

PAPER • OPEN ACCESS

## Deploying the ATLAS Metadata Interface (AMI) on the cloud with Jenkins

To cite this article: F Lambert *et al* 2017 *J. Phys.: Conf. Ser.* **898** 092001

View the [article online](#) for updates and enhancements.

You may also like

- [Looking back on 10 years of the ATLAS Metadata Interface. Reflections on architecture, code design and development methods](#)  
J Fulachier, O Aidel, S Albrand et al.
- [Conditions and configuration metadata for the ATLAS experiment](#)  
E J Gallas, S Albrand, J Fulachier et al.
- [An integrated overview of metadata in ATLAS](#)  
E J Gallas, D Malon, R J Hawkings et al.



**ECS**  
The  
Electrochemical  
Society  
Advancing solid state &  
electrochemical science & technology

**DISCOVER**  
how sustainability  
intersects with  
electrochemistry & solid  
state science research

# Deploying the ATLAS Metadata Interface (AMI) on the cloud with Jenkins

**F Lambert<sup>1</sup>, J Odier<sup>1</sup>, J Fulachier<sup>1</sup>, on behalf of the ATLAS Collaboration.**

<sup>1</sup>Laboratoire de Physique Subatomique et de Cosmologie, Université Grenoble-Alpes, CNRS/IN2P3, 53 rue des Martyrs, 38026, Grenoble Cedex, FRANCE

E-mail: fabian.lambert@lpsc.in2p3.fr

**Abstract.** The ATLAS Metadata Interface (AMI) is a mature application of more than 15 years of existence. Mainly used by the ATLAS experiment at CERN, it consists of a very generic tool ecosystem for metadata aggregation and cataloguing. AMI is used by the ATLAS production system, therefore the service must guarantee a high level of availability. We describe our monitoring and administration systems, and the Jenkins-based strategy used to dynamically test and deploy cloud OpenStack nodes on demand.

## 1. Introduction

This paper describes the software and hardware infrastructures to design the ATLAS Metadata Interface (AMI) ecosystem as a cloud service. AMI is a generic ecosystem for metadata bookkeeping system used by the ATLAS experiment [1] at the Large Hadron Collider (LHC), using either a SQL or NoSQL database. Several important tools, parts of the offline software of the ATLAS experiment, are based on AMI. As an example, the AMI dataset discovery tool is used to catalog the official ATLAS datasets available for analysis. It has been the official dataset bookkeeping tool of the experiment since 2006. As another example, the AMI-Tags tool is used to record the processing of all the official ATLAS datasets and is used in most ATLAS analysis jobs. These applications are critical and must be constantly available as they are widely used worldwide.

AMI can be considered a mature ecosystem, since its basic architecture has been maintained for over ten years. However, the availability and reliability requirements of the the ATLAS experiment make the AMI team continuously scale both software and hardware infrastructure so that the quality of service remains high. The recent software and hardware improvements are described in details in these proceedings [2][3].

In the following we recall the main components of the AMI ecosystem. Then we describe the OpenStack [4] cloud-based hardware infrastructure hosting AMI services, the Jenkins-based [5] software deployment procedure and the AMI-based administration tool designed to manage both of them. Finally we discuss the consequences in terms of flexibility and scalability of our technical choices.

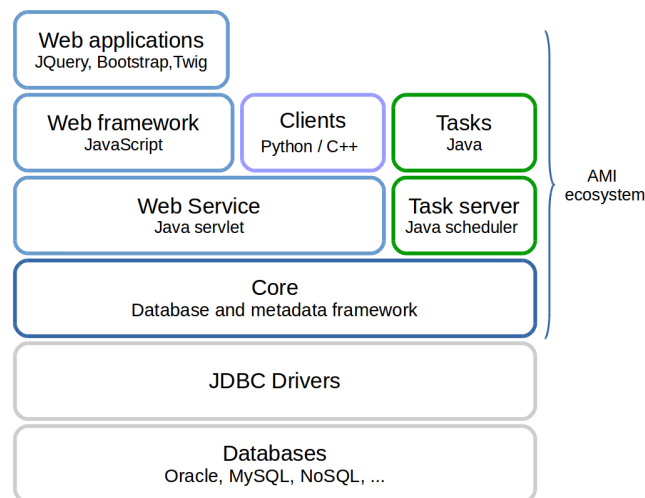


## 2. AMI ecosystem overview

The architecture of AMI was described in detail in some previous publications [6][7]. In this section, the main characteristics of the AMI ecosystem parts are described.

