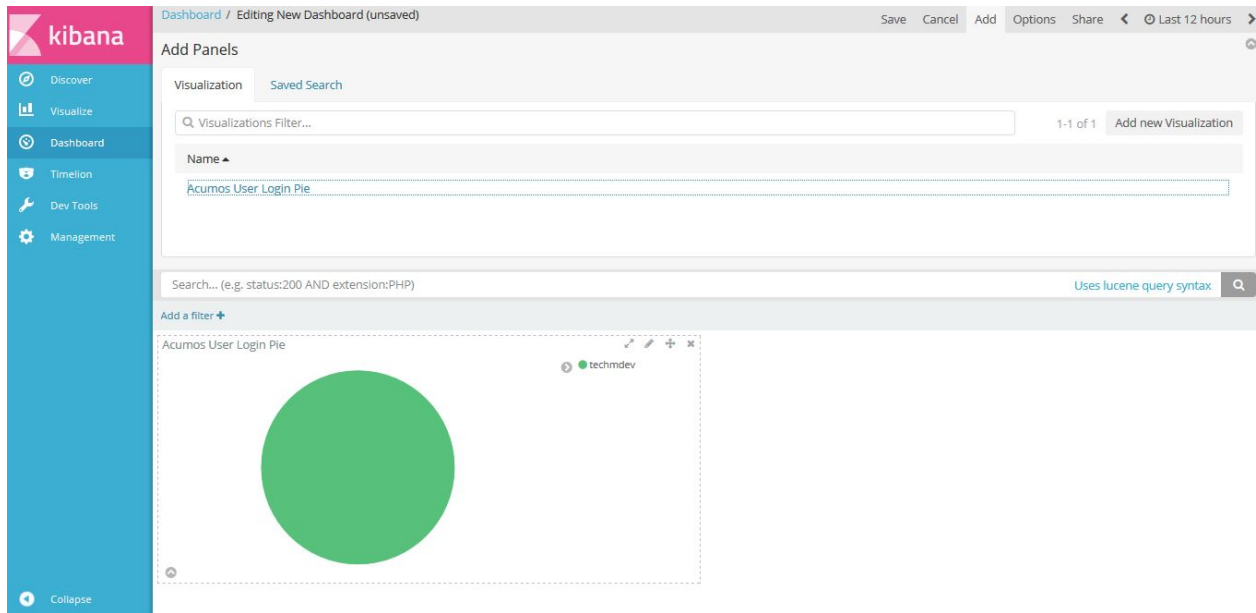
Click on “Dashboard”, On the below screen visualization namely “Acumos User Login” is appearing. For select this visualization click on “+” (i.e. plus sign) show in the search bar.



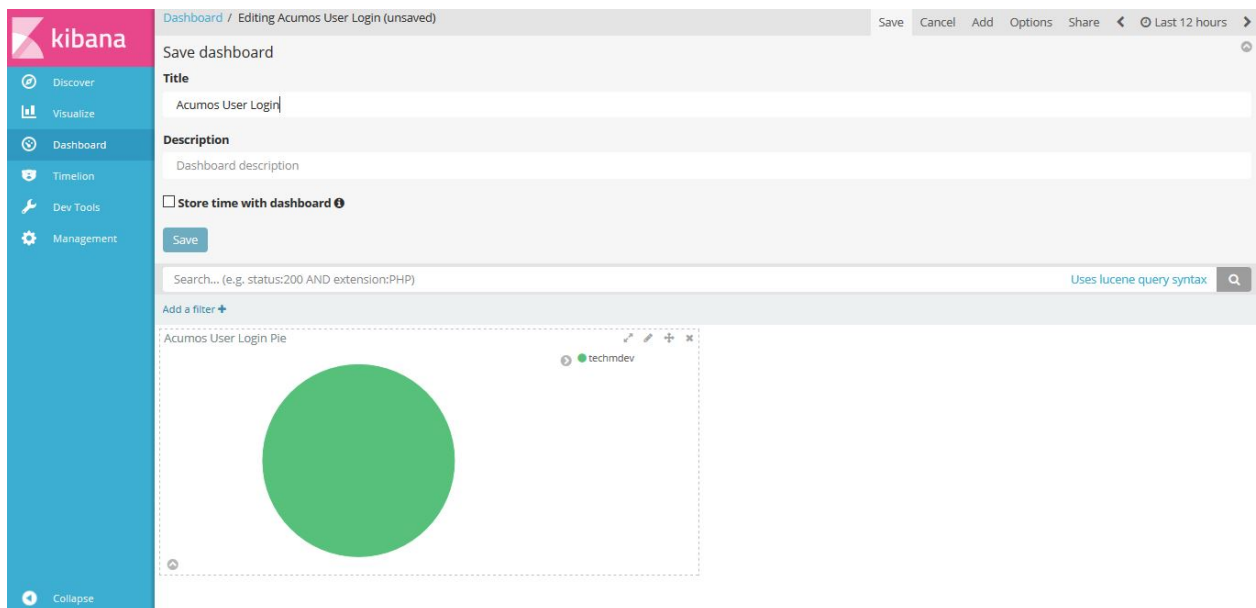
Click on “Add” button, to add the visualization.



