

AdHoc Software Version 1.1.0

Table of Contents

What is AdHoc?	2
When to use AdHoc?	2
Putting aside the medical context. What did we learn in general?	2
How do you use it?	3
How does it work?.....	4
Front-end client.....	5
Back-end access control.....	5
How AdHoc exploits pluggable security models?	5
Federated security with Shibboleth.....	7
Centralized security with LDAP	7
Features	8
Single-point management.....	8
Intuitive interface	8
Share easy - anytime.....	8
User access at a glance	9
Resources that can be shared	9
Supported systems	10

What is AdHoc?

AdHoc is GridwiseTech's software framework that simplifies sharing of distributed resources: data, applications, and machines. It comes with an intuitive graphical interface. The resource owners, with a single mouse drag, grant or revoke access to their assets. It takes seconds, without involving an admin or IT expert. Data stays secure within the local IT infrastructure and nothing leaves home, yet trusted people can access it remotely.

Admission rights may also be associated with groups of people. In certain circles such groups, often synonymous with project teams, are called Virtual Organizations.

When to use AdHoc?

The first users of AdHoc were institutions within ViroLab, which is a consortium of hospitals and research institutes united to combat HIV. It must be immediately clarified that there is nothing in the AdHoc architecture to restrict its use to a certain vertical market (such as healthcare) or to a certain type of community (such as clinicians or researchers). However, the ViroLab medical project provides an excellent illustration of what AdHoc can achieve.

Today the market offers numerous drugs for the treatment of HIV (called inhibitors). To maintain a patient's quality of life, a clinician needs to choose the optimal drug, by using a drug ranking application. And how does a drug ranking system suggest a treatment? This is possible by comparing vast amounts of historical data - for instance, 100 thousands of historical patient records. No single hospital is in possession of such quantities of data, but it can be achieved if hospitals collectively share their data.

In the HIV research community, the following scenario is possible:

At a conference, five people meet. Clinicians Laura, David and Teresa work for three hospitals and each of them possesses a database of case histories. Researchers Joanna and Alexander have developed two drug ranking applications. In a discussion, Laura brings up the recent case of her patient X:

Laura: *"We are uncertain what treatment to propose for X."*

Teresa: *"Here's an idea. In our three databases, we may have history of thousands of similar cases. Why don't we conduct a quick experiment: if we all allow Joanna's and Alexander's drug ranker to run over our latest data, it might come up with an interesting treatment suggestion for X."*

Alexander: *"My drug ranker is on my university server, and your data is on your hospital servers. But I have my laptop here, if you grant me access to your remote data, we can do what you propose right now."*

All five open up their laptops to set up an ad-hoc collaboration using their web browsers. Laura defines a group with all five participants. Laura, David and Teresa grant the group members access to their databases. Joanna and Alexander initiate the application search through the remote data and within a few minutes a treatment suggestion is produced. Both applications return the same answer, making Laura's treatment decision simple. Then, Laura, David and Teresa revoke the permissions. The collaboration is dismantled as quickly as it was created – within seconds.

Putting aside the medical context. What did we learn in general?

1. A group of people suddenly decided to share each others' assets
2. These distributed assets (resources) included: data, applications, or hardware
3. Access to these resources has been granted and revoked within seconds, on an ad-hoc basis

4. No administrator has been involved. Resource owners did everything themselves
5. No knowledge of an IT technology or technical skills were needed
6. No specialized software was needed. Users only operated their browsers
7. The entire “project” has been formed, completed, and dismantled within minutes

The above characterizes scenarios in which AdHoc can be useful.

How do you use it?

Sharing resources with AdHoc is extremely simple, especially when compared to the complex distributed systems used in science and engineering today (collaborative environments, cyberinfrastructures, Grids, etc.). Resource owners are free to allow user access to their data sets without involving administrators – clearing the way for easy sharing.

AdHoc integrates with a distributed security model used within an organization or project (one possible model is Shibboleth federated security, which will be described later). Prior to using AdHoc, each user needs to have an identity that allows him or her to log in to the system.

Laura logs in to the online AdHoc interface, using her credentials (such as password or certificate).

