Harry Sileoni

Design and Development of Software Solution to Improve Google Drive Shared Folders

Helsinki Metropolia University of Applied Sciences

Master's Degree

Information Technology

Master's Thesis

23 November 2015

Helsinki Metropolia University of Applied Sciences

Author(s)	Harry Sileoni
Title	Design and Development of Software Solution to Improve Google Drive Shared Folders
Number of Pages	42 pages + 4 appendices (5 pages)
Date	23 November 2015
Degree	Master of Engineering
Degree Programme	Information Technology
Instructor(s)	Mika Lausamo, Project Manager Ville Jääskeläinen, Principal Lecturer

The purpose of this thesis was to design and develop a piece of software to improve Google Drive shared folders in organizational use. The main reason why the topic was chosen, is that organizations that use Google Drive as a centralized data storage are especially vulnerable to a potentially vast amount of work in case of an accidental deletion of items.

The thesis proposes the design and development of software which monitors shared folders and sets file ownerships to one centralized user to allow an easy way to restore files to a shared folder which is monitored.

The software was designed and developed in the agile development method, which allowed and encouraged the requirement specification to change during the development phase.

A finished product was designed, developed and tested in a timeframe of nine months, and the software is in active pilot use on 11 shared folders of various organizations.

Keywords	Google Apps, Google Drive, file server, PHP, shared folder

Tekijä Työn nimi Sivumäärä Päivämäärä	Harry Sileoni Design and Development of Software Solution to Improve Google Drive Shared Folders 42 sivua + 4 liitettä (5 sivua) 23. marraskuuta 2015
Tutkinto	Master of Engineering
Koulutusohjelma	Information Technology
Työn ohjaajat	Mika Lausamo, projektipäällikkö Ville Jääskeläinen, yliopettaja

Tämän lopputyön tarkoituksena oli suunnitella ja kehittää ohjelmisto, joka parantaa Google Drivessä sijaitsevien jaettujen kansioiden toimintaa yrityskäytössä. Pääsyy aiheen valintaan oli se, että yritykset jotka käyttävät Google Driveä keskitettynä tiedostovarastona, ovat erityisen alttiita potentiaalisesti suurelle työmäärälle, mikäli jaetusta kansiosta poistetaan tiedostoja tai kansioita vahingossa.

Lopputyö tarjoaa suunnitelmaa ja toteutusta ohjelmistosta joka seuraa jaettuja kansioita ja asettaa näiden sisältämien tiedostojen sekä kansioiden omistusoikeuden keskitetylle käyttäjälle, täten mahdollistaen helpon keinon palauttaa vahingossa poistettuja tiedostoja sekä kansioita.

Ohjelmisto suunniteltiin sekä toteutettiin käyttäen ketterää ohjelmistkehitysmenetelmää, joka mahdollistaa ja kannustaa muuttamaan tarvekartoitusta myös kehitystyön aikana.

Valmis tuote suunniteltiin, toteutettiin sekä testattiin yhdeksän kuukauden sisällä ja ohjelmisto on tällä hetkellä koekäytössä usealla yrityksellä, 11 jaetussa kansiossa.

Avainsanat	Google Apps, Google Drive, file server, PHP, jaettu kansio

PREFACE

When I started my Master's studies at Metropolia University of Applied Sciences in 2014, I immediately started to figure out a topic for my upcoming Master's thesis. As Google Apps was one of the main tools we used in my workplace, I figured it could provide an interesting topic in some form. As all software, Google Apps also proved to include design flaws partially addressed by at least one third party vendor. Because the vendor did not include a solution which would have been feasible to our company and most of our customers, I decided to develop a software solution which would be feasible for reselling and provide a good topic for my thesis.

The design and development was more challenging than anticipated, but so was the actual report in a thesis form, as most of the development was done before the written report. All figures, tables and appendices were made completely by the author. All elements in this report were made with Google Docs, Google Sheets and Google Drawing, excluding screenshots. Due to limitations in Google Docs, the final document was rebuilt with Microsoft Word.

Last but not least, I'd like to give a loving thank you for my wife Tiina for taking care of our daughter, while allowing me to work on this project besides my regular day job.

Contents

Abbreviations and acronyms

Authentication	The process of defining whether or not someone or something is who or what it claims to be
Authorization	The function of defining whether or not someone or something has permitted access to something
Cron	A job-scheduler software utility, which is widely used in Linux based operating systems to run scheduled tasks
Crontab	A file which defines tasks for Cron to run in any given time. Crontab is usually available on a per-user or per-system basis. Usually both at the same time.
CSS	Cascading Style Sheets, a language for defining text and layout formatting in websites and web applications
Download	A term used to describe the function of fetching data from a remote system to a local system
Delta sync	A scan and monitoring of all files which have changed since the previous full or delta sync
Full sync	A recursive scan and monitoring of all files within a shared folder
GNU GPL	GNU General Public License, a widely used license for free software. Indicates that a software is free to use, edit and distribute by anyone.
Google Apps	A web-based office suite from Google, including email, calendar, contacts, a file storage system, a word processor, a spreadsheet application, a slideshow application and many other applications
Google Apps for Work	The enterprise version of Google Apps, including more advanced email, company personalization with the exclusion of advertisements
Google Apps Marketplace	The official website for activating third party applications for Google Apps
Google Drive	An internet-based service for storing files
Google Drive REST API	Google Drive Application Programming Interface - a set of protocols to interact with Google Drive programmatically. Runs over HTTP.

Google Drive SDK	Google Drive Software Development Kit - a set of tools for to integrate software with Google Drive
HSWDrive	The name of the developed software
HSWDrive instance	A single occurrence of the software, intended to monitor one shared folder.
HTTP	Hypertext Transfer Protocol, a non-secure protocol used commonly in web browsers to transfer website information
HTTPS	HTTP over SSL, a secure version of HTTP, which encrypts the transferred data
InnoDB	A database storage engine, which can be found in MySQL 5.5 and later
MySQL	An open source database server software
Oauth	An open standard for authorization over the internet
Oauth 2.0	The successor of Oauth. Provides more security and is considered to be easier to use and understand.
Oauth access token	An alphanumerical string, which represents a code to allow communication between a client and server
PHP	PHP Hypertext Preprocessor. A widely used scripting language.
TLS	Transport Layer Security, a cryptographic protocol to encrypt data. TLS is the successor of SSL. The latest released version of TLS is 1.2.
RAM	Random-access memory. A form of temporary file storage in computers. Provides significantly faster read-write speeds than a regular hard drive, but does not allow permanent storage.
Upload	A term used to describe the function of sending data from a local system to a remote system
vCPU	Virtual Central Processing Unit. Used in virtualized servers to indicate the amount of CPU's within the virtual server. Physical CPU's are usually shared between virtual servers.

1 Introduction

This thesis focuses on the design and development of a piece of software, which improves the functionality of Google Drive shared folders for organizational use. The goal of the project was to design and develop a software solution which monitors and sets file ownerships within a shared folder in a way that accidentally deleted files are easy to restore. The software is to be sold as a service. The finished product was named HSW-Drive.

1.1 Background

The study discusses the use of shared folders in Google Drive. Google Drive is an internet-based computer file storage service (also known as cloud storage) developed by Google Inc. It is part of a wider suite of web-based applications, called Google Apps, which is reasonably priced for companies (Google Apps for Work) and available free for consumers (with a free Gmail account) and education facilities (Google Apps for Education). The current service model was launched on 2012, although the same base has existed since 2010, when it was still known as Google Docs. Google Drive is currently used by roughly 240 million active users. (1)

The basic idea of a cloud storage is that a user has his/her own password protected account, which the user uses to add, modify or delete his/her personal files. With the use of cloud storage, the end-user does not have to carry any removable mass media (CD / pendrive / hard drive) with him/her as long as he/she has access to the internet. Because internet connectivity has spread so wide in today's world, it is usually unnecessary to carry the files with you physically.

Due to the fact that a user can share a file or folder to other users or user groups in Google Drive, it is also possible to use Google Drive as a centralized enterprise data storage. To establish a centralized data storage, a user must create a folder and share it to all other users which require access to the same files.

When a user adds files to a shared folder, the ownership of the files remain to the user who added the files. Otherwise any user could fill up the disk space of another user. When a file is deleted, only the file owner can restore it from the trash bin. Because of this logic, a problem arises when someone accidentally deletes a file from the shared folder. The original owner has to be found and the owner must restore the file back to the shared folder.

This could potentially cause a lot of manual work in case a whole file structure has been deleted. This challenge can be bypassed by creating a software solution which automatically monitors and logs all the changes done within a Google Drive shared folder. By changing the ownership of each file to a single dedicated user account, the files are easy to restore from one user interface in case of an accidental deletion.