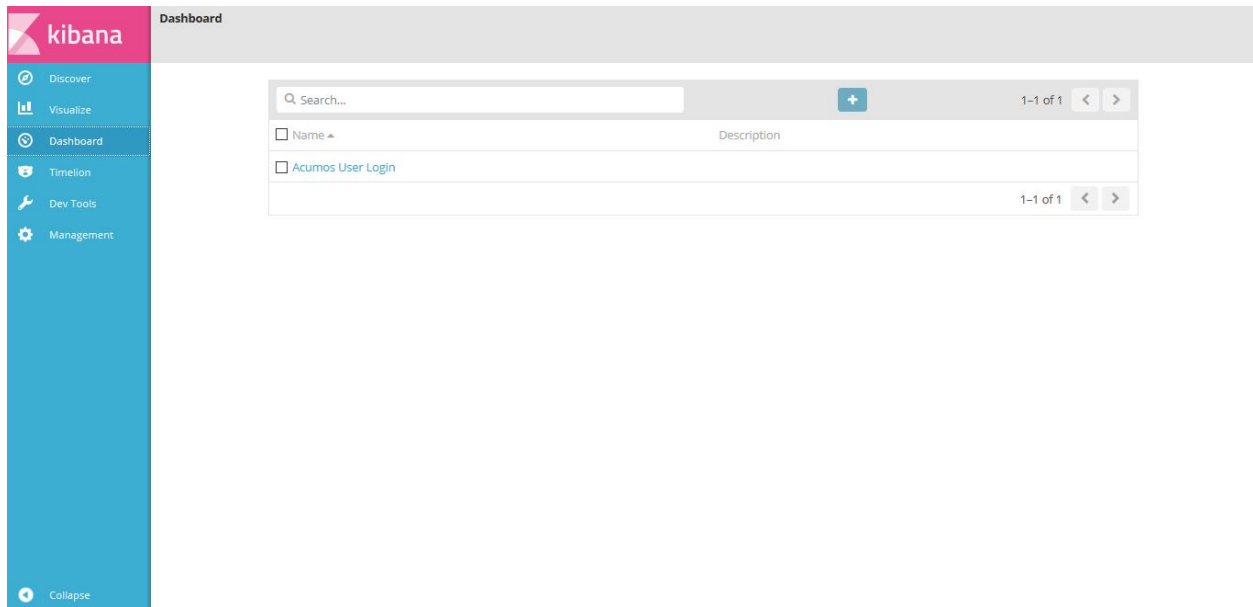
Select the visualization for example here we have visualization namely “Acumos User Login”.