### 2.1. Overview of the ecosystem architecture

The first design principles of the AMI ecosystem architecture were established in 2002-2003. In order to withstand the inevitable evolutions in technology and requirements, a layered architecture (Figure 1) was chosen. The bricks of the layered architecture are well decoupled, making AMI a flexible and scalable ecosystem for designing metadata-oriented applications.



**Figure 1.** AMI layered architecture.

Three families of software use the “core” layer of the AMI ecosystem. The first family is a set of Web applications, the second consists of the AMI tasks subsystem and the last one is a set of lightweight clients. While the lightweight clients are installed independently on user computers, the others are deployed on cloud nodes by dedicated AMI administration software.

### 2.2. Web applications

The first AMI web applications were written in PHP in 2001. They were not structured in layers and, consequently, any evolution became difficult. To address this concern, a Java class hierarchy, nevertheless still strongly coupled with the “core” layer, was written for the generation of HTML pages. This model was used until 2013 and was progressively replaced by a new model based on JavaScript and the Twitter Bootstrap [8] CSS framework. The idea was to decouple the Web interfaces client-side from the business logic server-side. This work was extended further in 2014 to develop the current Web framework JavaScript layer: a JQuery-based [9] library loaded in Web browser client side. At present, all AMI Web applications are based on the AMI Web Framework layer. It interacts with business class objects of the Core layer server-side via a unique java servlet, the Web service, using a standard HTTP protocol. The AMI Web framework layer follows a Model View Controller (MVC) pattern and implements the Twig rendering engine. This has considerably simplified AMI Web application developments and it is described in detail in earlier proceedings [3].

### 2.3. Task subsystem

It is essential for AMI users to have up-to-date metadata. To perform this essential job, the AMI ecosystem has a JAVA task subsystem. It consists of a priority scheduler executing “AMI core layer”-based tasks. While the scheduler is designed as a generic layer, the tasks are specific to the production system from which they are gathering the data.

### 2.4. Lightweight clients

Several lightweight clients are part of the AMI ecosystem, a JavaScript client (the AMI Web framework, see section 2.2), a Python client called pyAMI and a C/C++ client.

The Python client called pyAMI is the most used by the ATLAS experiment. It represents 92% of the requests on the AMI servers. pyAMI can run with both Python 2.6+ and Python 3.x. Based on HTTP/HTTPS, it allows credential and X509 certificate/proxy authentication. The core of pyAMI is generic and is available as a Python library on the Python Package Index (PyPI) [10]. In addition, an ATLAS-specific layer is distributed via PyPI and is also available within Athena, the ATLAS offline control framework.

It is important to notice that all clients share the same HTTP/HTTPS protocol to query AMI, thus, all the information contained in AMI can be retrieved by any client using the same workflow. The lightweights clients are totally decoupled from the AMI core layers and consequently, can be deployed independently.

## 3. AMI infrastructure

In this section, the evolution of the production infrastructure of AMI to a cloud environment is described.

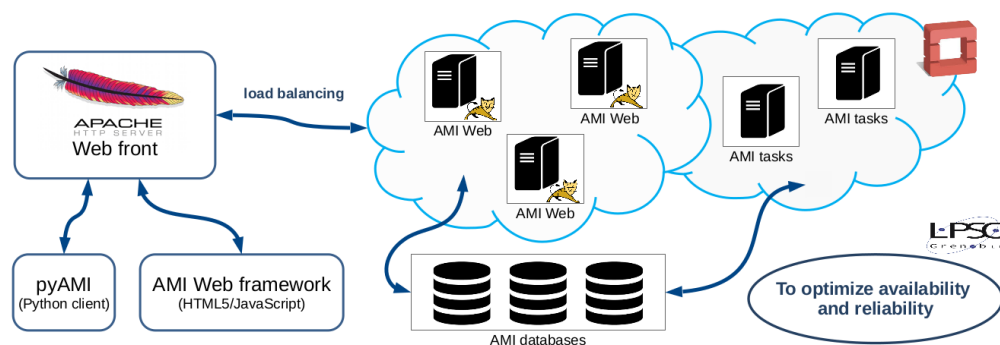
### 3.1. Infrastructures until 2015

The AMI ecosystem was initially deployed at LPSC, Grenoble in 2000 on one single server, using a single database. This model rapidly showed limitations regarding the increasing usage of AMI by the ATLAS community. After 2004, the AMI services were migrated to four load-balanced physical machines hosted at the French Tier 1 site of the Centre de Calcul IN2P3 (CC-IN2P3) at Lyon. This new architecture improved the global availability of the AMI services but implied important changes in the AMI deployment workflow. In order to ensure the homogeneity of the AMI software on the four nodes and to increase deployment efficiency, the automation tool Jenkins was used together with versioning tools (SVN and GIT).