1.2 Technology Problem

Although Google Drive provides a platform to share files within a company, there is one major problem in the concept: when a file is deleted from a shared folder by someone who is not the original owner of the file, it disappears from the shared folder and is left orphaned. An orphaned file is a file, which has an owner, but does not have a location, so it can only be found by the owner of the file by conducting a search. It cannot be found in any particular folder, not even in the trash. In case of an accidental deletion of large folder structures, this causes a significant problem in restoring the deleted files and folders. This has happened to the case company and its customers, thus a solution to avoid it in the future is required. (2)

As to this date the case company has found only one widely recognised commercial service to address this problem: AODocs. Although AODocs is much more than just a system to prevent accidental deletions, the case companies required functionality of AODocs is to convert any Google Drive folder to a controllable file server with advanced options. The problem with AODocs is that since they released a new version in the last quarter of 2014, it has not been possible for users to delete or add files to a shared folder thru the Google Drive interface, thus forcing the users to use the AODocs interface or a separate browser plugin for adding and deleting files.

The case company used AODocs for a trial period of 30 days at the end of 2014 and was almost ready to purchase it, but after noticing some properties which would have required the end-users to change working habits, a questionnaire was conducted within the company to decide whether AODocs was suitable or not for this environment. The questionnaire included the following questions:

- Would installing a separate browser plugin cause significant labour?
- If yes, would it cause significant labour to use the AODocs web interface for handling files, even if it currently does not support drag and drop?
- Would abandoning the Google Drive synchronization client and switching to a completely browser based solution cause significant labour?
- What applications are used to access Google Drive?
- Provide free comments.

Based on the results of the survey, 8 users out of 9 answered yes to the first three questions. All users preferred Google Chrome as the initial application to access Google Drive, but some users also used the Google Drive app on mobile devices and Google Drive synchronization client to have a local copy of the files. At this point it was clear that the employees did not like the idea that each and every user in the organization should either install a browser plugin or learn a new system to add, modify or delete files in a shared Google Drive folder. According to the free comments section of the questionnaire, AODocs was taking too much control over the user experience, which led to the conclusion that AODocs was not a feasible solution for the case company.

1.3 Objective and Outcome

The objective of this study was to explore the file deletion logic in Google Drive's shared folders and to suggest a solution for easy restoration of accidentally deleted files. The outcome of this project includes the following:

- Software to allow easy file restoration in case a user accidentally deletes a file or folder from a shared Google Drive folder

- Documentation for end-users

The software must be easy to implement on other folders and organisations, thus making it easy to sell as a service. Because this type of functionality requires continuous monitoring of the folders and files, the finished software must run on a dedicated server.

1.4 Project Methodology

The aim of this project was to design and develop a software solution which monitors and sets file ownerships in a way that accidentally deleted files are easy to restore. After the initial requirements (development server and Google Developer credentials) were obtained, the software was designed and developed using the agile software development method and object-oriented PHP language. Most of the programming was done with the combination of TextMate and Transmit. TextMate was used to write the code and Transmit was used to upload changes to the server. Programming was done in various locations, mostly in the developer's home or workplace, due to the fact that software design and programming can be done anywhere with a laptop and an internet connection.

The process of the project is described in the following steps:

1. The various ways how a file can be deleted and restored in a Google Drive shared folder was explored to design a program logic which has minimal impact on the user experience of Google Drive.
2. Based on the investigation of step 1, various solutions were tested for the best possible solution.
3. The solution was designed and developed in the following order:
    a. A dedicated virtual server to run the software was created in the Google Compute Engine cloud platform.
    b. Google Developer Console credentials were obtained by signing in to a Google account and following Google’s guidelines.
    c. A shared Google Drive folder was created for testing purposes.
    d. The software was designed and developed using Google’s client libraries, PHP programming and a MySQL database engine.
4. The developed software was tested in terms of functionality, speed and ease of use.

After the software is stress-tested and proven functional on various pilot companies for a time period of at least six months, it is ready to be submitted to the Google Apps Marketplace for selling. Due to time restrictions, this part was not in the focus of the present study.

The Google Developer Console credentials were used to gain access to the Google Drive API. Google Drive API was used to send commands and perform queries to Google Drive. The actual software was be developed using the PHP language. PHP was chosen due to the developers vast experience in it and because Google provides PHP client libraries for the Google Drive API.

1.5 Scope and Structure

This project was limited to Google Apps for Work and Google Apps for Education environments so it does not work with free Gmail accounts due to limitations of Google Drive API. Unlike in AODocs, a separate interface will not be created for handling files. All file and folder handling and restoration of deleted files can be done directly in Google Drive, although a separate file restoration interface was made for easier restoration of large file structures. All files saved in the shared folder must be owned by someone within the company of the shared folder. Files which are owned by external collaborators, were not included in the scope of the software, although this might be possible in a future release.

Section 2 explains more about the technical aspects of Google Drive, Google Drive API and the programming language of the developed software. Section 3 describes the requirements for the developed software to function. Section 4 describes the program logic and includes illustrating tables for the logic and database structure. Section 5 describes the evaluation for the finished software. Section 6 lists all encountered problems while developing the software. Section 7 expresses the current and future possibilities for licensing and distribution of the software and Section 8 includes measurement results and analysis of the finished product and discusses of the possible future insights.

2 Technology behind Google Drive and Developed Software

To better understand the technical aspects of Google Drive and the development of the software solution, the used technologies and software logic are described more thoroughly in the following sections.

2.1 Google Drive

Google Drive is an internet-based computer file storage service (also known as cloud storage) developed by Google Inc. It is part of a wider suite of web-based applications, called Google Apps.

Google Drive was introduced in April 2012, even though the core service did exist before this as a file storage system for the Google's web-based office suite. The early web-based office suite was called Google Docs and it included functionality for word processing, spreadsheets and presentations. After the launch of Google Drive these were separated into three individual apps: Google Docs, Google Sheets and Google Slides respectively.

2.2 Structure and Use of Google Drive

Google Drive is used to store and share files and folders in the cloud. It is directly integrated to Gmail (Google's email system), which enables file sharing directly via the email interface. The use of Google Drive for file sharing addresses the widely known problem of large email attachments filling mailboxes. By only sending a link to the file, the recipient's mailbox does not get filled by large attachments and the recipient is able to download the attachments on demand. The downside of this method is that the file can only be downloaded as long as the sender does not delete the original file.

By having files in a cloud storage, there is no need to carry any physical storage media. All changes made to the files are usually viewable by all privileged users and devices within a few seconds delay either from the web interface or in a locally synchronized folder, when using a separate synchronization client software.

A noticeable difference between Google Drive and many other Cloud Storage services is that Google Drive uses tags instead of folders. Although files appear to be inside folders, they are actually just tagged with the folder names. By using tags, it is possible to have a file located in various locations at the same time. This method is very similar to using hard links in a Unix-based operating system, such as Linux. The only practical difference between a hard link and tag is that hard links are done in the operating systems file system level, while tags are done in a separate database. By utilizing tags instead of traditional folders, it is possible to save a considerable amount of disk space, because there is only one copy of any given file, even if it is located in various locations of various users. (3)

Although having a separate backup system is important, it is not that crucial in such as vast system as Google Drive, because of the various safety and security implementations. All data is replicated between various data centers around the globe and each data center is equipped with emergency power generators, which enable data integrity in case of a natural catastrophe. The data centers operate on custom built servers which are exclusively built by Google and run a stripped version of the Linux operating system. (4)

All overwritten files are automatically added to a separate revision history, which allows file restoration in case of an accidental change in a file. Deleted files are added to the trash bin before final deletion, which allows users to restore them in case of an accidental deletion. Some files are even recoverable by Google Support and the organization administrator for a few days after emptying the trash bin. For extended data integrity, it is possible to take data backups from Google Drive to a local computer or a third party cloud storage, such as Backupify. (5) (6)

2.3 Google Drive Synchronization Client

Google Drive was originally intended for web browser use due to its integration with Docs, Sheets and Slides. As the number of users increased, Google released the synchronization client for Windows and Mac OS X. The idea of the synchronization client, is to have a local copy of all or selected files within Google Drive. This way users are able to access files without opening the web browser. According to the experience of the case company, many users face the problem of not understanding that deleting or moving items from a locally synchronized folder also deletes the items from Google Drive. This

functionality is one of the main drivers for this project. As of today, there is no official version of the synchronization client for Linux.

2.4 Google Drive File Permissions

All files and folders in Google Drive are always owned by someone. Even though teams and companies use Google Drive to collaborate within shared folders, the files within shared folders are by default always owned by the person who creates or uploads the file. Each file consumes disk space only from the owner, regardless of the file location. The file ownership can be transferred to another user within the same organization by the current owner or programmatically by using Google Drive REST API. After a file has been transferred to a new user, it will free the consumed disk space from the previous user and consume it from the new user.

Having files with different owners in a shared folder can create a problem, when a user account is deleted. Because the files are owned by the user, deleting a user will also delete the files owned by the user within a shared folder. Another possible problem arises when a file is deleted from such a shared folder. Because the file is owned by some user, it can only be found from the owners recycle bin, which makes it difficult to restore in a multi-user environment.