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

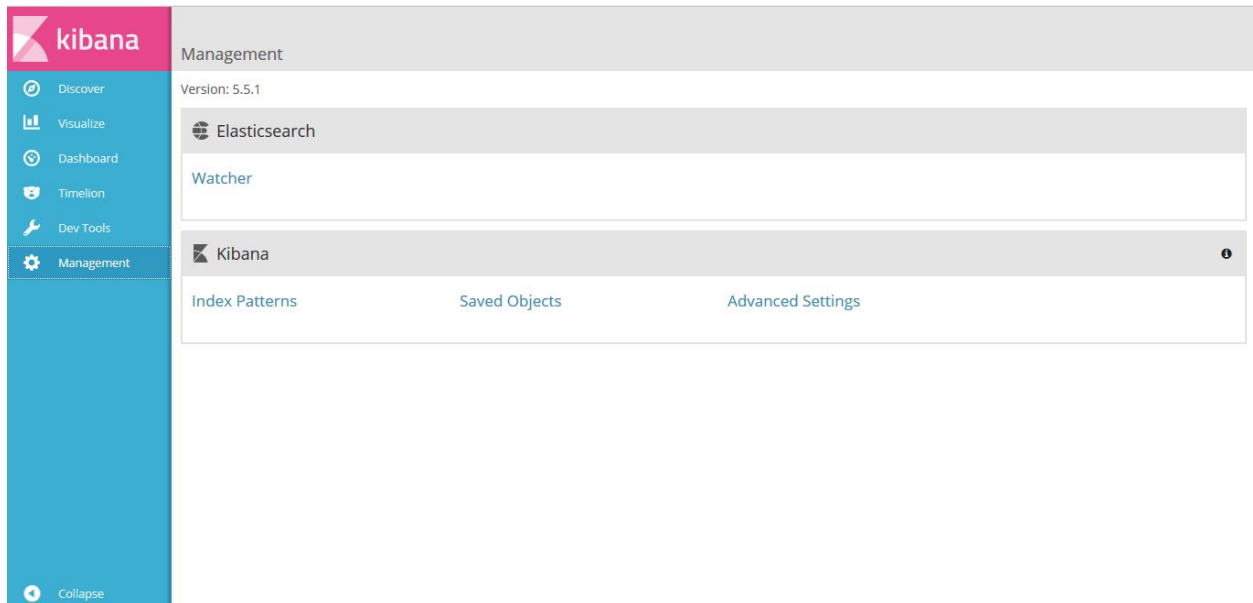


Click on “Dashboard”, On the below screen created dashboard can be viewed namely “Acumos User Login”.

