

Deliverable <2.4>

Code Release

|  |  |
| --- | --- |
| **Grant Agreement number:** | 786922 |
| **Project acronym:** | ASTRID |
| **Project title:** | AddreSsing ThReats for virtualIseD services |
| **Start date of the project:** | 01/05/2018 *dd/mm/yyyy* |
| **Duration of the project:** | 36 months |
| **Type of Action:** | Research & Innovation Action (RIA) |
| **Project Coordinator:** | Name: Orazio Toscano  Phone: +39 010 600 2223  E-mail: orazio.toscano@ericsson.com |
|  |  |
| **Due Date of Delivery:** | <Mx> *Mx* (<dd/mm/yyyy> *dd/mm/yyyy*) |
| **Actual Date of Delivery:** | <dd/mm/yyyy> *dd/mm/yyyy* |
| **Work Package:** | <WPn – Work Package Name> |
| **Type of the Deliverable:** | <R, DEM, DEC, ETHICS, ORDP, OTHER> |
| **Dissemination level:** | <PU, CO, EU-xxx> |
| **Editors:** | <SHORTNAME of the partners> |
| **Version:** | <A.B> |

|  |  |
| --- | --- |
| **List of Authors** | |
| ETI | ERICSSON TELECOMUNICAZIONI |
| <Name Surname 1>, <Name Surname 2>, <Name Surname 3> | |
| SURREY | UNIVERSITY OF SURREY |
| <Name Surname 1>, <Name Surname 2>, <Name Surname 3> | |
| CNIT | CONSORZIO NAZIONALE INTERUNIVERSITARIO PER LE TELECOMUNICAZIONI |
| <Name Surname 1>, <Name Surname 2>, <Name Surname 3> | |
| INFO | INFOCOM S.R.L. |
|  | |
| POLITO | POLITECNICO DI TORINO |
|  | |
| TUB | TECHNISCHE UNIVERSITAET BERLIN |
|  | |
| AGE | AGENTSCAPE AG |
|  | |
| UBITECH | GIOUMPITEK MELETI SCHEDIASMOS YLOPOIISI KAI POLISI ERGON PLIROFORIKIS ETAIREIA PERIORISMENIS EFTHYNIS |
|  | |
| DTU | DANMARKS TEKNISKE UNIVERSITET |
|  | |

Disclaimer

The information, documentation and figures available in this deliverable are written by the ASTRID Consortium partners under EC co-financing (Call: H2020-DS-SC7-2017, Project ID: 786922) and do not necessarily reflect the view of the European Commission.

The information in this document is provided “as is”, and no guarantee or warranty is given that the information is fit for any particular purpose. The reader uses the information at his/her sole risk and liability.

Copyright

Copyright © yyyy the ASTRID Consortium. All rights reserved.

The ASTRID Consortium consists of:

ERICSSON TELECOMUNICAZIONI (ETI)

UNIVERSITY OF SURREY (SURREY)

CONSORZIO NAZIONALE INTERUNIVERSITARIO PER LE TELECOMUNICAZIONI (CNIT)

INFOCOM S.R.L. (INFO)

POLITECNICO DI TORINO (POLITO)

TECHNISCHE UNIVERSITAET BERLIN (TUB)

AGENTSCAPE AG (AGE)

GIOUMPITEK MELETI SCHEDIASMOS YLOPOIISI KAI POLISI ERGON PLIROFORIKIS ETAIREIA PERIORISMENIS EFTHYNIS (UBITECH)

DANMARKS TEKNISKE UNIVERSITET (DTU)

This document may not be copied, reproduced or modified in whole or in part for any purpose without written permission from the ASTRID Consortium. In addition to such written permission to copy, reproduce or modify this document in whole or part, an acknowledgement of the authors of the document and all applicable portions of the copyright notice must be clearly referenced.

Table of Contents

[Disclaimer 3](#_Toc2261535)

[Copyright 3](#_Toc2261536)

[Table of Contents 4](#_Toc2261537)

[1 Executive Summary 4](#_Toc2261538)

[2 Introduction 4](#_Toc2261539)

[2.1 Standards for Deliverable editing 4](#_Toc2261540)

[2.1.1 Language 4](#_Toc2261541)

[2.1.2 How to integrate contributions 4](#_Toc2261542)

[2.1.3 Handling of graphics, figures, and tables in documents 4](#_Toc2261543)

[2.1.4 Blank between numerical values and their units 4](#_Toc2261544)

[3 Heading 1 4](#_Toc2261545)

[3.1 Heading 2 4](#_Toc2261546)

[3.1.1 Heading 3 4](#_Toc2261547)

[3.2 Heading 2 4](#_Toc2261548)

[3.2.1 Heading 3 4](#_Toc2261549)

[4 Heading 1 4](#_Toc2261550)

[4.1 Heading 2 4](#_Toc2261551)