2.5 File Locations in Google Drive

Each Google Drive user has a “My Drive” folder, which is also called the root or root folder. Google Drive allows files and folders to be shared and to exist in multiple locations at the same time. These locations are called parents in the Google Drive REST API. Having files and folders in multiple locations simultaneously is a handy feature, but in theory allows a recursive loop. A recursive loop occurs when a parent folder is added into its child folder. The problem of a recursive loop is that a folder structure is infinite, because the child folder includes its parent folder. In case a folder structure with a recursive loop would be synchronized as a local copy to the end users computer, it would cause an infinite number of items, which would rapidly fill the computer’s hard drive. Besides filling the end users hard drive, it would also make deleting the folder structure troublesome, because most file systems start the deletion process by calculating the

number of items to be deleted. If the number of items is infinite, the calculation process also takes an infinite amount of time to advance, thus never completes or causes an error message. A graphical representation of a recursive loop can be seen in Figure 1.

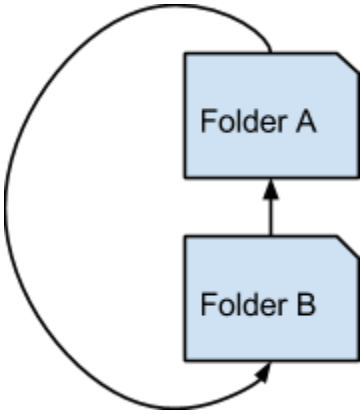


Figure 1. A graphical representation of a recursive loop in a folder structure

Each user can also add shared files and folders to the user's own root folder for easier access. The root folder of each Google Drive is called "My Drive". Even though a shared folder is owned by someone else, each collaborator can add it into any folder under My Drive, as long as the user has write access and the location does not cause a recursive loop. In case the user does not have write access or tries to create a recursive loop, the operation fails and an error message is shown to the user.

When adding files to a shared folder, the file access permissions are inherited from the parent. Thus when adding items to a shared folder, it is not required to explicitly share them to other users, unless the targeted user does not have access to the shared folder. Google Drive automatically increments access permissions to any object within a shared folder to match the access permissions of the parent folder. Manually added permissions are not removed in this process.

2.6 Deleting Files in Google Drive

Only the owner of files and folders can delete them permanently. If the owner deletes files or folders, they are first moved into the owner's trash. Deleted files can be restored as long as the trash section is not emptied. Trashed files are not accessible by collaborators.

In case a user with write access deletes files or folders from a shared location, the items will only be removed from the current parent location, but not from the owners My Drive, which is considered as the root folder. If the shared location was the only location for the deleted item, it becomes orphaned. An orphaned file or folder does not exist in any folder, because it does not have any parent locations. An orphaned item is not considered to be deleted or trashed, it just no longer exists in any location. The only way to find an orphaned file or folder, is to conduct a search in the Google Drive's web interface. Finding an orphaned file can be challenging, but it becomes even more burdening in case the file's owner is unknown. The visibility of files and folders in My Drive is illustrated in Figure 2. (2)

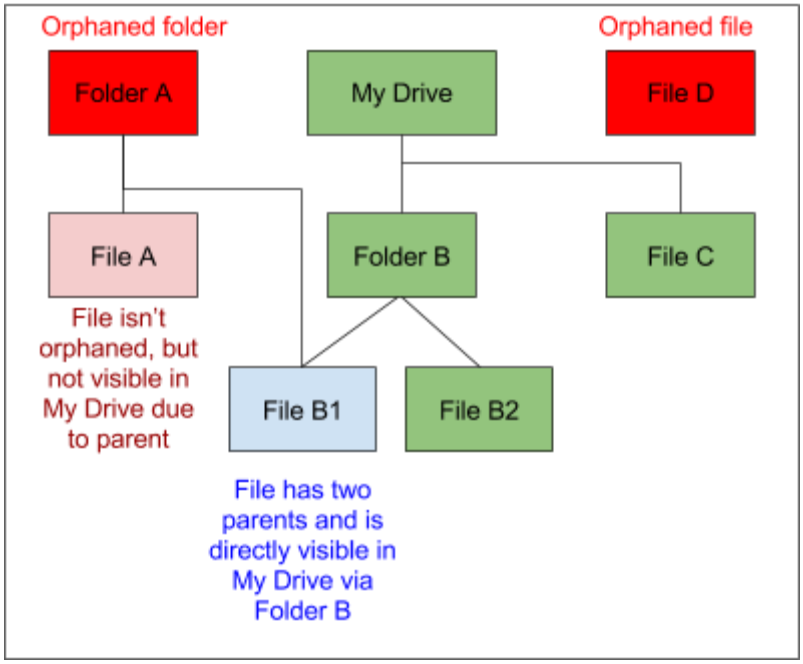


Figure 2. An illustration of orphaned files in Google Drive

As illustrated in Figure 2, files which reside in an orphaned folder (such as File A in Folder A) are also potentially hard to find due to the parent folder being orphaned. Even though File B1 is situated in the orphaned Folder A, it can still be easily found due to its existence in Folder B, which is situated in My Drive. Although having files directly in the root of My Drive is not considered as best practice for file organization, it is still directly visible to the user.

When someone deletes a single-parented item from a shared folder, one of the two situations occurs: If the deleting user is the owner of the item, the folder will be added to the owner's trash, from where it is no longer accessible to collaborators in any form before the user restores the item. If the deleting user is not the owner of the item, the item is deleted from the current location, but is still accessible to the owner and other users who are explicitly granted access via manually sharing the item. Even though access is still possible, it is still as an orphaned file and needs to located by searching via the web interface of Google Drive.

2.7 Google Drive REST API

Google Drive REST API is an essential part of Google Drive SDK, which includes all the required commands, protocols and tools to interact with Google Drive. By having an API, developers around the world are able to extend the functionalities of Google Drive. The API officially supports the following languages: Go, Java, JavaScript, .NET, Node.js, PHP, Python & Ruby. Besides these languages, it is also possible to use the API with any other programming language, as long as the language supports REST calls over HTTPS. The downside of using an unsupported language is that there is very little documentation to support development for such languages. (7)

Google provides up to one billion (1 000 000 000) free Google Drive API requests per project on a daily basis and more can be requested. The number of API requests is limited to 50 requests per second per user. Most operations are done as the file storage user of an organization. Depending on the requirements, additional API requests might be prone to financial expenses. Google Drive API consists of 13 different resources to access Google Drive functionalities. A resource can be thought as an object, in terms of programming. The available resources are: Files, About, Changes, Children, Parents, Permissions, Revisions, Apps, Comments, Replies, Properties, Channels and Realtime. These resources are explained more thoroughly in the following sections.

Files resource

The Files resource includes methods to list, view, follow, create, modify and delete files and folders. When using the listing method, this resource cannot be used directly to list files within a folder. The listing method works in a very similar way than using the search

bar in the web based Google Drive interface, with the addition of having more options, such as the ability to limit a search only to Google Photos or reducing the number of returned objects.

About resource

The About resource is very simple in terms of methods, because it only includes one method to get information about a user, such as the used amount of data in Gmail, Google Drive and Google Photos. The About resource also includes a vast amount of Google Drive related information, such as sharing policies, some Google Drive API settings and the file ID for the user's root folder (also known as My Drive).

Changes resource

The Changes resource includes methods to get, list and watch changes done to files and folders. Every time a change has been done in the user's Google Drive, a new change item is created, which includes basic information about the file which has been changed. The use of a change log for monitoring file modifications is considerably faster than recursively scanning for each file in a folder structure, because only changed files are listed.

Children resource

The Children resource includes methods to list, remove and add children objects to and from a folder. To use the Children resource, a file ID of a parent folder must be given before being able to use this resource. Also folders are considered as files in the Google Drive API, thus folders also have a file ID. This is the most used resource in this project, as it is required to list files within folders, while checking for user ownerships. This resource cannot be used to delete or trash files or folders. When deleting a child object, it only deletes the object from the selected parent. To actually delete a file or folder, the Files resource must be used.

Parents resource

The Parents resource provides methods to list, add and remove parents (locations) of any item. Like the Children resource, this resource cannot be used to actually delete files

or folders. It is only used to add or remove a parent from any child object. This resource is utilized as part of checking if a file is situated within a shared folder.

Permissions resource

The Permissions resource includes methods to list, modify and remove permissions from items. Each permission setting for every item has its own permission ID. The permission resource defines the permission role of an item. In case of modifying existing permissions for a file or folder, the current permission ID for the given item and user combination must be obtained by listing the permission objects for a given file. In case of changing the ownership of an item, the user needs to have a permission object assigned to the given item. If the user does not have any permission objects assigned to a given item, one needs to be created, before the user can be converted as the owner of the given file. This resource is one of the most used resource of this project, as the primary focus is to centralize the ownership of files to one storage user.

Revisions resource

The Revisions resource includes methods to list, modify and delete revisions of a file. Revisions are stored automatically for each file by Google Drive when a file has been updated. Although the possibility to restore file contents may be an option in the future, this project does not currently focus in restoring file contents, thus this resource is not utilized at all.