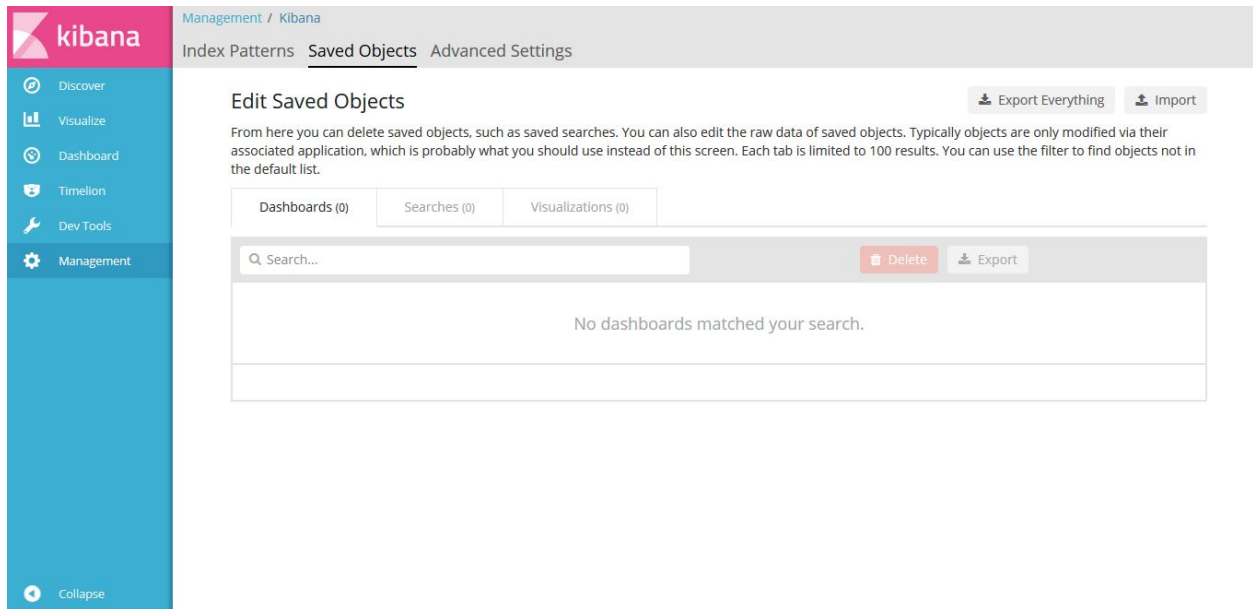


Acumos Kibana Dashboard Save

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



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



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

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

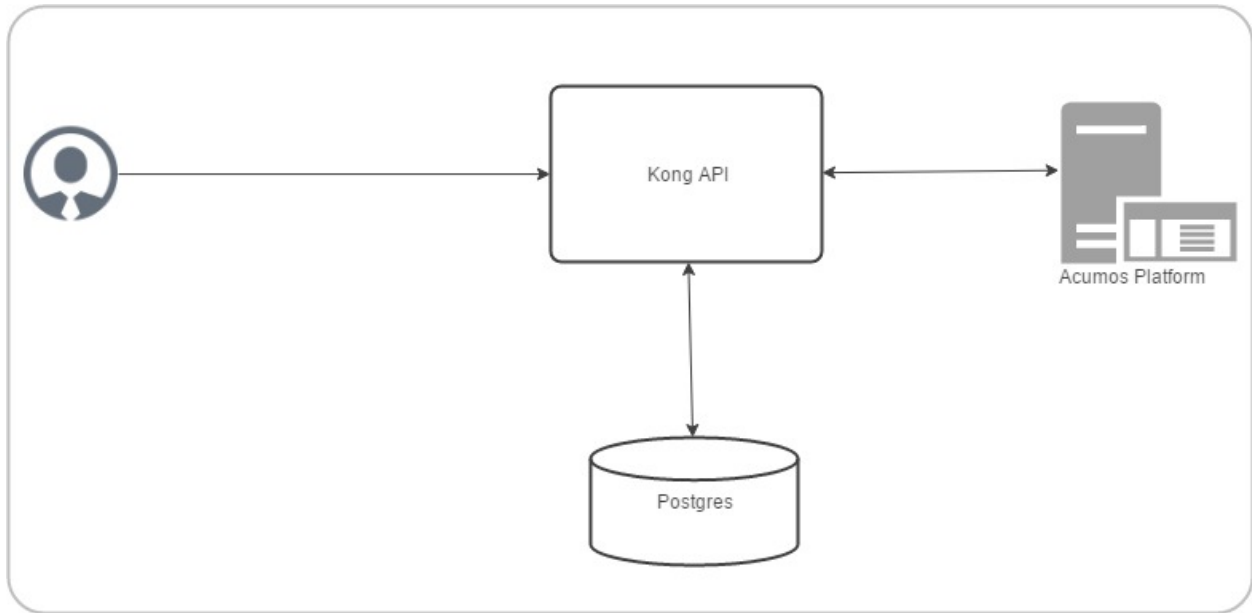
3.3 System Integration User Guide

3.3.1 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.

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-kubect1.env file to make changes related to latest assembly , database connection , credentials ,etc.

```
$ vi acumos-kubect1.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-kubect1.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 complete access to .pkcs12 and .jks file by making use of below command

```
chmod 777 acumosTrustStore.jks
chmod 777 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.

```
kubect1 -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'@'%';
```

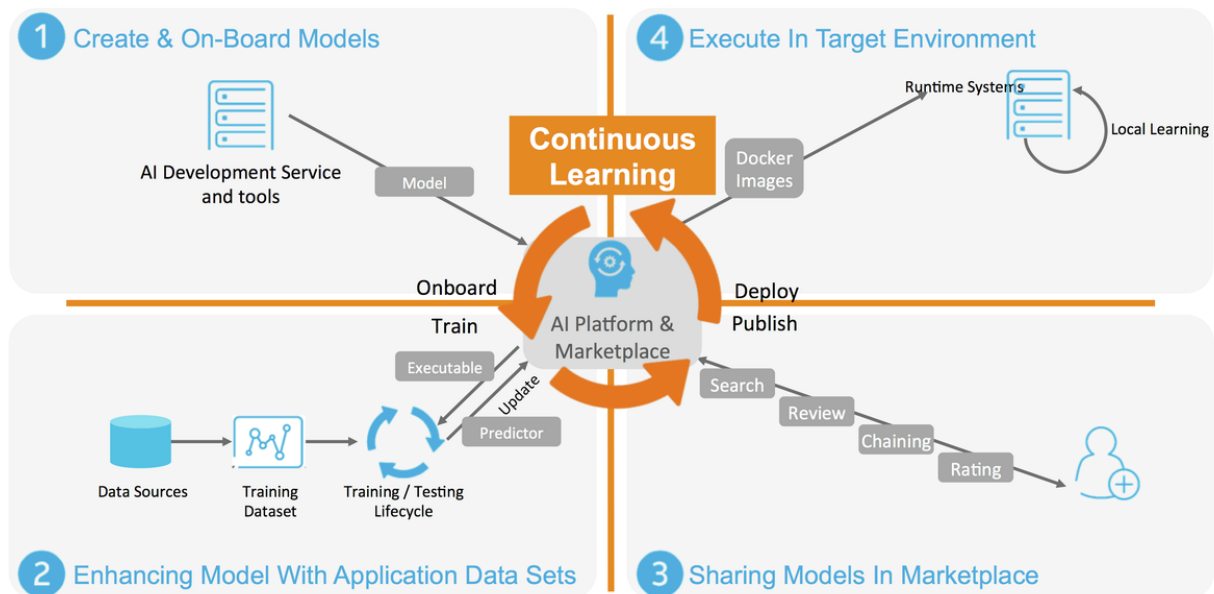
Execute the DDL and DML scripts for any database version that needs to be configured.

Contributors to the Acumos Platform Code

4.1 Platform Architecture

4.1.1 Architecture Guide

Introduction



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

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

Acumos is not tied to any specific modeling language or toolkit and not limited to any one run-time infrastructure or cloud service. Acumos creates an open source mechanism for packaging, sharing, licensing and deploying AI models in the form of portable, containerized microservices and publishes them in a shared, secure catalog. Using Acumos, data scientists can build abstract AI models, using their favorite or most appropriate 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 platform, based upon Kong
- various components that provide 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