[4.1.1 Heading 3 4](#_Toc2261552)

[5 Conclusions 4](#_Toc2261553)

[References 4](#_Toc2261554)

[Annex 1: Title of annex 1 4](#_Toc2261555)

[Annex 2: Title of annex 2 4](#_Toc2261556)

# Executive Summary

# The aim of this document is to outline Introduction

## This deliverable will cover the first release of the ASTRID components, including manuals. This deliverable is the accompanying documentation to the first software release ASTRID components acts as a guide to the documentation of the ASTRID public release, which can be found on Github[[1]](#footnote-1). For this, the document provides hyperlinks to the relevant documents in the Github repositories. The repositories provide approximately … pages of DOC documentation that describes: how to install and configure the system; how to use the code; descriptions of the key design decisions; system level implementation examples; and, a set of modelling and evaluation tools. This documentation is indexed and described in this deliverable. However, unlike the deliverable, the repositories may be modified with minor improvements to the functionalities outlined here, as well as bug fixes.

Following our strategy, the main objective of the release is to provide a self-contained, well tested, and thoroughly documented open source version of our ASTRID components, along with a set of selected support tools that enable rapid experimentation on diverse networking infrastructures. We do this primarily in order to support our push towards standardization. An open source release enables key stakeholders to experiment with our architecture and solutions, and is especially helpful for the standardization of the relevant interfaces.

## This deliverable describes new contributions to the project repositories. New work includes:

1. K8s Firewall Handler
2. SYN flood monitor
3. Cubebeat

The remainder of this document is structured as follows.

# Astrid Controller

Astrid Controller is the core element of the ASTRID conceptual architecture, including both service management and situational awareness. Starting from the descriptive and applicative semantics of the Context Model, controller is expected to deploy and manage the life-time of the service, by adapting the awareness layer of individual components and the whole service graph according to specific needs of detection algorithms. This means that monitoring operations, types and frequency of event reporting, level of logging is selectively and locally adjusted to retrieve the exact amount of knowledge, without overwhelming the whole system with unnecessary information. The purpose is to get more details for critical or vulnerable components when anomalies are detected that may indicate an attack, or when a warning is issued by cyber-security teams about new threats and vulnerabilities just discovered.

The supported actions, of astrid\_ontroller, includes: set and change at run-time security configuration (logging, packet filtering rules), de-provision the service, add/replace/delete virtual functions in the service graph to remove vulnerabilities or compromised entities, divert traffic for legal interception.

## Installing Controller via Maven

(Spring Boot application with Embedded Tomcat)

* install [jdk1.8.X YY]
* install [maven](https://maven.apache.org/install.html)
* mvn clean package
* java -jar target/controller-0.0.1-SNAPSHOT.jar

## API Resources Design

|  |  |  |  |
| --- | --- | --- | --- |
| **Resources** | **URLs** | **XML repr** | **Meaning** |
| **infrastucture** | /insfrastructure | InfrastructureInfo | XML file describing infrastructure information |
| **policy** | /policy |  | yaml file describing the policy |
| **even** | /event | InfrastructureEvent | XML file describing changes in the NFV |

# ASTRID-kube

ASTRID-kube watches for changes in Kubernetes related to namespaces and pods and later sends the resulting infrastructure to all modules interested in knowing it.

## Installation

After cloning the repository, you can just run the provided executable file ASTRID-kube. For convenience, you can add its path to PATH:

$ PATH=$PATH:<path-to-ASTRID-kube>

NOTE: make sure to make the appropriate edits to the conf.yaml in the settings folder.

#### Configuration

Below is a brief explanation on the conf.yaml configuration file:

* fwInitTimer: how many seconds to wait before creating the firewall when a pod is detected to be running. Unstable pods may compromise the stability of the rest of the graph, so this field must be set to a reasonable value to wait for any crashes to happen and to wait for all sidecars inside it to finit initializing.
* paths.kubeconfig: if your kubeconfig file resides in the default folder, leave this empty. Otherwise, please fill this field accordingly.
* endpoints.verekube.infrastructure-info: the endpoint where to send the resulting infrastructure. Usually, this is in the already provided format, you should only edit the provided ip with that of your machine running verekube.
* endpoints.verekube.infrastructure-event (experimental): the endpoint where to send updates about the infrastructure.
* endpoints.cb.configuration: the endpoint where the cb (the firewall rules pusher) is running.
* formats.infrastructure-info: specify the format you want the infrastructure information to be sent as. Accepted values are xml, yaml or json.
* formats.infrastructure-event: specify the format you want updates about the infrastructure to be sent as. Accepted values are xml, yaml or json.

## Usage

Once running, ASTRID-kube will keep watching for changes in Kubernetes. To work properly, you need to make some adjustments to a couple of Kubernetes resources.