Apps Resource

The Apps resource includes methods to list and view apps, which have been enabled in the users Google Drive. As this project is not focused in listing for existing external Google Apps, it is not used at all.

Comments resource

The Comments resource includes methods to list, modify, add and delete comments to file created in Google Docs, Google Sheets or Google Slides. Comments are not currently supported in other types of files. As this project does not include functionalities to view or alter comments in separate files, it is not used at all.

Replies resource

The Replies resource includes methods to list, modify, add and delete replies to comments in file created with Google Docs, Google Sheets or Google Slides. As this project does not include functionalities to view or alter comments in separate files, it is not used at all.

Properties resource

The Properties resource includes methods to list, create, modify and delete custom properties for drives stored in Google Drive. The properties can be public to all apps or private to just one app. As this project does not focus in file properties of separate files, it is not used at all.

Channels resource

The Channels resource is a resource, which is created when a File resource is set to be followed with the Watch method. The Channels resource can be stopped with the Stop method, thus ending the watching or following of a file. Due to requirement of following all items within a given structure, this resource does not have any practical use in the current project.

Realtime resource

The Realtime resource includes methods to get and update realtime API models, which are associated to a selected file. It can be used to monitor changes to a file in real-time, thus allowing external applications to reflect changes.

2.8 Authorization and Access to Google Drive

The Google Drive REST API uses mainly OAuth 2.0 for authorization of the software against Google's servers. OAuth 2.0 is the successor of Oauth 1.0 and an open standard for allowing secure delegation of data resources. The main idea of Oauth is to allow delegated access to only some part of resources, not to everything. (8) It is used for access authorization by many notable cloud service providers, such as Google, Microsoft, Amazon and Dropbox. (9)

To establish a successful connection between the software and Google Apps instance, a client ID and client secret must be generated within the Google Development console. The client ID must be granted appropriate permissions for Google Drive on each Google Apps instance, which will be hosting a shared folder. The permissions are granted from the Google Apps administration console by an administrator of the Google Apps instance. Once the client ID has been created and permissions are granted, the software is able to authenticate and gain authorization to Google's servers by using Oauth 2.0.

The actual authentication and authorization and communication between the software and Google's servers is usually done in the following steps: First the software requests an access token from Google. The access token is received upon a successful authentication. A successful authentication is achieved by using user consent. In this example, user consent is achieved with the client ID and client secret, which were generated in the Google Development Console. Assuming the authentication was successful, the software receives a session object and an access token, which is valid for a limited time. The session object is used as a proof of authentication by the software, when sending queries and commands to Google's servers. The access token is used to request a new token before the original token expires. (10)

2.9 PHP

PHP is a widely used server-side scripting language, which was officially introduced in 1995 by Mr. Rasmus Lerdorf. PHP started as a simple procedural language, but it has been further developed to an advanced object-oriented language. (11) The latest version branch of PHP is 5.6. (12)

Even though PHP is an object-oriented language, it is still not categorised as a programming language, due to the fact that PHP applications are not usually compiled to bytecode. PHP applications are typically processed directly on a server by a PHP interpreter.

2.10 MySQL

MySQL is a relational database management system (RDBMS), owned by Oracle Corporation. MySQL was initially released in 1995 by MySQL AB. MySQL is the world's second most used relational database management system and the most used open-source relational database management system. The latest stable version as of September 30th 2015, is 5.6.27. (13)

MySQL is widely used in website development, but it can also be used for other applications, such as server-side applications and even as a local database for individual applications. Since Oracle bought MySQL in 2008, a new open-source fork of MySQL was created, known as MariaDB, to address concerns about keeping MySQL free and under the GNU/GPL license. The intent of MariaDB is to maintain a high compatibility with MySQL for easier transition, in case the development of a free MySQL is seized. MariaDB's current lead developer is Michael Widenius, who was one of the founders of MySQL AB.

Because MySQL is still free and under constant development, it was chosen as the database platform for this project.

3 Requirements for Developed Software

The requirements for the software to function correctly are divided into four categories: Usability, hardware, software, and connectivity, which are further discussed in the following sections.

3.1 Usability Requirements

Based on the findings of the inquiry done in the case company at the end of 2014, most users preferred Google Chrome as the application to access the web interface of Google Drive. Some users also used mobile apps and the Google Drive synchronization client, to have a local copy of all files within a shared Google Drive folder. Based on this information, the developed software needs to work discreetly, without affecting current working habits or applications. File restoration should be easy enough for regular office workers, although restoration should be done by an experienced administrator for best results.

3.2 Hardware Requirements

The requirements for the development environment are not high, because the software is initially used only to monitor a very limited number of shared folders within Google Drive. The official requirements of the selected operating system (Debian GNU/Linux) are also so minor, that any modern PC is sufficient for development purposes. (13)

While the development environment does not need to be powerful, the production environment on the other hand needs to be powerful enough to provide a good user experience. Initial benchmarks (Appendix 1) indicate that the monitoring of one shared folder consumes an average of 3 to 6 % of CPU resources on a typical virtual machine with 1 vCPU and 4 GB's of RAM. This result was measured by recursively listing all files within a shared folder. After changing the software's logic from recursively listing items directories into a change-based scanning, the CPU consumption was reduced to a stable 3 % of CPU usage (Appendix 2).

The change-based scanning lists only files which have been changed since the last scan. With this change, the number of required files to list was reduced exponentially, thus reducing the amount of consumed resources. Based on these measurements, one relatively slow virtual machine is able to withstand the monitoring of at least 32 concurrent shared folders within Google Drive. This assumption was made on the basis that each new shared folder to monitor would cause an average of 3 percentage points increase in CPU usage. By multiplying 3 percentage points by 32, a 96 % CPU usage is achieved. The actual consumption should be considerably lower due to the fact that the server's operating system itself could consume up to 2 % of CPU while on standby, without any shared folders to monitor, due to automatic updates, indexing and connectivity checks. A more specific measurement can be achieved with a larger number of folders to monitor. The CPU consumption is directly linked to the number of API calls (API requests in Appendix 1 and API Response in Appendix 2) done via Google Drive API, as can be conducted by comparing the CPU usage to the number of API calls per second.

3.3 Software Requirements

The software requirements to run the developed software can be divided roughly in two sections: server-side requirements and client-side requirements. Server-side requirements must be met, for the software to be able to work in the server. Client-side requirements are to be met for the user interface to work for the end-user, who intends to restore file structures to an earlier state.

As for the server-side requirements, Google Drive API requires Google Developer Console authorization credentials to function, which can be obtained from Google Developers Console. The Google Drive API PHP client library requires at least PHP version 5.3 to function and the user interface requires at least Apache 2.0 to function. The targeted development and production environment is Debian GNU/Linux (wheezy), which already includes Apache 2.2 and PHP version 5.4, so these requirements are automatically fulfilled just by choosing this operating system. (14)

The client-side requirements are exactly same as using Google Drive itself, because all file controlling is made within Google Drive or with a separate file restoration interface, which originates partly from Google Drive. Google Drive requirements include any of the

following web browsers: Chrome, Firefox, Internet Explorer, Safari (only on a Mac), as long as the browser is the newest or second newest version release. (15)

3.4 Connectivity Requirements

All interaction between the software and Google Drive REST API is done via regular HTTPS requests over port 443, so there is no major requirements for the internet connection speed or firewall settings. Due to the potentially vast number of requests per shared folder instance, the connection needs to be stable and able to withstand multiple requests within a small timeframe.

Initial measurements (Appendix 1) indicated that the monitoring of one shared folder handles about 2 API requests per second and requires about 0.12 Mb/s of download and about 0.035 Mb/s of upload speed to interact seamlessly with the Google Drive REST API. After changing the software logic from a full recursive scan to a change-based scan, the number of API requests was significantly reduced to about 0.2 requests per second on average. A full recursive scan was named as full synchronization and a change-based scan was named as delta synchronization (Appendix 2).

3.5 Summary

The requirements for this project consist of three main categories: First the hardware requirements for the server, which runs the software application must be fulfilled. Second, software requirements must be fulfilled to provide the necessary platform for programming and using the file restoration interface. Third, running the software and connectivity requirements must be fulfilled to provide a consistent quality of service for the software.

Debian GNU/Linux was chosen as the operating system for the server platform. According to the official system requirements guide of Debian GNU/Linux, any modern computer would satisfy the basic hardware requirements. Software requirements are divided in two sections: server-side requirements and client-side requirements. Server-side requirements are automatically met by having the latest version of Debian GNU/Linux and client side requirements are automatically met if the end-users are able to access Google

Drive via a web browser, since the restoration interface is only plain HTML, with the exception of the instance creation interface, which uses Google Drive file picker for selecting a folder to be monitored. Because the measured network usage was very low, connectivity requirements are quite modest in terms of speed. Due to this, any modern broadband connection should suffice. More important than the connection speed would be the reliability of the connection. Long connection outages might lead into a situation where a file has been created and deleted before it has been noticed by the software. This could lead into losing accidentally deleted files permanently.