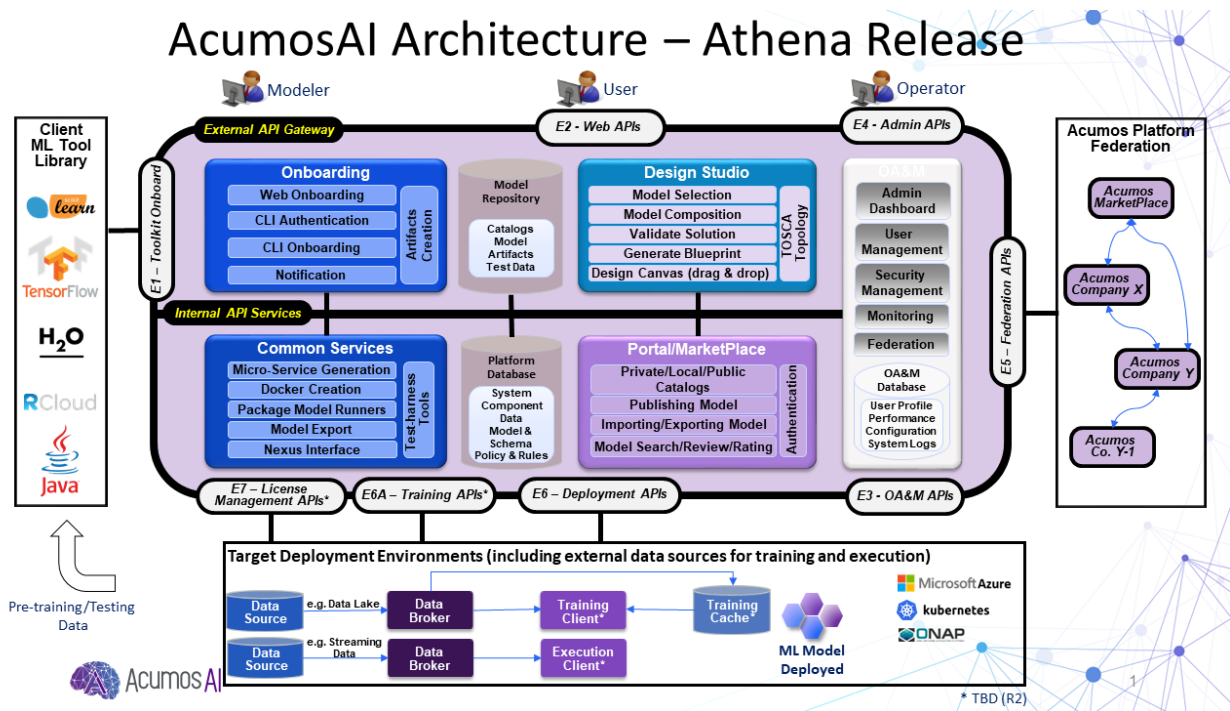
As described on the [Acumos.org website](https://acumos.org), Acumos AI is a platform and open source framework that makes it easy to build, share, and deploy AI apps, and operate the Acumos portals that enable those capabilities. Acumos standardizes the infrastructure stack and components required to 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 collection, storage, and visualization