#### Graphs

Kubernetes does not have a concept of Graphs, but namespaces are the only thing that comes close to that definition. ASTRID-kube abstracts the idea of graphs with Kubernetes namespaces, and in order to be able to detect which namespaces you want to be managed by ASTRID-kube, you need to write an appropriate annotation under the namespace's metadata before deploying it containing a json list of all the deployments contained inside it:

kind: Namespace  
apiVersion: v1  
metadata:  
 name: mygraph  
 labels:  
 name: mygraph  
 annotations:  
 astrid.io/deployments: "[\"simple-service\", \"nodejs\", \"apache\"]"

Please make sure the names in the list match exactly the name of the corresponding deployment, otherwise ASTRID-kube will wait indefinitely for the applications to appear. Additionally, make sure the list is definitive, as this will signal ASTRID-kube the infrastructure you want to be built. All applications deployed later will be ignored.

#### Security Components

Once running, all applications will be protected with the appropriate security components, as specified in the astrid.io/security-components annotation. This is a json list of all security functions that the application needs. As of now, firewall is supported, but other components will be available soon.

Take a look at the following deployment, which needs to be protected with a firewall.

apiVersion: apps/v1   
kind: Deployment  
metadata:  
 name: simple-service  
 namespace: mygraph  
 annotations:  
 astrid.io/security-components: "[\"firewall\"]"  
spec:  
 selector:  
 matchLabels:  
 app: nginx  
 replicas: 1  
 template:  
 metadata:  
 labels:  
 app: simple-service  
 spec:  
 containers:  
 - name: simple-service  
 image: asimpleidea/simple-service:latest  
 env:  
 - name: APP\_NAME  
 value: "example"  
 ports:  
 - containerPort: 80

## Polycube

ASTRID-kube relies on [Polycube](https://github.com/polycube-network/polycube) to instantiate all the proper network functions and, to do so, polycube must be injected as a sidecar in your applications.

#### Automatic sidecar injection

The [polycube sidecar injector](https://github.com/SunSince90/polycube-sidecar-injector) is strongly recommended, as it will perform this job automatically for you. To know more about the polycube sidecar injector, please refer to the provided link.

Once installed, add the following **label** to the example namespace provided above:

template:  
 metadata:  
 polycube.network/sidecar: enabled

Additionally, the same key/pari must be added as **annotation** under spec.template.metadata of your deployments. Example:

polycube.network/sidecar: enabled

#### Manual sidecar injection

If you want to inject polycube manually, you have to add the following container to your deployments, under spec.template.containers:

- name: polycubed  
 image: polycubenetwork/polycube:latest  
 imagePullPolicy: Always  
 command: ["polycubed", "--loglevel=DEBUG", "--addr=0.0.0.0", "--logfile=/host/var/log/pcn\_k8s"]  
 volumeMounts:  
 - name: lib-modules  
 mountPath: /lib/modules  
 - name: usr-src  
 mountPath: /usr/src  
 - name: cni-path  
 mountPath: /host/opt/cni/bin  
 - name: etc-cni-netd  
 mountPath: /host/etc/cni/net.d  
 - name: var-log  
 mountPath: /host/var/log  
 securityContext:  
 privileged: true  
 ports:  
 - name: polycubed  
 containerPort: 9000  
 terminationMessagePolicy: FallbackToLogsOnError

Add the following volumes, under spec.template.volumes (if not present, please add it)

- name: lib-modules  
 hostPath:  
 path: /lib/modules  
- name: usr-src  
 hostPath:  
 path: /usr/src  
- name: cni-path  
 hostPath:  
 path: /opt/cni/bin  
- name: etc-cni-netd  
 hostPath:  
 path: /etc/cni/net.d  
- name: var-log  
 hostPath:  
 path: /var/log  
- name: netns  
 hostPath:  
 path: /var/run/netns  
- name: proc  
 hostPath:  
 path: /proc/

## Examples

The artifacts folder contains two examples that you can deploy after running ASTRID-kube. Deploy graph-psi.yaml if you are using polycube-sidecar-injector; or graph.yaml if you are injecting polycube manually.

# PolyCube

Polycube is eBPF/XDP-based software framework for fast network services running in the Linux kernel. Available: <https://github.com/polycube-network/polycube>

# Astrid-Config

##### astrid-config is a fork of the VEREFOO project (VErified REFinement and Optimized Orchestration) designed to provide an automatic way to allocate packet filters – the most common and traditional firewall technology – in a Service Graph defined by the service designer and an auto-configuration technique to create firewall rules with respect to the specified security requirements.

The astrid-config involves the formulation of a MaxSMT problem, whose objective is to maximize the sum of the weights assigned to the satisfied soft clauses, with respect to hard constraints that always require to be satisfied. Its targets are on one side the allocation of the minimum number of NSFs instances to reduce the resource consumption due to the allocation of the corresponding virtual functions, on the other side the reduction of the rules describing their configuration to improve the efficiency of the filtering operations. The MaxSMT problem is formulated so as to provide also a formal verification that the achieved solution is formally correct. This document describes the installation details and the initial set of interfaces required for its interaction with other modules of ASTRID.