Because the software is intended to run quickly and serve multiple customers reliably 365 days a year, all the minimum requirements must be exceeded with significantly better hardware and network connections. While considering data security aspects, it is also a best practice to use the latest stable release of required software services and tools.

4 Software Structure and Functionality

This section begins by explaining the methodology used to design and develop the software. After the methodology section, the technical details are explained more thoroughly.

4.1 Practical Methodology

The project started with the knowledge of a problem with Google Drive's shared folders in organizational use. The first step was to find out of any already existing commercial products which could resolve the current problem. According to a conducted questionnaire in the case company, the only commercial product was not considered feasible. The questionnaire provided a good starting point for the design of the requirements for a new software to overcome the encountered problem. The software was designed and developed in various locations and with various devices in a timeframe of nine months. As the agile development method was chosen, the initial design did not include too strict guidelines on how the software should act in different situations. While developing the software, design plans were constantly changed to reflect the encountered problems.

As a starting point the project required a dedicated server with a PHP runtime environment, a MySQL database, a Google Apps for Work account and Google Developer credentials. A virtual server with the product name of *n1-standard-1* was bought as a Google Cloud Computing service. The *n1-standard-1* was the cheapest standard typed virtual server from Google, which was still considered to be more than enough for the software to function in a pilot phase. The required runtimes and databases were installed on the server separately. The required Google Apps for Work account and Google Developer credentials were obtained from Google before the project started.

Due to the developer's many years of experience with the PHP language, it was chosen as the language to develop the software. Although the developer had a considerable amount of knowledge in PHP, some additional information had to be studied via the official PHP website during the development. As PHP is categorized as a scripting language, it did not require any code compilation and all changes were updated in real-time. The developer did not have any previous experience with the Google Drive REST API, so information regarding this had to be gathered from the Google Drive REST API reference page.

The software was developed to run on the server as scheduled task, so it does not need any user interaction. The evaluation of this project was done by simulating real-life situations of file and folder deletions while trying to restore folder structures after a deliberate deletion. The ease and speed of file restorations were the primary metrics of the evaluation for this project. Both metrics were measured by assigning end-users of pilot companies the task of restoring files, which were deleted from a shared folder. The evaluation methods of the finished product are described in the Solution Evaluation section.

4.2 Authentication and Authorization

User authentication and authorization process was developed by utilizing Google's *Users PHP API*. This means that for a user to access the system, the user has to be in the *super administrators* group of the respective organization. The authentication process was developed to function in the following way: First the user fills a registration form to sign up for the service. After registering, the user receives a follow-up email with instructions on how to permit the software access to the user's Google Apps domain. After the user has permitted access to the software, the user is able to log-in to the software's user interface. The user interface's login page does not process any credentials, but redirects the user to a Google's login page. When the user logs in to his/hers Google Account, the user will be redirected back to the software's user interface as an authorized user. All traffic between the end-users web browser and the developed software is processed via a secure HTTPS (HTTP over SSL) connection with the TLS 1.2 protocol, which is the same technology as used in most commercial online banking systems.

4.3 User Interface

The user interface was designed to allow Google Apps for Work administrators a one-click restoration of a file structure to a given time. The interface was programmed to allow access only to registered users with the *Super Administrator* role for their Google Apps for Work domain. Once the user has registered to HSWDrive, the user must login to the system via Google's authentication system. Google's authentication system provides HSWDrive the user's administrative role, which is used to verify access privileges Authorized users are given the following options: Create a new shared folder instance,

modify the organization’s current instances and restore an existing instance to a certain point of time.

The date restoration interface only restores item locations, not the actual data. In case the content of a file needs to be restored to a certain date, Google Drive’s built-in revision control can be used. Items are also never removed, even though a restoration would be done to a time, when a certain item did not exist. Table 1 illustrates the previous statements in a more understandable way.

	Monday	Tuesday	Wednesday	Thursday	Friday
File X	*Morning: Created file in folder Y.*	*Morning: Moved to folder Z.*	*Morning: Updated file content.*	*Evening: Deleted file.*	***Morning: Restored status of Monday evening***
File A		*morning: Created file in folder B*		*updated file B*	***restored status of Monday evening***
	File X: The restored file will have the same content as it had on Thursday evening, but it will be situated in folder Y (not folder Z). File A: The file will not be removed, even though it didn't exist on Monday. The file will still exist in folder B.				

Table 1. HSWDrive file restoration logic

As designed, the restoration interface did not include anything else than a item listing for the selected date and current date, a date selector and a restoration button. The restoration button was implemented with a confirmation alert to prevent accidental restorations. Figure 3 illustrates the file restoration interface in practice.

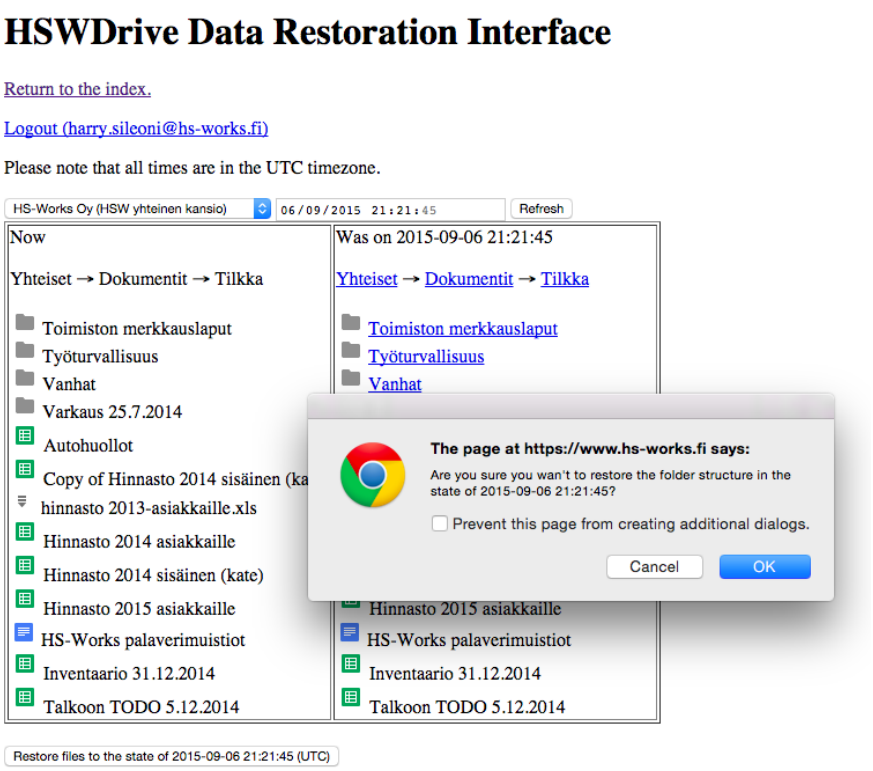


Figure 3. Screenshot from the restoration interface

As can be seen in Figure 3, the logout link includes the email address of the currently logged user. The email address is directly linked to the currently logged Google Apps user and in case the user wishes to login again, a new authentication must be done against Google's authentication system. The left-hand side column displays the current status of the selected folder, while the right-hand side column displays the status on the given date.

4.4 Functionality to Change Item Owner

The main purpose of the designed software, is to change the ownership of all files within a shared folder to one centralized user for easier file restoration. Once the files are owned by one centralized user, the files cannot be permanently deleted by other users as explained in the technology behind Google Drive and the developed software section. The centralized user is called as the data storage user. Due to the nature of Google Drive API, various steps had to be taken in order to change the owner of an item within a shared folder. Because the ownership of an item cannot be changed centrally by one administrative user, the software has to act on behalf of the previous owner to grant

ownership to a new user. Figure 4 illustrates how the owner changing process was achieved in the developed software.

