Introduction
The ATLAS experiment comprises a significant number of hardware and software resources accessed and operated by many users. The Access Manager (AM) implements the resources access restriction to reasonable levels required for the successful running of the ATLAS experiment. Given the large number of ATLAS users, the AM only grants resources usage permission to users according to their current duties. The service is designed on top of the Role Based Access Control (RBAC) model. Every role describes a set of actions granted to users (rules). The rules allow login to a set of machines, to execute some commands with the security privileges of another user (via sudo), or to perform Trigger and Data Acquisition (TDAQ) system specific operations. The roles can be defined using inheritance. Every ATLAS user has a well-defined set of access privileges corresponding to a specific set of assigned and enabled roles. In total, there are several hundred roles and several thousand users.

Given the size and the complexity of the system, a tool to browse and inspect the AM configuration is required. We present the deployment of a new visualization tool named Policy Browser. It is the primary tool for role administrators to inspect all the aspects of the AM configuration via a rich web-based interface.

Back-end Graph Engine
The roles are organized into an inheritance hierarchy reflecting different levels of users expertise. For a given role, it is often necessary to visualize its “inherits from” and “inherited by” graphs.

There are various ways to draw graphs. Comparing them objectively requires defining appropriate metrics. In the case of a dependency graph (directed acyclic), a common approach is the layered graph drawing (also called Sugiyama drawing). Ideally no edge should go upward, and the goal is to minimize the number of edge crossing for readability. As for a large graph this can be a costly operation, the server relies on a dedicated graph engine, Graphviz, for all the graphs generation.

Front-end Graph Explorer
The Graph Explorer is a tool that allows displaying the inheritance hierarchy. It allows browsing large graphs comprising many layers and hundreds of roles with zoom and pan. It permits easy navigation from role to role with a graph morphing animation. Lateral filters are shown in order to highlight subgraph and an advanced search mode allows to fully customize the display.

Back-end Storage and Data Update
The actual AM permissions used by the system are stored on LDAP server. Calculations of their interrelations for visualization require complex manipulations with its data.

The first in-memory prototype resulted in a complicated and hardly maintainable codebase. The second implementation leveraged a relational database (SQLite). It allowed to make the code more readable and to benefit from the query optimizer for a fast response time. Django ships with an Object-Relational Mapping (ORM) that accelerates the initial design of the schema and simplifies data access. A cron job periodically checks if the permissions have changed, and updates the local database accordingly.

Conclusion
The application has been deployed to the ATLAS experiment site for production. It uses live data from the experiment. All the initial requirements have been implemented. Performance-wise, the graph generation is the critical part of the API. Benchmark results show that the current implementation requires an average of 70 ms per graph (CERN CentOS 7, i7-3770 CPU) which should not be perceptible by end users. The preliminary results from informal user feedback are generally very positive. They tend to indicate an intuitive layout and general ease of use.

References
1. RBAC https://csrc.nist.gov/Projects/Role-BasedAccess-Control
2. REST https://www.w3.org/2001/sw/wiki/REST
3. Django https://www.djangoproject.com
4. MVC https://dl.acm.org/citation.cfm?id=50757.50759
5. AngularJS https://angularjs.org
8. SQLite https://www.sqlite.org