## Installation and deployment

## Z3 library support

[Download](https://github.com/Z3Prover/z3/releases) the correct version of Z3 according to your OS and your JVM endianness. For the correct functioning of the application, you must have the Z3 native library and include it to Java Library Path. The most convenient way to do this is add the path that the library to the dynamic linking library path.

* In Linux is LD\_LIBRARY\_PATH
* In MacOS is DYLD\_LIBRARY\_PATH
* In Windows is PATH

e.g., \* sudo nano /etc/environment \* LD\_LIBRARY\_PATH = $LD\_LIBRARY\_PATH:/home/verefoo/z3/bin/ \* Z3 = /home/verefoo/z3/bin/ (also required)

Troubleshooting

If z3 throws java.lang.UnsatisfiedLinkError: no libz3java in java.library.path exception, add all \*.so files of z3 release into /usr/lib folder.

## Installing astrid-config via Maven

(Spring Boot application with Embedded Tomcat) [Solution 1]

* install [jdk1.8.X YY](http://www.oracle.comntechnetwork/java/javase/downloads/jdk8-downloads-2133151.html);
* install [maven](https://maven.apache.org/install.html)
* mvn clean package
* java -jar target/verifoo-0.0.1-SNAPSHOT.jar

Swagger documentation can be accessed at <localhost:8085/verefoo>.

## Installing astrid-config via Ant

(Apache Tomcat required) [Solution 2]

* install [jdk1.8.X YY](http://www.oracle.comntechnetwork/java/javase/downloads/jdk8-downloads-2133151.html);
* install [Apache Tomcat 8](https://tomcat.apache.org/download-80.cgi);
  + set CATALINA HOME ambient variable to the directory where you installed Apache;
  + (optional) configure Tomcat Manager:
  + open the file %CATALINA\_HOME%\conf\tomcat-users.xml
  + under the tomcat-users tag place, initialize an user with roles "tomcat, manager-gui, manager-script". An example is the following content: xml <role rolename="manager-gui"/> <role rolename="manager-script"/> <role rolename="admin-gui"/> <role rolename="admin-script"/> <user username="admin" password="admin" roles="manager-gui,manager-script,admin-scripts"/>
  + edit the "to\_be\_defined" fields in tomcat-build.xml with the before defined credentials;
* execute the generate ant task in order to generate the .war;
* launch Tomcat 8 with the startup script %CATALINA\_HOME%\bin\startup.bat or by the start-tomcat task ant;
* (optional) if you previously configured Tomcat Manager you can open a browser and navigate to this link <http://localhost:8080/manager> and login using the proper username and password (e.g., admin/admin in the previous example);
* (optional) you can deploy/undeploy/redeploy the downloaded WARs through the web interface.

## API Resources Design

|  |  |  |  |
| --- | --- | --- | --- |
| **Resources** | **URLs** | **XML repr** | **Meaning** |
| **ROOT** | / | Hyperlinks | XML file with the hyperlinks to the other resources |
| **deployment** | /deployment | NFV | XML file with integrated deployment information |
| **dc** | /dc | InfrastructureInfo | XML file describing infrastructure information |
| **podevent** | /podevent | InfrastructureEvent | XML file describing changes in the NFV |

# SYN Flood Monitor

This service exports some metrics that can be used to detect a possible SYN Flood attack.

## Features

* Retrieves a set of TCP/IP parameters that can be used to detect SYN flooding attacks

## Limitations

* It must be launched on the host that needs to be monitored. It cannot operate as a 'main in the middle' mode, i.e., inspecting network traffic directed toward a remote host.

## How to use

Technically, pcn-synflood is a transparent service, hence it should be attached to an existing network interface (e.g., netdev or a virtual link between Polycube services). However, given that the current implementation retrieves traffic statistics using the metrics provided by the operating system, it can be even instantiated without attaching it to any network interface.

## Exported metrics

* tcpAttemptFails: number of failed TCP connections [1].
* tcpOutRsts: number of TCP segments sent, containing RST flag.
* deliverRatio: ratio between the number of IP pkts delivered to application protocols and the total number of received pkts.
* responseRatio: ratio between the number of IP pkts requests to send by application protocols and the total number of received pkts.

## Additional details

[1] tcpAttemptFails

It measures the number of times TCP connections have made a direct transition to the CLOSED state from either the SYN-SENT state or the SYN-RCVD state, plus the number of times TCP connections have made a direct transition to the LISTEN state from the SYN-RCVD state. It refers to variable tcpAttemptFails documented in RFC 1213.