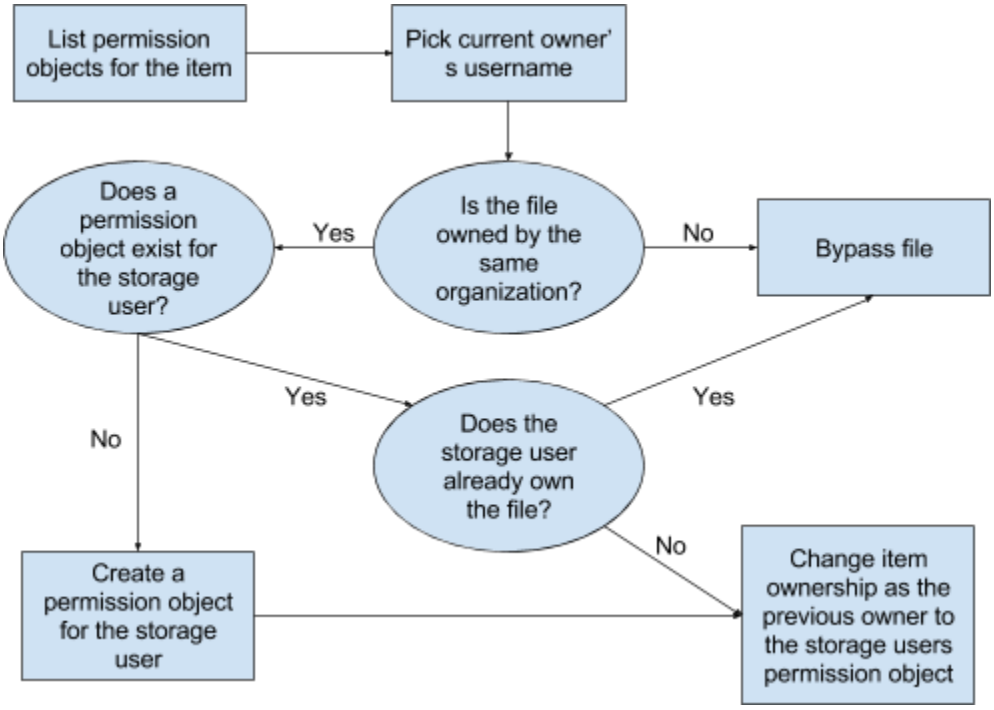


Figure 4. A flowchart to the illustrate how to change the owner of an item

Based on the flowchart in Figure 4, the first step is to list all permission objects for the given item. A permission object is an instance of the Permission resource, which was explained in the Google Drive REST API section. Once the list of permission objects has been obtained, the software looks for the permission object with the owner role. Based on the found object, the software is able to detect the username of the file's owner. If the file is owned by an external organization or the storage user is already the current owner, the file is bypassed. After obtaining the owner's username, the software checks if the storage user already has a permission object, the ID for that object is stored for future use. In case the storage user does not have a permission object, the software creates a permission object for the storage user and stores the ID for the newly created object. Once the software has the permission object for the current and future user, the software creates a new Google Drive service as the previous owner, and transfers the ownership to the storage user.

4.5 Programming Language and Framework

Most of the programming was done with the PHP language in combination with a MySQL database for file status recording. According to the original plan, a PHP based programming framework called Symfony2 was intended to be used for the project. As the software logic advanced, it seemed Symfony2 would have been too resource consuming for such software. Thus plain PHP was finally selected to be the programming language for this project. Some sections of the user interface required the use of JavaScript.

As the project advanced, it was clear that the original plan of using at least some sort of PHP framework would have been better in terms of manageability. All functions were achieved, but as a result of using plain PHP, the software became more complex to update in the future. If the software becomes a success in terms of sales, a next logical step would be to re-write most of the code in some PHP framework to support easier updates in the future.

4.6 Database Structure

MySQL was selected as the database engine. The database consists of four separate tables. The first table includes a list of all instances. An instance is basically just a row of information, consisting of the basic information of the centralized data user, company information, file ID for the shared folder and file ID's for the lost files folders. The second table consists of the status of files in the shared folder. This table includes the basic information of each file, such as the file name, instance ID, and last update time. The third table is structurally identical to the file table, but only includes the history of old files. Every time a file is being updated, the third table shows the status history of each file, as can be deducted from the table below. The fourth table consists of registered users. Only registered users are allowed to add, modify and restore instances to an earlier stage. Only registered users are allowed to add, modify and restore instances to an earlier stage.

HSWDrive database table diagram			
instances	**files**	**files_history**	**users**
id	id	id	id
folderName	**InstanceID ***	**InstanceID ***	firstName
ownerEmail	FileID	FileID	lastName
rootFolder	FileStatus	FileStatus	email
customerCompany	updatedTimestamp	updatedTimestamp	isAdmin
customerContactEmail	title	title	enabled
lastChangeId	mimeType	mimeType	
enabled	parents	parents	
lostAndFoundFolderId			
escapedFolderId			
releasedFolderId			

Table 2. HSWDrive database diagram

Each row in the instances table represents a shared folder for a single organization. The shared folder may have numerous subfolders and files. The instances table also defines the location for the files with the status of “Escaped”, “Released” or “Orphaned”, which are all moved under respectively named folders in the “Lost and found” folder for that instance.

The file tables and instance table are connected to each other with the instance ID parameter, by using the built-in relation model of MySQL InnoDB. This relation is indicated with an asterisk character in Table 2. The main idea of this database scheme is to have all the up to date file information in one table and another table is used only for logging purposes. By having a link to the instance ID, if is easy to query for files which belong to a single instance.

Each row in the users table includes basic information about the user. The isAdmin column of the user table describes whether the user has full administrative privileges for the whole system. Users with full administrative privileges are automatically granted full access to all organization’s instances. Unlike most database systems, there is no password column in the users table. The reason for this, is that authentication is done via Google’s *Users PHP API*. Login passwords are never stored or handled within HSWDrive.

Each item (file or folder) in the database has a status. Due to the fact that one item can reside in multiple locations within Google Drive, one item might be listed multiple times

in the database, if the item is in multiple shared folders that are monitored by HSWDrive. The items are separated by an instance ID, which indicates to the folder that is being monitored. Possible item statuses are: ok, deleted, orphaned, escaped, released, trashed or external. Table 3 explains thoroughly what each status indicates.

deleted	The item has been permanently deleted or moved outside of the shared folder while ownership transferred to another user and access revoked from centralized data user.
orphaned	The item has been deleted from the shared folder and as a result, it does not have any parents. This type of item is automatically added to the "orphaned" folder.
ok	The item is owner by the centralized data user and situated within the shared folder.
escaped	The item is owned by the centralized data user but is moved out from the shared folder. This type of file is automatically added to the "escaped" folder.
released	The item is accessible by the centralized data user, but ownership has been given away and the item is no longer in the shared folder. This type of item is automatically added to the "released" folder.
trashed	The item has been trashed by the data storage user and it is located in the trash of the data storage user.
external	The item exists in the shared folder, but is owned by an external organization.

Table 3. HSWDrive file status table

As stated previously, all changes to items are scanned once every minute. Each scan checks the following for all items: Is the item the root folder of the shared folder, has the item been deleted, does the item exist in the database, what are the current parents, are the parents currently within the shared folder, is the item owned by the storage user, is the file owned by the same Google Apps organization, is the file trashed, have the parents changed and has the item's name changed. To decide what to do to a file in case of a detected change, the table of Appendix 3 is used to determine the next action and file status.

4.7 Proposed Solution

The software was named as HSWDrive. It was developed to function in the following steps: First a company authorizes the software to have access to its Google Drive. Next the company decides and provides the software a shared folder and a centralized data user account, which will perform as the central data storage owner. In the third step, the software performs a recursive scan for the whole shared folder. The software lists all files and folders stored in the shared folder and adds or updates the files in a database table.

While scanning the files, the software changes the ownership of each file to match the centralized data user account. Only files owned by someone in the same organization will be modified. After these two steps are done, the software starts to monitor all changes done within the shared folder. From this point on, it will perform a full scan only once daily, but individual changes are monitored every minute. In a perfect world the monitoring alone would be enough, but due to possible errors in file ownership updates or network connectivity issues, a daily scan was scheduled to overcome possible update errors.

The end-product is a fully-functional server-side software, which runs as scheduled tasks via crontab. The software is programmed with the PHP language and it uses the Google Drive API for interaction with Google Drive. The stages of development are roughly in the following steps: First the prerequisites (software, hardware, credentials) must be met, second follows the initial development and deployment, third comes piloting and feedback and last comes the final release. The finished product is sold as a service, so the customer never gets any executable files or code.

The software consists of two separate main components to handle files: Delta synchronization and full sync. The delta synchronization component only checks for changes done to files in the given folder since the last change. The full synchronization component runs a recursive scan of all files and folders within the shared folder. A delta synchronization is run every minute, while a full synchronization is run only once daily. The reason for this division is that neither of the components alone are effective alone.

A delta synchronization is very quick to run, because it detects only the last changes. On the other hand, in case of a slight network failure in the delta sync, some files might not be processed. The full synchronization is less prone to miss files in case of a slight network error, but it might take a very long time to scan a large file structure. Another downside of a full synchronization is that it consumes a lot more network and processor resources, due to scanning each and every single file in the folder structure.

Due to the increasing number of shared folder instances, two minor components were made to run all folder instance synchronizations simultaneously. The main benefit of running a synchronization for all folders simultaneously is the noticeable increase of speed, compared to running the synchronization on all folders consecutively.

Besides the finished product, a deployment guide was made for administrators to easily setup HSWDrive on their organization. This was done to make deployment as easy as possible, considering not being able to use Google Apps Marketplace, which would have been the easiest solution. Due to time constraints, the deployment guide was considered to be more worth the effort than submitting a beta staged software publicly to Google Apps Marketplace.

5 Solution Evaluation

The following section describes the designed and implemented methods for evaluating and testing the developed software solution. The evaluation was done in terms of usability, resource consumption and speed.