Architecture

Architecture Overview



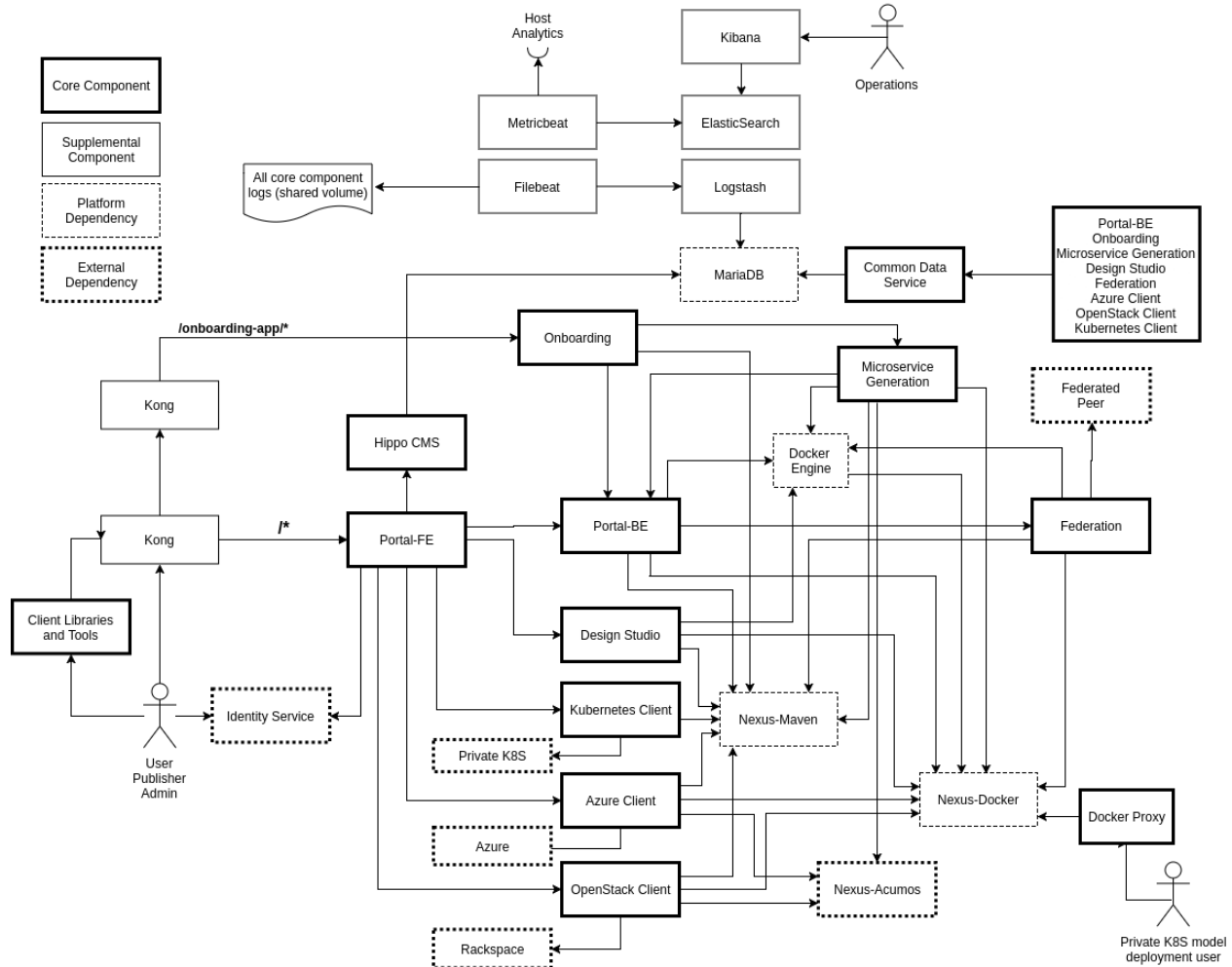
Component Interactions

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 Onboarding 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-Marketplace 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: [Onboarding 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 Onboarding app.

For more information: [Java Client Documentation](#).

Python Client

The Acumos Python 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 Onboarding 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 Onboarding 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 environments 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 kubernetes 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.