It is worth mentioning that this parameter is rather general and can be used to check for unusual situations on both client and server side. For instance, it can be used to detect either (1) that a server is under attack (e.g., TCP state machine goes from SYN-RCVD to LISTEN or CLOSED), or (2) that a client is currently attacking a remote target (e.g., TCP state machine goes from SYN-SENT to CLOSED).

The most common case, i.e., that a server is under attack, corresponds at least to the following unusual TCP sequences: - [SYN, timeout]. The server receives a SYN packet, but it cannot answer any more because it is overwhelmed. This connection will be ended after server time?out, as described earlier. - [SYN (Client, Server), RST (Server, Client)]. This sequence means either that the server is the victim of a DoS attack because it cannot reply to the legitimate client any more, or that there is not applications listening on that port. - [SYN, SYN/ACK, timeout]. The server waits indefinitely for the ACK packet, either because the IP source address is spoofed or because the ACK packet is rejected because of network congestion. This sequence can correspond to a DoS attack. This connection will be ended after server time?out. - [SYN, SYN/ACK, RST]. This handshake sequence can correspond to a DDoS attack. At the reception of the SYN/ACK packet, the client host then transmits an RST packet to the server because it never sent a SYN packet.

[Available] <https://github.com/polycube-network/polycube/tree/master/src/services/pcn-synflood>

# CB

CB is a rule forwarder which creates the appropriate firewall rules after receiving configuration [astrid-config](https://gitlab.com/astrid-repositories/wp2/astrid-config) and pushes them on the firewall of the pods involved.

## Installation

To run this, execute cb in the root folder

## API

|  |  |  |  |
| --- | --- | --- | --- |
| **Resources** | **URLs** | **XML repr** | **Meaning** |
| **ROOT** | / | Rules configuration | Receive configuration from verefoo |

# Context Broker

APIs to interact with the Context Broker's database. Through a **REST** Interface, it exposes data and events stored in the internal storage system in a structured way. It provides uniform access to the capabilities of monitoring agents.

## Table of Contents

* [Context Broker APIs](#context-broker-apis)
* [Table of Contents](#table-of-contents)
* [Terminologys](#terminologys)
* [Data Model](#data-model)
* [References](#references)
* [Methods](#methods)
  + [ExecEnv](#execenv)
  + [ExecEnv type](#execenv-type)
  + [Network link](#network-link)
  + [Network link type](#network-link-type)
  + [Connection](#connection)
  + [Agent Instance](#agent-instance)
  + [Catalog](#catalog)
  + [Data Collection](#data-collection)
  + [Full Query](#full-query)
* [Installation](#installation)
* [Usage](#usage)
* [Display help](#display-help)
* [Production environment](#production-environment)
* [Debug enabled in Development environment](#debug-enabled-in-development-environment)
* [Extra](#extra)

## 

## Terminology

|  |  |
| --- | --- |
| **Term** | **Meaning** |
| *ACL* | Access Control List |
| *API* | Application Program Interface |
| *BA* | Basic Authentication |
| *BPF* | Berkeley Packet Filter |
| *CB* | Context Broker |
| *CRUD* | Create - Read - Update - Delete |
| *DB* | Database |
| *eBPF* | extended BPF |
| *ELK* | Elastic - LogStash - Kibana |
| *ExecEnv* | Execution Environment |
| *gRPC* | Google RPC |
| *HOBA* | HTTP Origin-Bound Authentication |
| *HTTP* | Hyper Text Transfer Protocol |
| *ID* | Identification |
| *IP* | Internet Protocol |
| *JSON* | Java Object Notation |
| *LDAP* | Lightweight Directory Access Protocol |
| *RBAC* | Role-Based Access Control |
| *regex* | regular expression |
| *RFC* | Request For Comments |
| *RPC* | Remote Procedure Call |
| *SCM* | Security Context Model |
| *SLA* | Service Level Agreements |
| *SQL* | Structured Query Language |
| *VNF* | Virtual Network Function |
| *YANG* | Yet Another Next Generation |

## Data Model

Data Model

Data Model

Each table in the figure represents an *index* in the Elastic nomenclature. Considering the *NoSQL* nature of the Elasticsearch engine for each document of any indices it is possible to add additional properties. The schema is not static and defined at priori, but it is dynamic allowing the possibility to add custom properties for a specific document. Elastic requires the name of the index in a lowercase format. For this reason, all the index names follow the *dash-case*[[2]](#footnote-2) format. Instead, each property[[3]](#footnote-3) follows the snake-case[[4]](#footnote-4) format. All the properties marked with an asterisk (\*) are required by the APIs in order to make the POST request (i.e. to create a new resource, for example an ExecEnv). Instead the ones underlined are identifies the specific resource (aka document in Elastic nomenclature or record in traditional DB one) and must be unique.