5.1 Planned Solution Evaluation Methods

The product quality was evaluated in terms of functionality, ease of deployment, ease of use and speed of item restoration in case of an accidental file or file structure deletion. The functionality evaluation was considered as successful, if the software is able to keep track of all files and file changes within a shared folder. Besides being able to keep track of files and changes, the software must also be able to function quickly to provide a fluent user experience. A delay of 5 minutes was chosen to be the maximum accepted time for a change to be recorded within normal use, although the expected average delay is less than one minute. These could be tested by making changes to items in a shared folder and confirming that the changes are registered within the software's database. Ease of deployment was measured by providing the customer administrators a deployment guide and observing if the customer is able to deploy the software to their organization with minimal help. Ease of use and speed of item restoration was measured by asking regular users to deliberately delete files and attempt restoration via the HSWDrive file restoration interface. This type of evaluation was considered to be the only way to concretely see if the software is reliable, fast and easy enough for production use with future customers.

5.2 Used Methods for Testing

Besides evaluating the basic requirements of the solution, practical testing was done in the following categories: resource consumption, speed of use and ease of use with a total of 11 shared folders from various organizations. The solution was stress tested for resource usage by deploying various shared folders from various organizations for pilot use. After the deployment, resource usage was monitored via the server's own reporting tools and Google Developer Console. The detailed measurements can be found in the Results and Analysis section.

Speed was monitored by adding items in a shared folder and calculating the time that the software requires to process the item. The file processing speed was observed afterwards, by using Google Drive's activity pane for the selected items. This method provided a reliable way to measure speed, without having to compare update time between Google Drive and the software's database.

Ease of use was tested in two sections: Implementation and usage. The implementation section was tested by providing the administrators of customer organizations a deployment guide and testing if the customers were able to deploy the software by themselves. The usage section was tested by giving the pilot customers a tasks to recover files by using the restoration interface. If the customer was able to implement the software with minimal help, the requirement for ease of deployment was considered as a success. If a user was able to restore files to any earlier location with minimal help, the requirement for ease of use was considered a success. A separate quick guide (Appendix 4) was given to the end-users to help in the deployment and recovery process.

6 Encountered Problems during Development of Software

As most software development projects, also this project had problems which were not anticipated in the design phase. This section describes the encountered problems and how they were resolved.

6.1 High Resource Usage

While testing the functionality of the first working version of the software, a high amount of system resource usage was measured particularly in CPU, network and disk usage (Appendix 1). While this did not cause a major problem in the speed or reliability of the software with one case company, it would have had a greater impact in an environment of multiple companies and shared folders. Because the software is aimed to be sold as a service, it is expected to handle various instances within one physical server without having noticeable latency or reliability issues.

After a logic survey of the software the cause of high resource usage was pinpointed to the logic of recursive file scanning. As the initial version of the software used to scan through all files within a shared folder every hour, it caused a major resource usage peak every hour. Due to the large number of files within the case companies shared folder, this scan took almost an hour for each scan. To overcome this problem, a change in program logic was implemented in the following way: Instead of scanning through all the files every hour, only the changes made to files within a shared folder were monitored after the initial recursive scan. This was possible by using the Changes resource, provided by the Google Drive REST API. With this logic, all changes could be detected each minute, and still the average resource usage would was measured to be only 10 % compared to the original resource usage (Appendix 2).

6.2 PHP Client Library Mismatch with Documentation

Because the Google Drive API PHP Client library is updated very frequently, the documentation provided by Google did not always comply with the actual functions of the

client library. This caused some significant problems in making the software work correctly. These problems were exceeded by investigating the source code of the client library. Requests for correlation of the documentation were sent to Google.

6.3 Code Re-writing Due to Not Using a Software Development Framework

The final decision of choosing not to use Symfony2 framework, unlike originally planned, caused a significant increase in programming time, because many functions had to be re-written in the event of a change to the database structure. When this deficiency was noticed, the software was already developed so far, that moving it to a framework was not considered to be worth the effort. The software was finished without a framework, but if time and budget allows, it will be re-written within Symfony2 framework in the future.

6.4 Complexity of Google Drive API

While starting the project, it was not obvious that even seemingly simple operations, such as changing the owner of a file required multiple phases to achieve. Even though the software can be granted full administrative privileges to an organization's Google Drive, each operation is done subjectively with a user account. This means that instead of changing the owner of a file needs to be done as the current owner. As an example: It was not possible to just define a new owner to a file with one simple command. To change the owner of a file, the first operation was to list all permission objects for the file. After obtaining a list of permission objects, each object had to be scanned for the username and permission level. While scanning the permission objects, the current owner had to be selected to confirm if the file was already owned by the storage user. In case it was not, the second phase was to check if the storage user had any permission object at all. In case the storage user did not have any permission object, one needed to be created. Once the current owner was found and the storage user had a permission object, the ownership could finally be transferred from the owner's permission object to the storage user's permission object. The transfer had to be done as the previous owner.

Another complexity aspect of Google Drive API was that all items are considered as files. Understanding that within Google Drive, folders are actually just empty files with children objects was a new revelation. For simplicity, "item" was selected as the word to represent

any file or folder in this thesis. Due to the fact that one item can co-exist in multiple folders, it was required to understand the concept of parent objects and children objects. Parent objects can be thought as the locations for the item, while child objects can be thought as files within a folder. Naturally only folder objects can have children objects.

6.5 Files or Folders Not Visible to All Users

After having the software online for a few days, some pilot companies reported on lost files, which could only be found by some users in the respective organization. These reports led to an investigation, which indicated that although the lost items were in the corresponding folder, they would not be visible to some users. The users who experienced these problems had sufficient privileges to the respective folder, but still some files would appear have vanished. The underlying reason for this has been unknown up to this date but a work-around has been implemented to overcome this problem. The work-around to overcome the issue was to re-share the folder to the users who are experiencing problems. After the folder was re-shared, no further complaints were received by the users.

7 Licensing Terms and Distribution Methods

This section covers basic information about possible licensing terms and distribution methods for the developed software solution. Even though licensing terms and distribution methods were not the focus of this project, they are relevant or future reference.

7.1 Current State of Licensing and Distribution Methods

As mentioned in the previous chapters, the software is made to be sold as a service, so a separate license agreement is not made between the buyer and the seller. On the other hand, the customer is required to accept a service contract, which describes the content and price of the subscription. The pricing and contract terms are not the focus of this thesis, so they will not be discussed any further.

The initial target group consists of companies which have bought Google Apps for Work from the case company. Due to this target group, most agreements are initially done via telephone or email, while the distribution is done remotely as a service. After the software has been distributed to the initial target group, it will be targeted to other companies in the future.

7.2 Commercial Use

Although the commercial use of the developed software is not the main focus of this thesis, it is still a focus point of the future. Once the software has been successfully tested for at least 6 months on various test companies, it can be considered to be a ready for commercial use. To truly make it commercial, it has to be published in Google Apps Marketplace for easier distribution.

The following procedure must be done in order to publish the app: First a payment system needs to be integrated to the registration form. This can be done with PayPal or another similar payment system. After that, it is required to register as a developer to Google Apps Marketplace. Registering requires a 5 USD payment to Google. After registering, the second phase is to take screenshots of the software and create a manifest file. The manifest file should include basic information about the software, such as the

name, description and icons. After a manifest file has been created and screenshots are taken, the third phase is to submit the manifest file, screenshots and application URLs to Google Developer Dashboard. After submitting the app, the fourth phase is to send a request for listing to Google Apps Marketplace by filling the Google Apps Marketplace Listing Review Request form. Once the form has been submitted, Google reviews the software and decides whether or not it is acceptable for publishing. If the software is considered as acceptable, the fifth phase is to test the installation of the software via Google Apps Marketplace. If everything works as intended, the final phase is to start marketing the software via various internet-based channels. (14)

7.3 Source Code Licensing

Although the software itself is intended to be sold as a service for monetary gain, the source code may be published with the GNU General Public License (GNU GPL) or similar license, thus granting other developers (individuals and organizations) the freedom to use, modify and copy the source code. There are numerous reasons why this is considered as a good practice. First, all the developers programming knowledge has been gained from free resources, including most of the API resources and code snippets. Based on this, it would only be fair to give something back. Second, publishing the source code would allow other developers to provide feedback and recommendations to improve the code. Third, publishing the source code could also function as public demonstration of excellence, thus potentially provide work opportunities in the future. Finally, as the software is intended to be run on a server, if is very unlikely that small business organizations would have the resources or knowledge for deploying the software, while larger business organizations usually tend to avoid open source solutions without a guarantee of support.

If GNU GPL is chosen as the license model for the software, each source code file should include a preamble, indicating that the source code is under the GNU GPL license. For the time being, no decisions have been made regarding the licensing of the software. This decision will be made in the future when the software has been further developed and implemented to a larger scale of customers.

8 Discussion and Conclusions

This section discusses the results and outcome of the finished software solution. The amount of work and resource consumption is the main focus, while the summary section includes the final measurements.