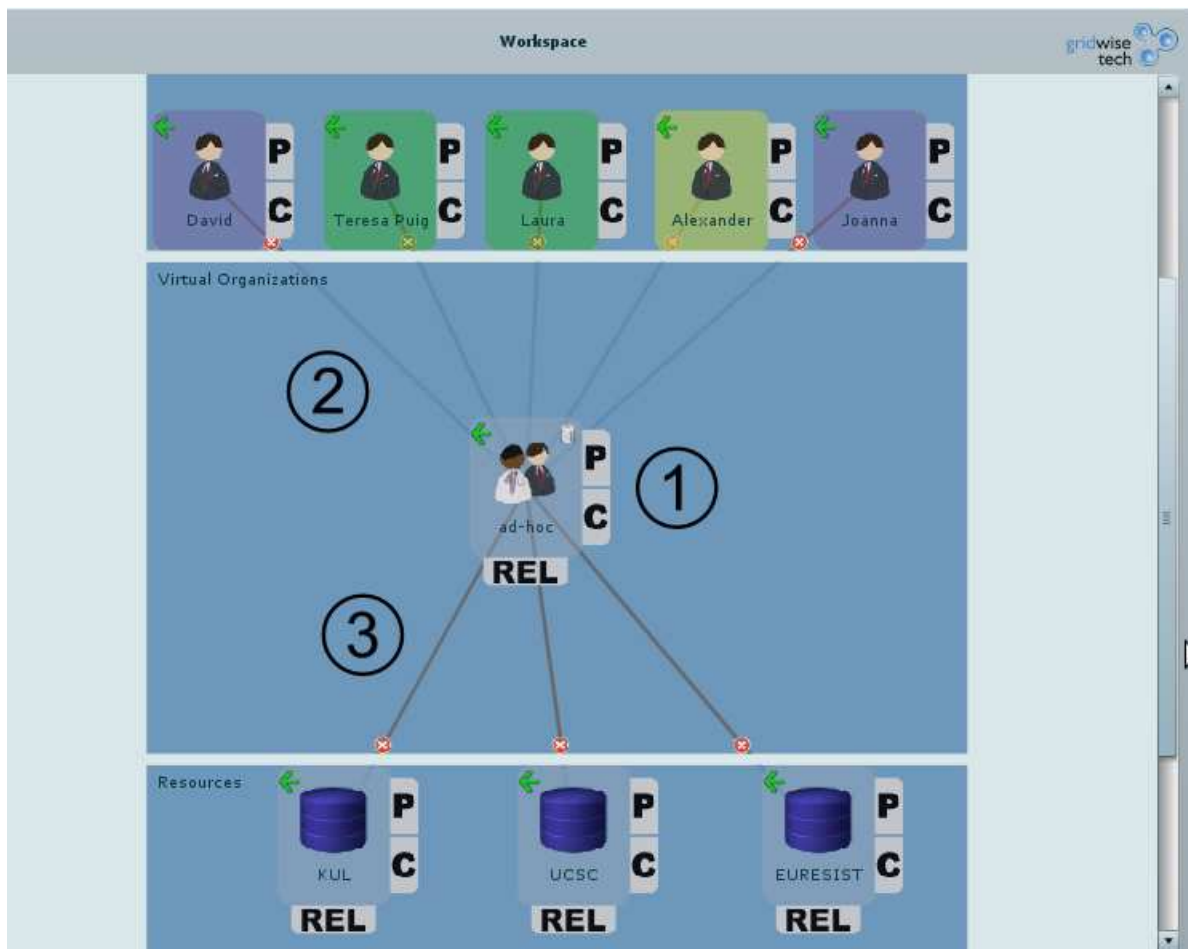


Illustration 1: Resource management in 3 steps

1. She creates a new ad-hoc group, automatically becoming the owner of the newly created group also known as a Virtual Organization (VO).
2. She drags to the group: herself, David, Teresa, Joanna and Alexander. *Note: only Laura can do this, as she created the group and is the group owner*
3. She drags her database to the group, immediately giving all members of the VO the default-level access to the data (Laura can setup the defaults as she likes)

From their laptops, David and Teresa, who have partial administrative rights over their respective databases, log in to AdHoc, drag their databases to the newly created group. This allows everyone in the group access the data. *Note: AdHoc will not allow anyone but the database owner to drag the database to a group.*

Alexander, who wants to run his application over Laura's data, only has to wait for Laura to complete the above scenario before he can start his experiment. It is assumed that Alexander's application uses the underlying security framework credentials (password or certificate) and understands the underlying database schema.