Nevertheless, using physical nodes has some important limitations. First, when a server reaches the end of its maintenance period it has to be removed from the cluster and another one has to be purchased, increasing the maintenance cost. Another problem is that the hardware tends to become inhomogeneous with years. That is why, during 2015, the AMI ecosystem was migrated to a cloud infrastructure.

### 3.2. Current infrastructure

The limitations of an infrastructure based on physical machines are attenuated in a cloud environment. Replacing a virtual machine does not affect hosted services and furthermore the costs are lower. For all these reasons, the several AMI services are at present hosted on an OpenStack cloud-based infrastructure (Figure 2) at the CC-IN2P3 computing center.



**Figure 2.** AMI cloud infrastructure.

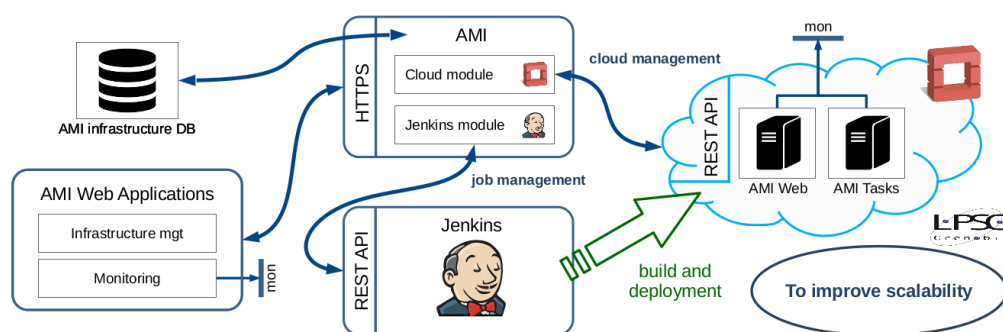
In order to guarantee the availability and reliability of the AMI Web service, an Apache front-end cluster implements load balancing on four cloud nodes running Tomcat [11]. Five task servers are also hosted on cloud nodes ensuring the stability of another essential AMI service: the aggregation of meta-data from the ATLAS production system to the AMI catalogs. In the near future, this infrastructure will be replicated at CERN in order to ensure the continuous availability of AMI services in case of CC-IN2P3 center downtime.

At present, the AMI ecosystem benefits from the *Infrastructure as a Service* (IaaS) model of the OpenStack cloud at CC-IN2P3 and can easily adapt to a load increase by creating new nodes “on demand”. This is a first step for the migration of the AMI ecosystem to a *Software as a Service* (SaaS) model. The aim is to bundle AMI as “on demand” software, easily installable and configurable.

#### 4. AMI deployment workflow

### 4.1. Overview

The AMI administration web interfaces are the visible parts of the hidden machinery for deploying AMI. To do that, two Java modules, parts of the AMI Core layer, interact with Jenkins and OpenStack cloud via their own REST APIs and the AMI infrastructure DB (Figure 3).