The *data* index contains the all the data collected from the ExecEnvs by means of the agent[[5]](#footnote-5). The common attributes are:

1. *id* (unique identifier)[[6]](#footnote-6);
2. *id* of the source ExecEnv (*exec\_env\_id*); and
3. *id* of the agent instance that collect the data (*agent\_instance\_id*).

The ID property type accepts only lowercase values without space that start with an alphabetic character, e.g: apache is valid but not Apache. Then, the other two properties are related to the time-stamp:

1. *timestamp\_event* when event is occurred; and
2. *timestamp\_agent* when the agent instance collect the data.

With the term agent instance, we refer to a specific agent installed in the ExecEnv.

The *exec-env* index contains the *hostname* of the remote host where is it allocated and the *type\_id* field that correspond a specified type of ExecEnv. The different type of ExecEnv are defined with the index exec-env-type. Currently, the available ones are: *i*) *Virtual Machine* and *ii*) *Container*. Obviously, it is possible to add other types depending on the specific requirements.

The *agent-catalog* index contains specific information related to the agents, and in particular the Beats of the Elastic stack and eBPF-based services deployed with the *Polycube* framework. For a detailed description of these properties see [[7]](#footnote-7) and [[8]](#footnote-8). Each agent in the catalog is characterized by one or more options. The options are defined with the *agent-option nested-index*[[9]](#footnote-9). This index described the option in terms of name and relative type. At this moment, the supported types are: *integer* (e.g. 1, 2, etc.), *number* (e..g 1, 2.3, etc.); *time-duration* (e.g. 1s, 2m, 3h, etc.); *string*; *choice*, *obj*, and *boolean* (i.e. true or false). There two additional (and optional) properties: *list* and *values*. The first one indicate if the option is a list or not (default value: false); while the latter one described the data in the case the type is choice or obj. To accept different types, the values property can be of any type.

All the data of the installed agent is stored in the *agent-instance* index. This index contains the options got from the catalog with the actual values. In addition, it includes the ID of the ExecEnv where the agent is installed (*exec\_env\_id*) and the current *status* in terms of *started* or *stopped*.

The network links are defined with the relative index where it is indicate the type. All the possible network link types are defined in the *network-link-type* index. At the moment, the possible types are: *Point to Point* (Point 2 Point), *Multi-point*, and *Slice*. Similar to the ExecEnv case, also for the network link, it is possible to add additional types depending on specific needs.

In addition, the data model allow to see the status of the connections between the ExecEnvs. The *connection* index couples the ExecEnv and the network link to which it belongs. This index should contains all the information regarding the network link and the ExecEnv as, for example, the IP address (version 4 and/or 6) or if the link is encrypted and how (which method, etc.).

The *software* index contains the installed software with relative properties. Each software record is referred to a specific ExecEnv that indicate where the software is installed. This part is out of scope of the ASTRID project context and, for this reason, it is this highlighted with a dashed box. The API implementation does not consider this index. Nervelessness, it represents a typical solution for various common cases. The proposed data model allows the customization with the integration of additional entities in very simple way.

## References

## Methods

### ExecEnv

|  |  |  |
| --- | --- | --- |
| **HTTP Method** | **Path** | **Action** |
| GET | /config/exec-env | Returns the ExecEnvs selected by the query in the request body (or all it the request body is empty). |
| POST | /config/exec-env | Create a new ExecEnv. |
| PUT | /config/exec-env/{*id*} | Update the ExecEnv with id = {*id*}. |
| DELETE | /config/exec-env | Delete the ExecEnvs selected by the query in the request body (or nothing it the request body is empty). |

### ExecEnv type

|  |  |  |
| --- | --- | --- |
| **HTTP Method** | **Path** | **Action** |
| GET | /config/exec-env-type | Returns the ExecEnv types selected by the query in the request body (or all it the request body is empty). |
| POST | /config/exec-env-type | Create a new ExecEnv type. |
| PUT | /config/exec-env-type/{*id*} | Update the ExecEnv with id = {*id*}. |
| DELETE | /config/exec-env-type | Delete the ExecEnv types selected by the query in the request body (or nothing it the request body is empty). |

#### Network link

|  |  |  |
| --- | --- | --- |
| **HTTP Method** | **Path** | **Action** |
| GET | /config/network-link | Returns the network links selected by the query in the request body (or all it the request body is empty). |
| POST | /config/network-link | Create a new network link. |
| PUT | /config/network-link/{*id*} | Update the network link with id = {*id*}. |
| DELETE | /config/network-link | Delete the network links selected by the query in the request body (or nothing it the request body is empty). |

#### Network link type

|  |  |  |
| --- | --- | --- |
| **HTTP Method** | **Path** | **Action** |
| GET | /config/network-link-type | Returns the network link types selected by the query in the request body (or all it the request body is empty). |
| POST | /config/network-link-type | Create a new network link type. |
| PUT | /config/network-link-type/{*id*} | Update the network link type with id = {*id*}. |
| DELETE | /config/network-link-type | Delete the network link types selected by the query in the request body (or nothing it the request body is empty). |