After Alexander's experiment is complete, Laura may want to use the AdHoc interface again to delete the connection between her database and the group. At this moment, the group participants including Alexander lose access to Laura's data.

How does it work?

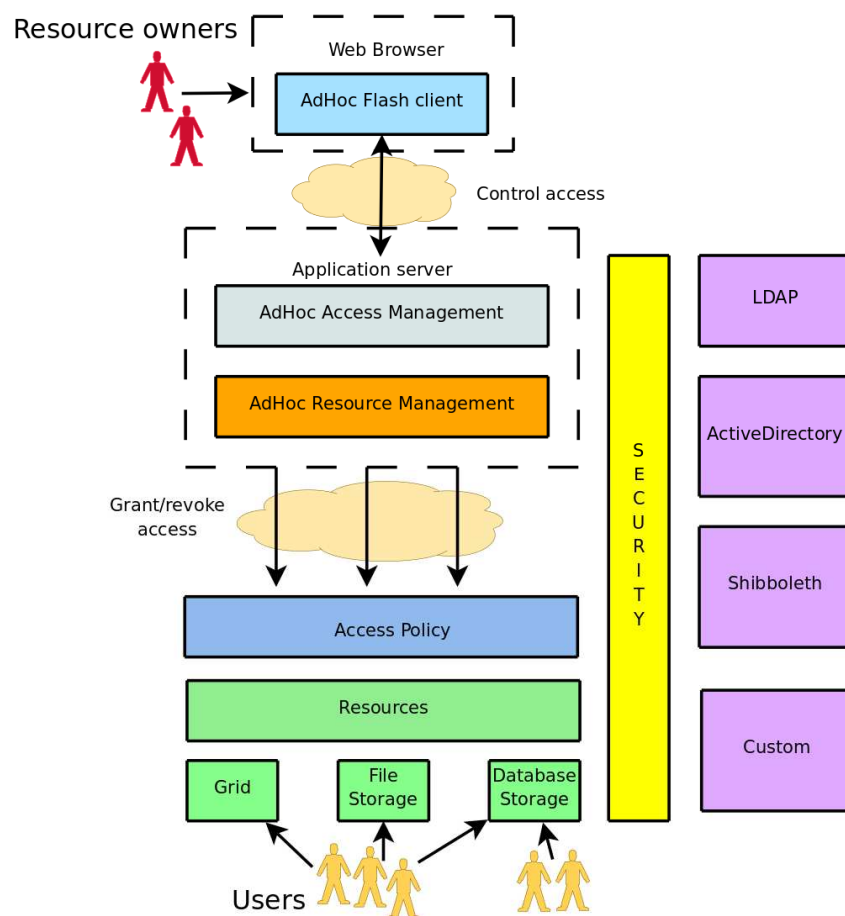


Illustration 2: AdHoc layered architecture

AdHoc is architected from three core layers; AdHoc Web client, AdHoc access management and AdHoc resource management.

The AdHoc Web-based front-end (client) connects users to distributed resources enabling collaborative work. By simply logging into the AdHoc interface explained in the previous section, assets owned by various stakeholders can be secured (user access can be granted or denied) with minimal time and effort. Resource owners can securely access the standards-compliant AdHoc Web interface from anywhere and anytime via their corporate Intranet or public Internet.

In the ViroLab deployment, AdHoc managed the access policies of six resources, including databases and applications, for up to 200 users from four institutions. AdHoc's scalability limits are probably beyond the need of any institution. The back-end can scale to a theoretically unlimited number of users and resources using one of the supported security frameworks and the front-end is limited only by users' ability to interpret the visually displayed data.

Front-end client

The AdHoc client provides a rich graphical user interface based on Adobe's Flex 3 technology making it available as an Adobe Flash-based application. Communication relies on Flex's built-in support for data sources using remote objects. The Flash client runs inside the user's web browser and connects directly to the server-side AdHoc's access and resource management layers which control the granting and revoking of user access.

Back-end access control

AdHoc supports pluggable back-end access control layers that delegate the authentication and authorization decisions to the underlying security framework. AdHoc has been successfully integrated with Shibboleth and LDAP and can be easily adapted to run atop other existing security frameworks such as Active Directory, Public Key Infrastructure, Kerberos, amongst others. It must be stressed that not all security paradigms are equal (explained below) and AdHoc will differ in the functionality that it provides based on the security framework that it is above.