### Figure 3. Deployment workflow

#### 4.2. OpenStack and REST API

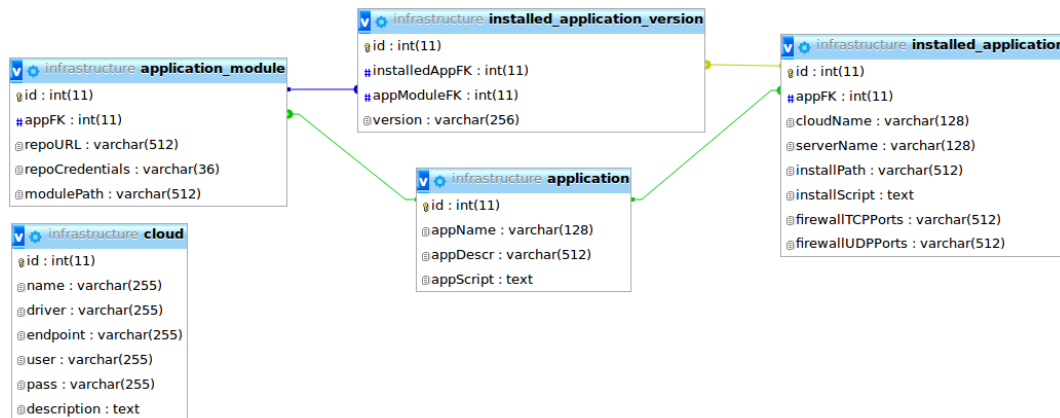
The OpenStack cloud infrastructure can be driven via its REST API. This feature is used by the AMI administration Web application to “on demand” deploy new nodes, with given flavors, and gives full control of the cloud to the AMI ecosystem. The OpenStack REST API is subdivided into six orthogonal services [12]. At present the AMI ecosystem interacts with three of them: the NOVA service for managing the lifecycle of compute instances, the KEYSTONE service for managing authentication and authorization and the GLANCE service to retrieve virtual machine disk images. The AMI administration application interacts with the OpenStack cloud via its REST API using asynchronous HTTP requests.

### 4.3. Infrastructure database

The infrastructure database reflects the AMI ecosystem software structure (Figure 4). It is composed of applications that consist of sets of software modules. Applications are recorded in the “application” table and are linked to their modules recorded in the “application\_module” table. For example, the “AMI-Web” application contains several modules like AMI-Tags, Dataset Discovery, ...

The infrastructure database also models which application is installed, where it is installed and what are the set of corresponding deployed module versions. The cloud infrastructure where applications are deployed is stored in the “installed\_application” table and is of course linked with the “application” table. The module versions deployed are linked to both application modules and installed applications and are recorded in the “installed\_application\_version” table. Lastly, cloud endpoints and their credentials are recorded in the “cloud” table.

At present, this database, hosted on the CC-IN2P3 cluster, is the core of the deployment system. It is populated by administrators using the AMI administration application, modeling at a given time the states of all the AMI software deployments.



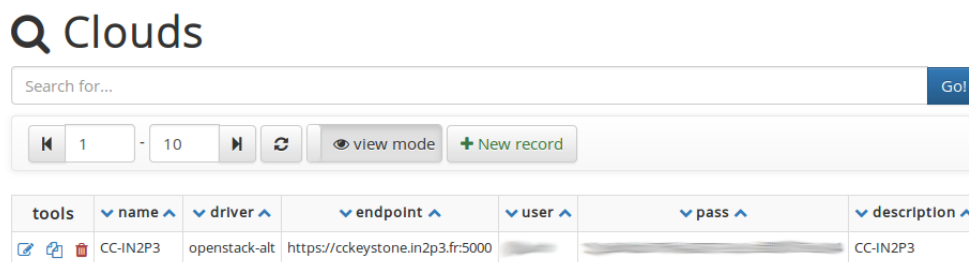
**Figure 4.** AMI infrastructure database

## 5. AMI administration application

The AMI administration Web application has been developed to manage the infrastructure hosting the AMI ecosystem and to easily control the applications deployed on each cloud node. As all the other AMI Web applications, it is based on the JavaScript AMI Web Framework layer.

### 5.1. Clouds management

This part of the administration application is for declaring the cloud system used by AMI. The interface provides tools to edit, clone or delete a specific cloud record. To declare a new cloud, one has to give: the endpoint, the credentials and the driver to be used (Figure 5).

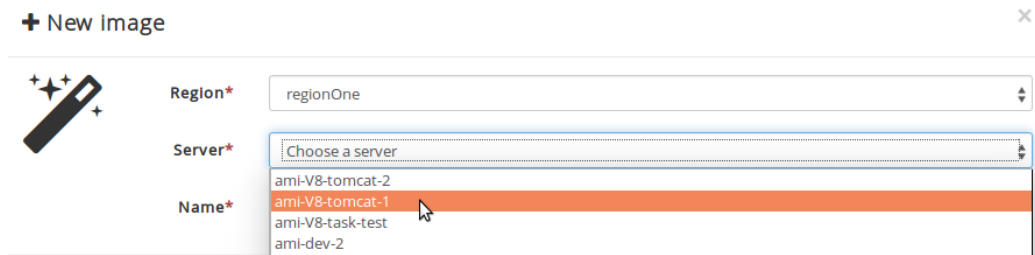


**Figure 5.** Cloud providers management

Currently, only the OpenStack Java driver is developed. It uses the stored credentials to interact with the OpenStack KEYSTONE service for authorization and authentication. Nevertheless the system is generic and can adapt to other types of clouds if an appropriate driver is implemented.