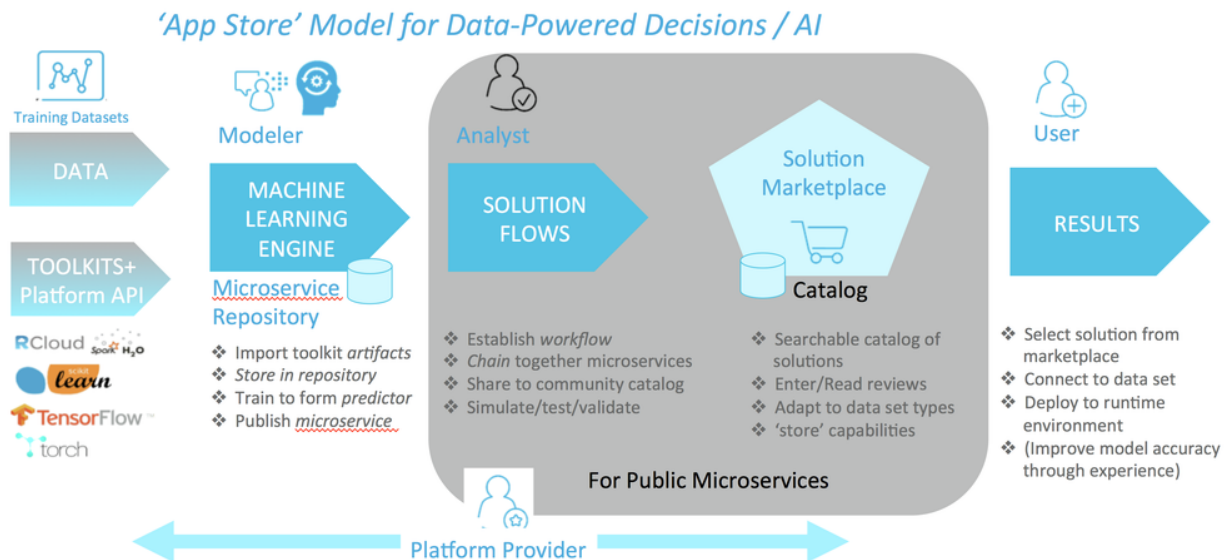
Platform Flow

User Journeys

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

Acumos Platform User Flow

Acumos Platform – User Flow



Acumos User Signup Flow

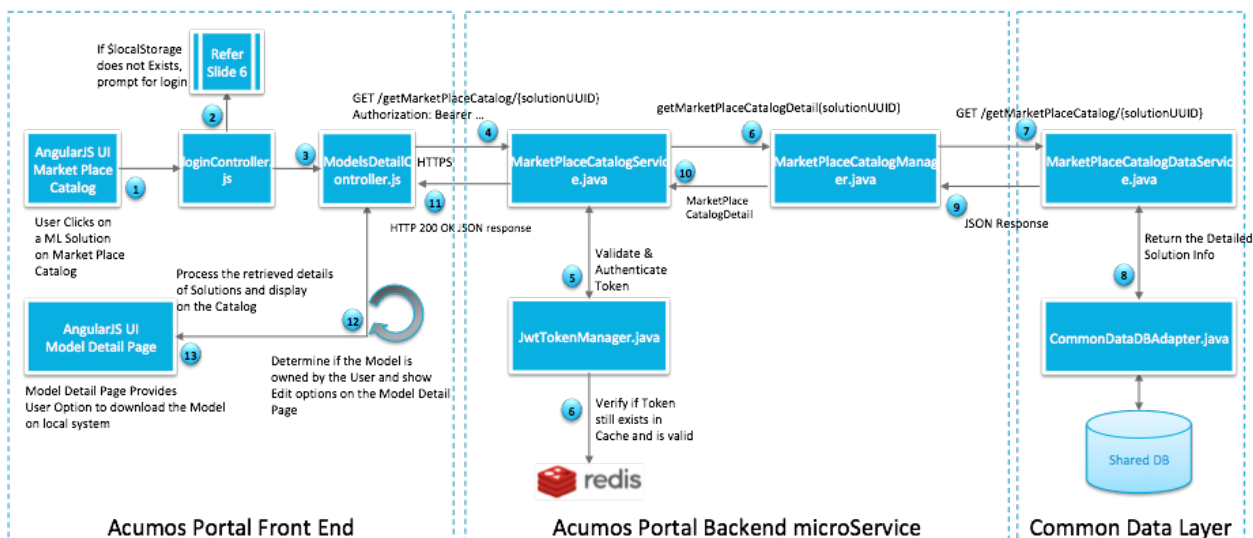
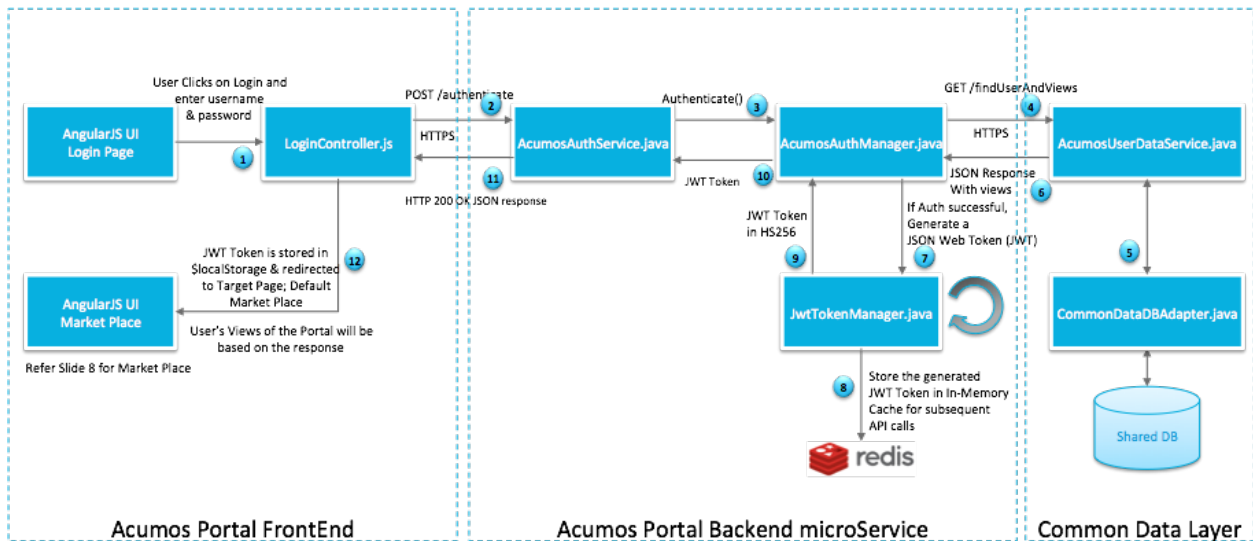
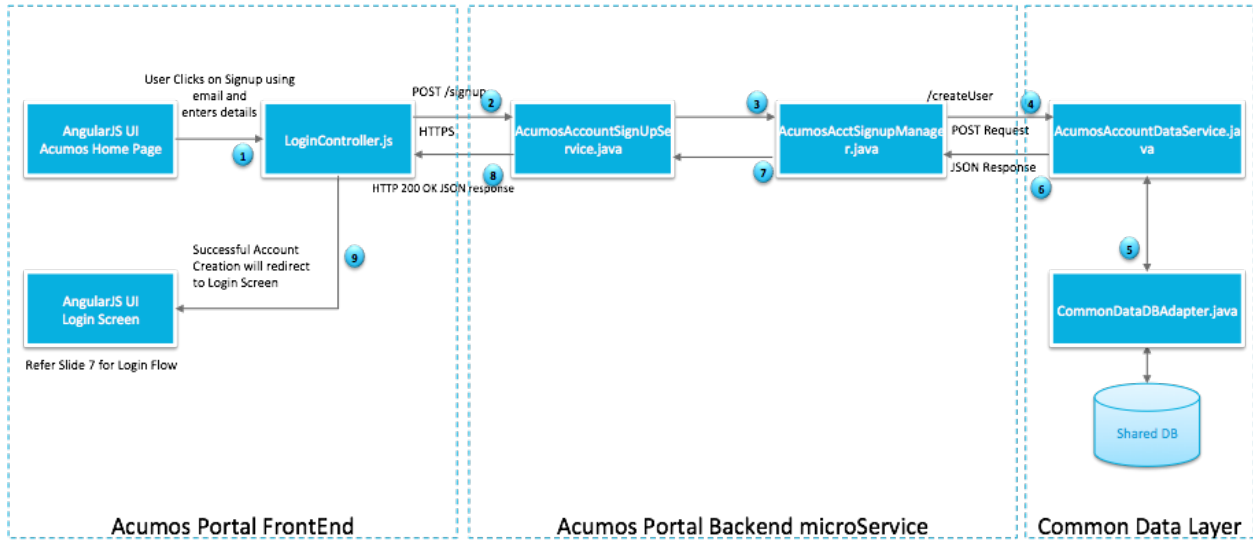
Acumos User Login Flow