AdHoc access management ensures that users are authorized to log in to the AdHoc Web interface. It delegates authentication and authorization decisions to the underlying security layer. AdHoc resource management is responsible for granting and revoking user access to the given assets. It is implemented using the Resource Gateway that provides a flexible and pluggable interface, allowing virtually any asset to have its access control policies exposed and manipulated.

How AdHoc exploits pluggable security models?

AdHoc can be integrated with federated (Shibboleth) and centralized (LDAP) out-of-the box security models. The two models provide different answers to the question: where to store user data when collaborating?

- Centralized – collaborating member organizations stores all user data within a single infrastructure.

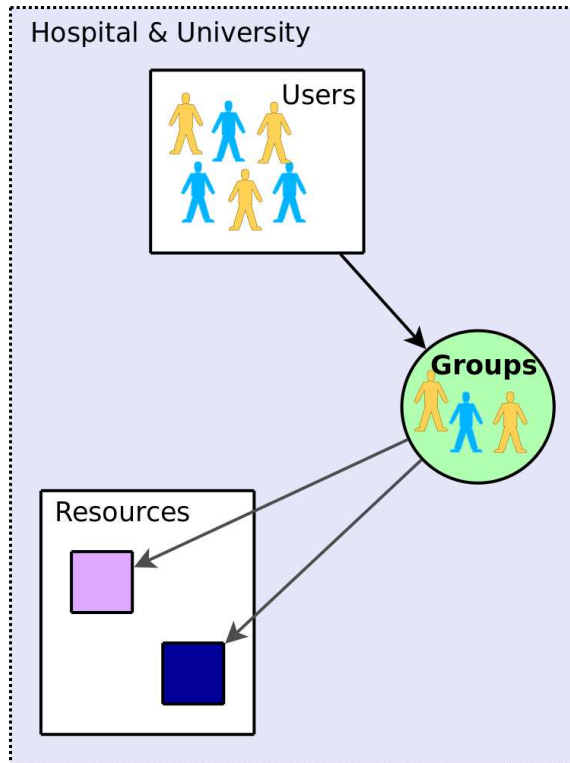


Illustration 3: Centralized security

- Federated – collaborating member organizations stores user data within their own infrastructure and allows others access, based on a trust relationship