### 5.2. Cloud nodes management

**5.2.1. Images.** Images are preset configurations of cloud nodes that can be used as a starting point to deploy a new server. The AMI administration application uses the OpenStack GLANCE service to provide the image list. It is possible to create new images from existing nodes in order to save and re-produce complex server configurations (Figure 6).



**Figure 6.** Create a cloud image

Two types of configurations are preset for the AMI ecosystem and can be used to quickly deploy new nodes. One is for the nodes that host AMI web applications, where a Tomcat server is preconfigured and comes with all the main AMI web applications. Another one is for nodes hosting the AMI task service where a task server is preconfigured and ready to use.

**5.2.2. Servers.** In order to handle a load increase, or for testing purposes, it is possible to “on demand” instantiate new cloud nodes from the administration application (Figure 7).

Tools	name	region	flavor	Image	status	IPv4List	IPv6List
	ami-V8-tomcat-2	regionOne	ami.l.large	ami-V8-tomcat-test	ACTIVE		
	ami-V8-tomcat-1	regionOne	ami.l.large	N/A	ACTIVE		
	ami-V8-task-test	regionOne	ami.l.large	ami-V8-task-test	ACTIVE		
	ami-dev-2	regionOne	m1.large	ami-dev-2-16-12-15	ACTIVE		

**Figure 7.** Cloud nodes management

Administrators can also easily stop, start or reboot an existing server from the web interface. The tools provide also an interface to add a new server using one of the available predefined flavors (disk, memory, ...) and one of the existing images (applications and system presets).

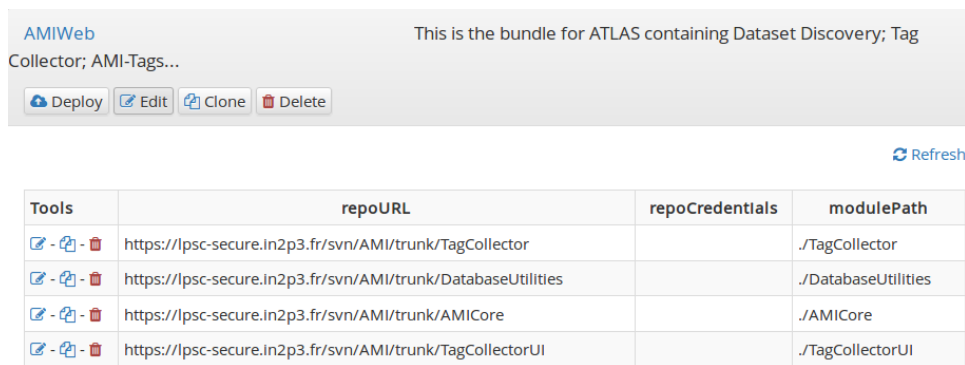
### 5.3. Jenkins job management

Jenkins is an automation server initially designed for continuous integration projects. AMI deployment tasks are automatized within Jenkins jobs since 2011. As for OpenStack, Jenkins has a REST API for driving the execution of its jobs. AMI uses this REST API to manage the versions of AMI software that are deployed on cloud nodes.

Two main job families, one for AMI tasks, the other for AMI web applications, are configured in Jenkins taking as parameters the target server and the version of the AMI software to deploy. The management of the version and the location of the deployed software are handled by the AMI administration application, as described in the following section.

### 5.4. Applications management

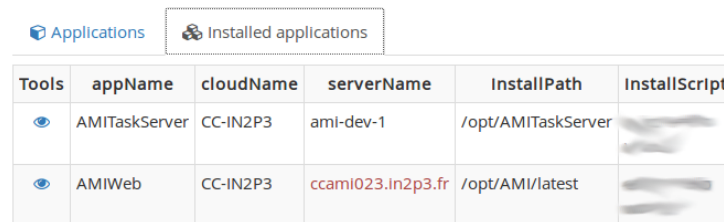
Applications are defined in the AMI administration tools as sets of modules located in a source code repository (Figure 8).