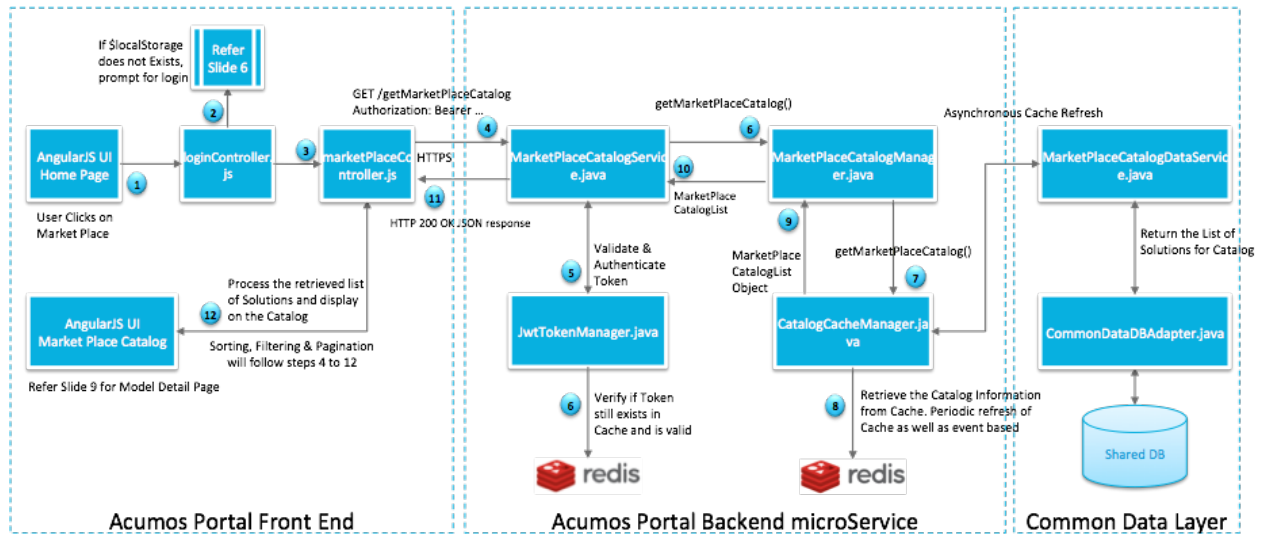
Component Interaction

Following are some illustrative diagrams for common Acumos component interactions.

Acumos Model Detail Flow

Acumos Catalog Flow





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 subscription to a solution published by a peer platform, at the subscribing platform. Note that some subsequent 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 subscription to a solution published by a platform, when received at the publishing platform. Note that some subsequent actions to these steps are not shown in this flow version, e.g. retrieval

of the artifacts for the subscribed solution.

4.2 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](#).

4.2.1 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 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

This project maintains clients for deploying models to different environments.

- Azure Client
- Kubernetes Client
- OpenStack Client

Model On-Boarding

- Java Client
- On-boarding
- Python Client
- R 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

4.3 Documentation Guide

`docs-contributor-guide/index`

Please also visit the [Developer wiki](#), which includes sections on how to contribute to Acumos.

CHAPTER 5

Indices and Tables

- search