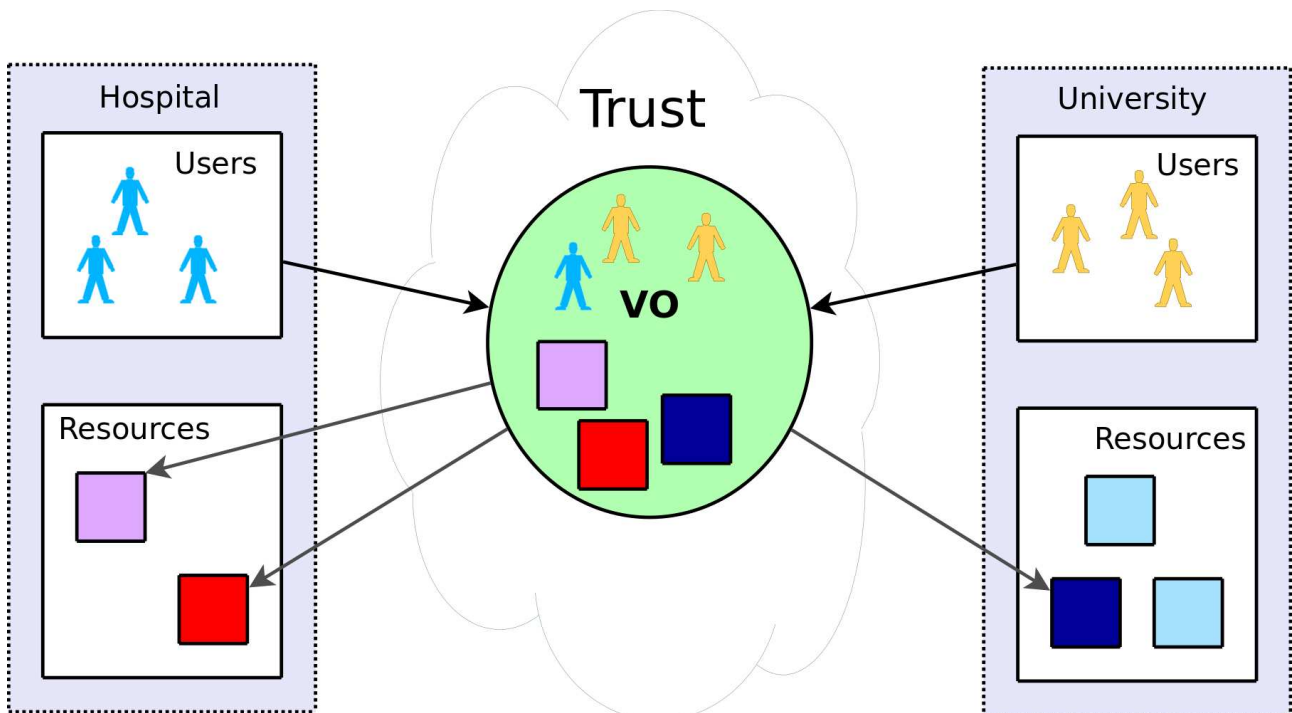


Illustration 4: Federated security

Federated security with Shibboleth

Shibboleth is a software framework, originating from the Internet2 project, developed to provide federated security services for multiple independent institutions (member organizations). The federated paradigm can be conceptualized as Passports. One country issues the document. Other countries accept the identification validation provided by that issuing country. As a citizen, you get one passport only, but it works everywhere because there is trust between users. Without this trust, a Country cannot participate.

In Shibboleth, each Home Organization (HO), such as a hospital, is analogous to a Country and carries the responsibility for securing its own perimeter - granting or revoking access to its resources as well as confirming the identity of its members.

Such a model enables multiple organizations to join a project allowing access to the sensitive resources they are holding whilst maintaining full access control over them. The system scales to a theoretically unlimited number of member organizations with no centralized bottleneck.

Benefits of Shibboleth include user single-sign-on feature, that reduces password fatigue and secure network communication using standards-based security data exchange protocol Security Assertion Markup Language (SAML).

The Shibboleth model is limited by the requirement that each to-be-secured resource needs to be Shibboleth-enabled. However, with a certain amount of custom development, for any asset a Shibboleth gateway can be created, using pre-existing AdHoc libraries.

Examples of such Shibboleth-enabled resources which have been made available through AdHoc include Subversion, a version-control system where access to files can be granted, and PostgreSQL database federation.

Centralized security with LDAP

LDAP is an open application protocol defining industry standards for querying and modifying directory services. These directory services are used to store users and their attributes. Any application may delegate its authentication and authorization decisions to LDAP. In other words, users are authenticated if they are known to the LDAP server in their organization. Authorization decisions can then be taken if the users attributes match criteria, required by the resources.

This model is commonly used to centralize the management of access policies that are controlled on a resource by resource basis. It removes administration complexity and reduces misalignment of business access policy management with its implementation. For example, business managers view policy abstractly as a way to secure information access whereas IT operations will see it concretely in terms of configuration settings. Theoretically unlimited scalability can be provisioned by creating a tree of LDAPs using replication mechanisms.

The main advantages of using LDAP include the availability of free open-source implementations such as OpenLDAP and its wide acceptance with its status as an Internet standard. Users benefit from single-sign-on, same identifying credentials, for all resources. Access to LDAP is supported from almost any computing platform, from any one of the increasing number of available LDAP-aware applications. Minimum customization is required for in-house developed applications to add LDAP support

This model's limitations include: creation of a central store of user information that presents potential synchronization problems with the underlying data. Additionally, a central store of data acts presents greater security issues - with one breach all user data is accessible. It is important to highlight that LDAP is limited to securing resources that are LDAP aware (delegate access control to LDAP).

LDAP is commonly used to control the admission of employees within companies Intranets. Additionally it can be used on an application specific level, for example the Grid workload manager Sun Grid Engine.

Features

Single-point management

AdHoc provides a single Web based visual interface to manage access policies of data, applications, and hardware in distributed infrastructures. Resource owners are in complete control of their assets, granting and revoking access as they see fit, using a single mouse drag.