#### Connection

|  |  |  |
| --- | --- | --- |
| **HTTP Method** | **Path** | **Action** |
| GET | /config/connection | Returns the connections selected by the query in the request body (or all it the request body is empty). |
| POST | /config/connection | Create a new connection. |
| PUT | /config/connection/{*id*} | Update the connection with id = {*id*}. |
| DELETE | /config/connection | Delete the connections selected by the query in the request body (or nothing it the request body is empty). |

### Agent Instance

|  |  |  |
| --- | --- | --- |
| **HTTP Method** | \*\*Path\* | **Action** |
| GET | /config/agent | Returns the agent instances selected by the query in the request body (or all it the request body is empty).. |
| POST | /config/agent | Set a new instance in the ExecEnv with the specified initial configuration. |
| PUT | /config/agent/{*id*} | Update the agent instance with id = {id}. |
| DELETE | /config/agent | Delete the agent instances selected by the query in the request body (or nothing it the request body is empty). |

### Catalog

|  |  |  |
| --- | --- | --- |
| **HTTP Method** | \*\*Path\* | **Action** |
| GET | /catalog | Returns the agents in catalog selected by the query in the request body (or all it the request body is empty). |
| GET | /catalog/ebpf | Returns the eBPF cubes in catalog selected by the query in the request body (or all it the request body is empty). |
| POST | /catalog | Create a new agent in catalog. |
| POST | /catalog/ebpf | Create a new eBPF cube in catalog. |
| PUT | /catalog/{*id*} | Update the agent in catalog with id = {id}. |
| PUT | /catalog/ebpf/{*id*} | Update the eBPF cube in catalog with id = {id}. |
| DELETE | /catalog | Delete the agents in catalog selected by the query in the request body (or nothing it the request body is empty). |
| DELETE | /catalog/ebpf | Delete the eBPF cubes in catalog selected by the query in the request body (or nothing it the request body is empty). |

### Data Collection

|  |  |
| --- | --- |
| **HTTP Method** | **Path** |
| GET | /data |

#### Full Query

|  |  |  |
| --- | --- | --- |
| **HTTP Method** | **Path** | **Action** |
| POST | /data/dsl | Returns the collected data filtered by the query in the request body using the [Elastic DSL](https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl.html) syntax. |
| POST | /data/graph-ql | Returns the collected data filtered by the query in the request body using the [GraphQL](https://graphql.org) syntax. |
| POST | /data/sql | Returns the collected data filtered by the query in the request body using the [SQL](https://www.w3schools.com/sql/sql_syntax.asp) syntax. |

## Installation

1. Prerequisite

* python3
* pip3

1. Clone the repository.

git clone https://gitlab.com/astrid-repositories/wp2/context-broker-apis.git

1. Install the dependencies.

pip3 install -r requirements.txt

## Usage

### Display help

python3 context\_broker-rest-api.py -h

### Production environment

python3 context\_broker-rest-api.py -n production

### Debug enabled in Development environment

python3 context\_broker-rest-api.py --debug -n development

## Extra

See the \*Issues\*\* for *features* in development.

# Cubebeat

Custom Beat of the Elastic Stack to interact with the Polycube-based eBPF cubes.

Ensure that this folder is at the following location: ${GOPATH}/src/gitlab.com/astrid-repositories/wp2/cubebeat

## Getting Started

### Requirements

* [Golang](https://golang.org/dl/) 1.7
* Follows the instructions at [Setting Up Your Dev Environment](https://www.elastic.co/guide/en/beats/devguide/current/beats-contributing.html#setting-up-dev-environment).

### Download the code

mkdir -p ${GOPATH}/src/gitlab.com/astrid-repositories/wp2/  
cd ${GOPATH}/src/gitlab.com/astrid-repositories/wp2/  
git clone https://gitlab.com/astrid-repositories/wp2/cubebeat.git

### Build

To build the binary for Cubebeat run the command below. This will generate a binary in the same directory with the name cubebeat.

mage build

### Run

To run CubeBeat with debugging output enabled, run:

./cubebeat -c cubebeat.yml -e -d "\*"

To run CubeBeat without debugging output enabled, run:

./cubebeat -c cubebeat.yml -e

### Configuration

Cubebeat reads the configuration file (default: cubebeat.yml) that is passed as argument.

This file accepts the common beat configurations as described at [Config file format](https://www.elastic.co/guide/en/beats/libbeat/current/config-file-format.html).

In addition, it accept specific configurations as shown in the next example:

cubebeat:  
 config.inputs:  
 path: config/\*.yml  
 reload:  
 enabled: true  
 period: 10s

The different options will be explained in the following sections.