Tools	repoURL	repoCredentials	modulePath
	https://lpsc-secure.in2p3.fr/svn/AMI/trunk/TagCollector		./TagCollector
	https://lpsc-secure.in2p3.fr/svn/AMI/trunk/DatabaseUtilities		./DatabaseUtilities
	https://lpsc-secure.in2p3.fr/svn/AMI/trunk/AMICore		./AMICore
	https://lpsc-secure.in2p3.fr/svn/AMI/trunk/TagCollectorUI		./TagCollectorUI

**Figure 8.** Application modules

Once defined, these applications are deployed by Jenkins deployment jobs providing the mandatory information to the Jenkins REST API.



Tools	appName	cloudName	serverName	InstallPath	InstallScript
	AMITaskServer	CC-IN2P3	ami-dev-1	/opt/AMITaskServer	
	AMIWeb	CC-IN2P3	ccami023.in2p3.fr	/opt/AMI/latest	

**Figure 9.** Installed applications

The AMI administration application shows where the software is deployed (Figure 9) and gives full control of the installed module versions.

## 6. Outlook

The infrastructure hosting the AMI ecosystem was migrated progressively from a system based on physical machines to a cloud-based infrastructure at CC-IN2P3. This migration increases the stability and the scalability of the AMI services and reduces the maintenance cost. A dedicated AMI administration Web application was developed to manage both the cloud nodes infrastructure at CC-IN2P3 and the deployment of the AMI softwares. This tool benefits from the generic AMI Web framework facilities and makes heavy use of the Jenkins and the Openstack REST APIs to administer the deployed applications. At the request of ATLAS management, the CC-IN2P3 infrastructure model will be duplicated at CERN soon in order to enhance the AMI replica instance availability and stability.

This work on the AMI deployment workflow makes the ecosystem an easily manageable solution for the experiments, reducing to a minimum the support required of the core team.

## Acknowledgements

Over the years we have been helped and supported by many people in the ATLAS collaboration, and at the CC-IN2P3, in particular Philippe Cheynet, Benoît Delaunay, Noel Dawe, Markus Elsing, Pierre-Etienne Macchi, Yannick Perret, Emil Obreshkov, Mattieu Puel, David Quarrie and Jean-René Rouet.

## References

- [1] ATLAS Collaboration 2008 The ATLAS Experiment at the CERN Large Hadron Collider *JINST* **3** S08003 doi:10.1088/1748-0221/3/08/S08003
- [2] Odier J, Aidel O, Albrand S, Fulachier J, Lambert F 2015 Evolution of the architecture of the ATLAS Metadata Interface (AMI). Proceedings of the 21st International Conference on Computing in High Energy and Nuclear Physics (CHEP2015) *J. Phys.: Conf. Ser.* **64** 042040
- [3] Odier J, Aidel O, Albrand S, Fulachier J, Lambert F 2015 Migration of the ATLAS Metadata Interface (AMI) to Web 2.0 and cloud Proceedings of the 21st International Conference on Computing in High Energy and Nuclear Physics (CHEP2015) *J. Phys.: Conf. Ser.* **664** 062044
- [4] OpenStack: <http://www.openstack.org/>
- [5] Jenkins open-source automation server: <https://jenkins.io/>
- [6] Fulachier J, Aidel O, Albrand S, Lambert F 2013 Looking back on 10 years of the ATLAS Metadata Interface. Proceedings of the 20th International Conference on Computing in High Energy and Nuclear Physics (CHEP2013) *J. Phys.: Conf. Ser.* **513** 042019 doi:10.1088/1742-6596/513/4/042019
- [7] Albrand S, Doherty T, Fulachier J, Lambert F 2008 The ATLAS Metadata Interface *J. Phys.: Conf. Ser.* **119** 072003 doi:10.1088/1742-6596/119/7/072003
- [8] Twitter Bootstrap: <http://getbootstrap.com/>
- [9] JQuery: <http://jquery.com/>
- [10] pyAMI sources: [https://pypi.python.org/pypi/pyAMI\\_core/](https://pypi.python.org/pypi/pyAMI_core/)
- [11] Tomcat: <https://tomcat.apache.org/>
- [12] The OpenStack projects: <http://www.openstack.org/software/project-navigator/>