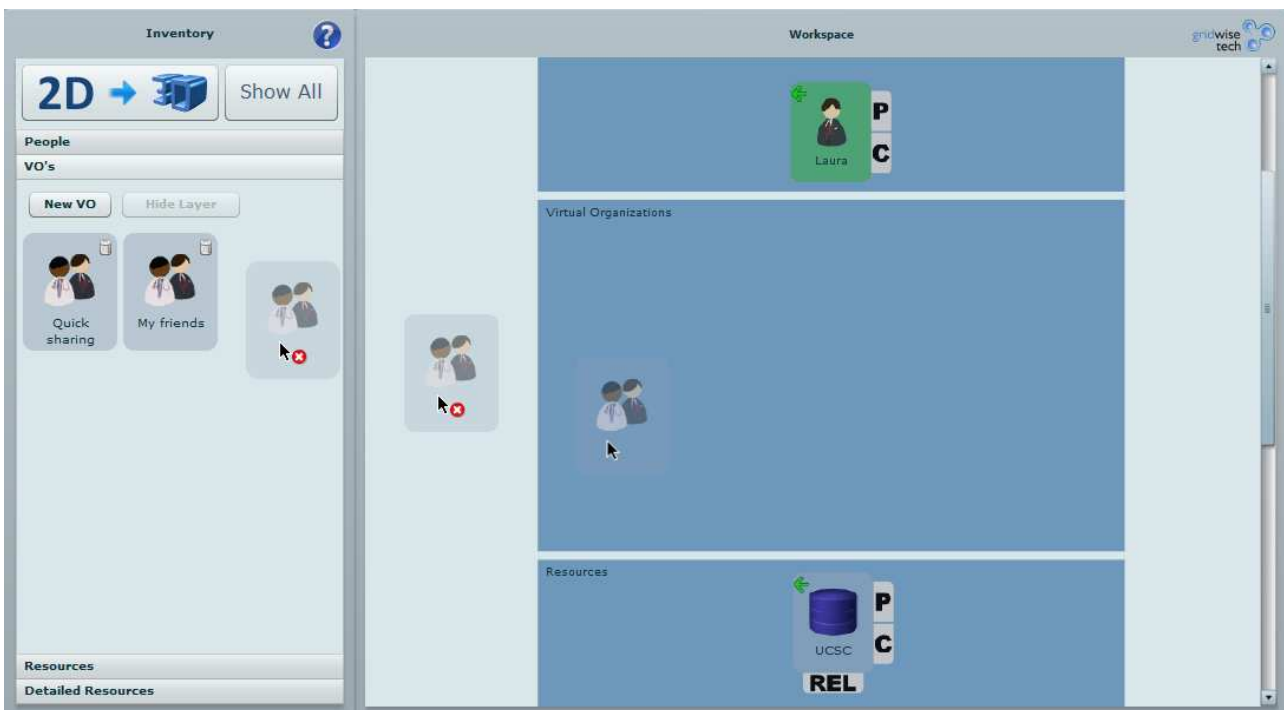


Illustration 5: Drag&Drop

Intuitive interface

No specialist IT technical skills or knowledge is required to operate AdHoc. Simply use the Web interface to drag users, resources and sub-resources onto the workspace and link them together to grant access. Revoke it by effortlessly deleting links - user friendly and quick.

Share easy - anytime

One of the most important benefits AdHoc provides is the full and exclusive control of resources remaining in the hands of their owners. It's them, not the systems administrators, who have the power to grant or revoke access to their own data sets. This can be performed rapidly, anytime and from anywhere.

User access at a glance

View all access policies over the entire IT infrastructure in an easy-to-read 3D interface. Users, groups, resources and sub-resources all reside on their own layers with simple connections between the entities indicating access. No more logging into multiple services to gather the required information about which user is entitled to use which resources.