#### Load external configuration files

Cubebeat can load external configuration files for inputs and modules, allowing you to separate your configuration into multiple smaller configuration files.

On systems with POSIX file permissions, all configuration files are subject to ownership and file permission checks.  
 For more information, see [Config File Ownership and Permissions](https://www.elastic.co/guide/en/beats/libbeat/7.4/config-file-permissions.html) in the *Beats Platform Reference*.

You specify the path option in the cubebeat.config.inputs section of the cubebeat.yml. For example:

cubebeat:  
 config.inputs:  
 path: config.d/\*.yml

Each file found by the path Glob must contain a list of one or more input definitions.

The first line of each external configuration file must be an input definition that starts with - name.

For example:

- name: synflood  
 enabled: true  
 period: 10s  
 polycube.api-url: "http://localhost:9000/polycube/v1/synflood/sf/stats/"  
  
- name: packetcapture  
 enabled: true  
 period: 5s  
 polycube.api-url: "http://localhost:9000/polycube/v1/packetcapture/pc"

It is critical that two running inputs DO NOT have same name. If more than one input the same name, only the first one is accepted; while the other ones are discarded.

When the option enabled is true, the specific cube input periodically interact with the specific Polycube Cube each time interval defined in period making an HTTP request to the URL defined in polycube.api-url.

If the cube is not reachable or there are some error when retrieves the data, cubebeat will continue to work, trying a new connection after a period of time defined in period.

Each period of time, the specific cube input send a new Elastic event to the output as defined in the config file cubebeat.yml

#### Live reloading

You can configure cubebeat to dynamically reload external configuration files when there are changes. This feature is available for input configurations that are loaded as external configuration files. You cannot use this feature to reload the main cubebeat.yml configuration file.

To configure this feature, you specify a path (Glob) to watch for configuration changes. When the files found by the Glob change, new inputs are started and stopped according to changes in the configuration files.

This feature is especially useful in container environments where one container is used to tail logs for services running in other containers on the same host.

To enable dynamic config reloading, you specify the path and reload options under cubebeat.config.inputs section. For example:

cu  
 config.inputs:  
 path: config/\*.yml  
 reload:  
 enabled: true  
 period: 10s

|  |  |
| --- | --- |
| Option | Description |
| path | A Glob that defines the files to check for changes. |
| reload.enabled | When set to true, enables dynamic config reload. |
| reload.period | Specifies how often the files are checked for changes. Do not set the period to less than 1s because the modification time of files is often stored in seconds. Setting the period to less than 1s will result in **unnecessary overhead**. |

On systems with POSIX file permissions, all configuration files are subject to ownership and file permission checks.  
 For more information, see [Config File Ownership and Permissions](https://www.elastic.co/guide/en/beats/libbeat/7.4/config-file-permissions.html) in the *Beats Platform Reference*.

# Conclusions

Body Text First Indent Body Text First Indent Body Text First Indent Body Text First Indent Body Text First Indent Body Text First Indent Body Text First Indent

References

1. Reference Reference Reference Reference Reference Reference Reference Reference Reference Reference Reference Reference Reference
2. Reference Reference Reference Reference Reference Reference Reference Reference Reference Reference Reference Reference Reference
3. Reference Reference Reference Reference Reference Reference Reference Reference Reference Reference Reference Reference Reference

Annex 1: Title of annex 1

If you need annexes or if you need a horizontal layout.

Annex 2: Title of annex 2

If you need more annexes.

1. www.wwww [↑](#footnote-ref-1)
2. In the dash-case (also referred as *hyphen-case* or *kebab-case*) format all the letters are lower-case, the punctuation is not allowed and the words are separated by single dash (or hyphen: -). Example: *exec-env*. [↑](#footnote-ref-2)
3. We use the terms properties, fields and attributes interchangeably. [↑](#footnote-ref-3)
4. In the snake-case format all the letters are lower-case, the punctuation is not allowed and the words are separated by single underscore (\_). Example: *exec\_env\_id*. [↑](#footnote-ref-4)
5. In our architecture the agents are Beats from Elastic Stack. Notwithstanding, the data model refers to a generic agent allowing to possibility to use different types. [↑](#footnote-ref-5)
6. In Elasticsearch, each document is identified by a unique id. For obvious reasons, in the description of the following indices, we omit the description of all the id fields. [↑](#footnote-ref-6)
7. "Getting started with Beats,"[Online]. Available: https://www.elastic.co/guide/en/beats/libbeat/current/getting-started.html [↑](#footnote-ref-7)
8. "Polycube. eBPF/XDP-based software framework for fast network services running in the Linux kernel," [Online]. Available: https://github.com/polycube-network/polycube. [↑](#footnote-ref-8)
9. With nested index, we refer to index that are embedded inside your parent one, https://www.elastic.co/blog/managing-relations-inside-elasticsearch. [↑](#footnote-ref-9)