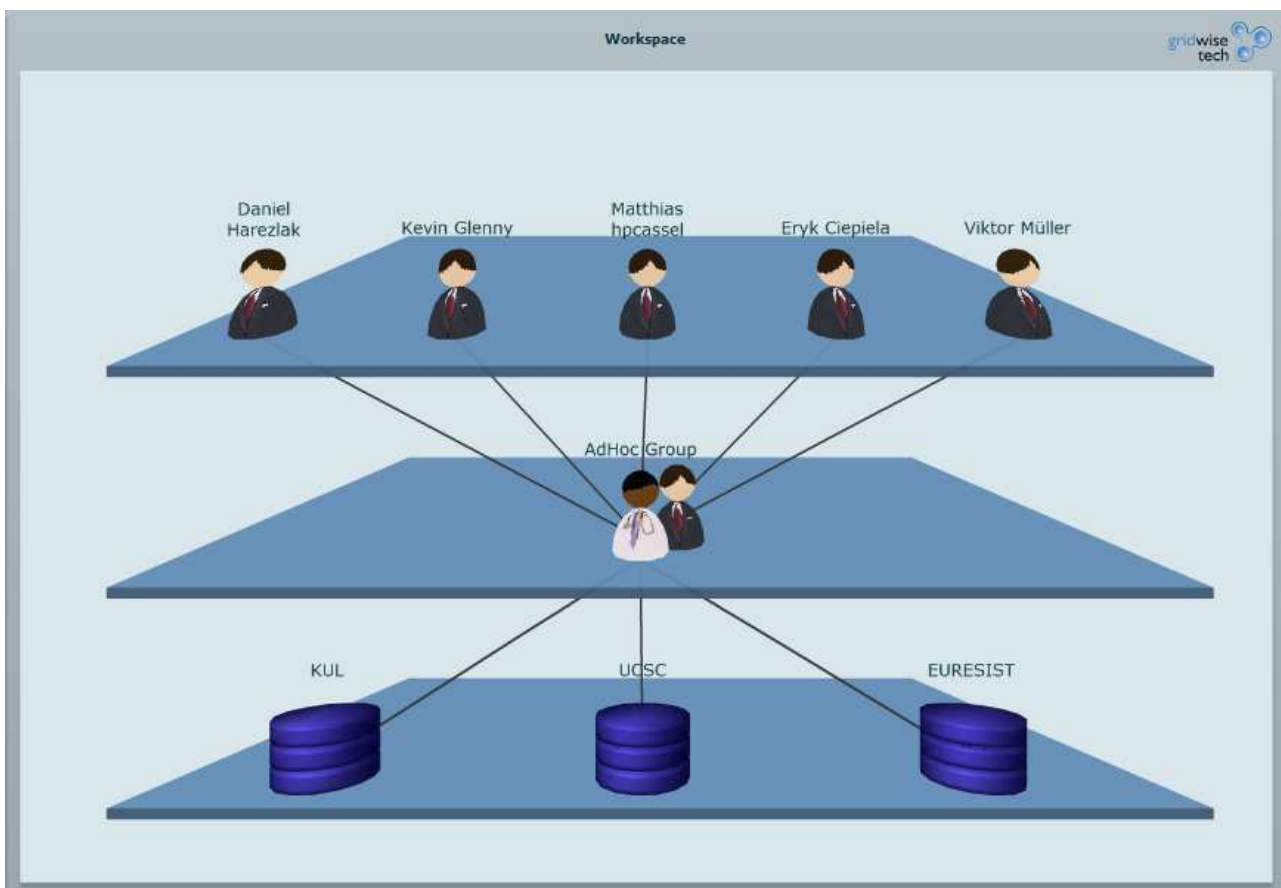


Illustration 6: Rich 3D

Resources that can be shared

The security model used defines how resources access policies are managed. For example, if using Shibboleth federated security, assets must be Shibboleth-enabled. When using LDAP centralized security, assets must be able to delegate their access control to LDAP.

As mentioned earlier, example resources that can be integrated with Shibboleth include:

- Databases - PostgreSQL federation
- File data stores - Subversion source control

Example resources that can be integrated with LDAP include:

- IT infrastructure – Linux OS
- Databases – Oracle, PostgreSQL federation

- Grid workload manager – Sun Grid Engine
- CRM – SugarOS
- Collaboration tools – Mantis bug tracking, docuWiki, WordPress blog

The above is not an exhaustive list of assets that can be shared. With a certain amount of custom integration effort, AdHoc can support any resource that has authorization/authentication concepts built in, through other security frameworks (Active Directory, Public Key Infrastructure, Kerberos, amongst others), access policy files or in-house developed mechanisms.

Supported systems

Operating System	Web Browser
Linux	Firefox, Opera
MS Windows	Firefox, MS Internet Explorer 8, Opera, Safari
Mac OS X	Firefox, Opera, Safari

Table 1: AdHoc client requirements

Operating System	Application Server	Portal Container (optional)
Linux	Apache HTTP Server & Apache Tomcat	JSR 168 / JSR 268 compliant portlet containers including GridSphere, JBoss Portal, Liferay, IBM WebSphere Portal Server, or Oracle IAS Portal
Windows	<i>On request</i>	<i>On request</i>
UNIX	<i>On request</i>	<i>On request</i>

Table 2: AdHoc sever requirements