8.1 Results and Analysis

The development of the software was done within 9 months, which was within the set timeframe and did not cause any delays to other work matters. The software had a total of 4194 lines of PHP code and 865 lines of CSS code for the user interface. The original assumption of required PHP code was less than 1000 lines, so the amount of implementation work was exceeded multiple times. The main reason for the vast amount of code was clearly because of not using a PHP framework, which would have minimized the required amount of manually written code. If the original plan of using Symfony2 Framework had been followed, most of the manually made PHP methods could have been automized and database structure changes could have been updated from one central location, instead of updating all functions manually in the case of a change in the database structure.

The software was measured to function with very minimal resource consumption. By increasing the number of shared folders to be monitored from one to eleven, the only noticeable resource increase was observed in the CPU usage. As indicated in Figure 5, the CPU usage with one folder was measured to be approximately 3 %, increasing the number of folders to eleven increased the CPU usage to approximately 17 %. This indicates that the server should withstand at least 32 concurrent folders, while still having a 50 % CPU consumption average on the current virtual machine. Taking in account that the used virtual machine is the slowest standard typed virtual machine within the Google Cloud Computing Platform, it is just a matter of upgrading to a faster virtual machine, in case more processing power is required.

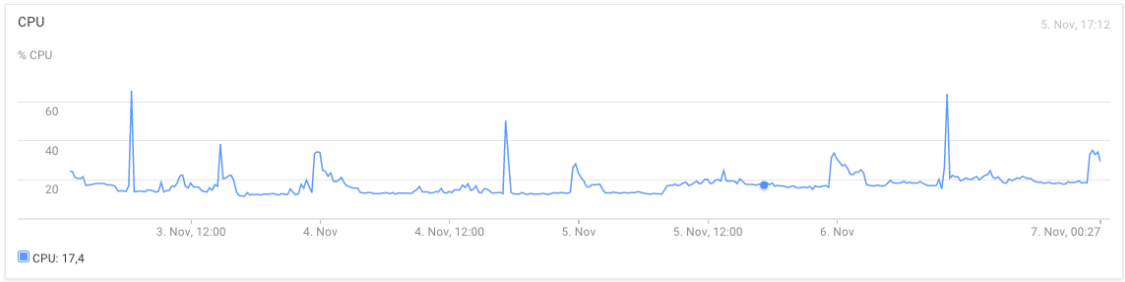


Figure 5. CPU consumption with 11 folders to monitor

By observing Figure 6, it is clear that even with 11 shared folders to monitor, network and disk usage activity had almost zero impact to the server with less than 50 Kbps network usage and 30 KBps disk usage on an average. Some traffic spikes were observed, but even those had very minimal constraint to the server.

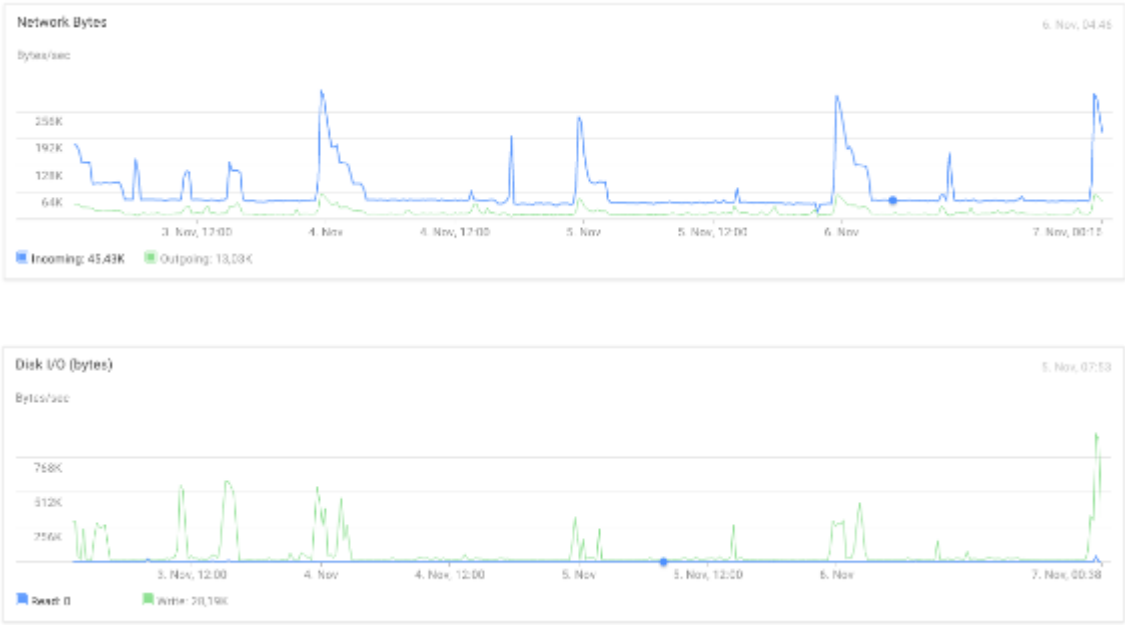


Figure 6. Network and disk usage consumption with 11 folders to monitor

The only significant increase of resource consumption was measured as the number of API requests done to the Google Drive API, as can be observed in Figure 7. While the first version of the software consumed an average of 2 requests per second for one shared folder, the improved version with a change-based scan only consumed an average of 0.2 requests per second for one shared folder. The measurement of 11 shared folders consumed an average of 7 requests per second, which increases that the average of one folder would be 0.64 requests per second. This sudden increase of requests can be explained with the fact that most of the latest customers had just migrated to Google Drive, thus adding new files on a daily basis. Once the new customers have

finished their migration, the average number of requests should be considerably lower. When considering that the maximum number of daily API requests is one billion (1 000 000 000), there is still capacity for 1000 shared folders, even with the measured peak value of over almost one million daily requests. The success rate of requests was measured as 100 %, which indicates in a flawless network connection and in the adequacy of quota for API requests.

Figure 7. API Request measurements with 11 shared folders to monitor

As the requirements indicated that the software had to be easy to use, various tests were performed by giving customers the task to deploy an instance and restore deleted items. According to the tests, all 5 customer administrators were able to restore deleted files successfully. The deployment phase was considered as more difficult to most customers, due to the advanced setting changes within Google Apps Administrator Console. Only 2 users out of 5 users were able to deploy the software by themselves. Taking in consideration that most of the customers were not technically experienced, this was still considered as a success in terms of deployment easiness. Once the software is deployed in Google Apps Marketplace, the deployment will be considerably easier for users with a less technical background.

While building the solution, many new aspects of Google Drive were learned, including many which are not obvious to the regular user. Before the project was started, it was quite unclear how Google Drive handles file deletions in shared folders. Even with the knowledge of Google Drive's file handling logic, it might still be challenging to explain the operations logic to regular users in a clear way, due to its owner-based file system. In the beginning of the project, Google Drive seemed like any other file system, where files can be listed, copied and modified by simple commands. As the development advanced, it was clear that the sophistication and design of Google Drive was done in a very different way, since each operation to a file required multiple phases to accomplish the desired outcome. Most of the functions were combined to PHP methods, which made the operations easier in the future, but the amount of work required to build these functions was far greater than was anticipated while starting the project. One of the new discoveries was the fact that a file which is not deleted, but inaccessible to a user is considered to be deleted in the Google Drive's point of view.

8.2 Outcomes

A software solution was designed and developed according to the initial requirements. The software was named as HSWDrive and evaluated successfully with various pilot customers. The software runs a full synchronization to all HSWDrive instances on a daily basis and a delta synchronization every minute. File structure restorations were tested and proven to work as designed.

Even though the software has been proven to work as expected with a set of customers, further analysis and bug tracking is required before publishing it in Google Apps Marketplace for wide distribution. The software is already in active use by the case company and many of its customers. Further development for new features, such as file revision restoration and support for externally owned files is scheduled for the near future.

8.3 Summary

The aim of this project was to design and develop a software solution to allow easy restoration of accidentally deleted files in a Google Drive shared folder. The software

was designed, finished, tested and deployed to various pilot customers within 9 months, which was considered to be within an acceptable timeframe. The development would have been faster with a PHP framework but on the other hand, it probably would have had consumed more system resources. If time allows, the software will be re-written in a PHP framework in the future.

The software was a success in terms of functionality and demand, as it was proven to function fluently on various pilot companies. Ease of use was also measured to be satisfactory, as all pilot users were able to restore files and 40 % were able to deploy the software independently. Some technical issues were discovered during initial deployment, but once these were overcome, no significant deficiencies were observed. The software is still in a testing phase, but after a more comprehensive and successful testing period of at least 6 months, the software can be submitted to Google Apps Marketplace for easier deployment and sales purposes.

A future release of the software will include the possibility to restore files not only to earlier locations, but also earlier states. Due to Google Drive's automatic revision history, this can be achieved without having to store file data in the software's own database. Another functionality to be developed in the future is the possibility to restore files owned by another organization. Currently this is not allowed, because HSWDrive only handles files owned by the same organization and file ownerships cannot be transferred between organizations due to Google Drive's limitations. Even though it is not possible to take ownership of such files, it is still possible to monitor and log the files in case of an accidental deletion.

References

1. Google Ltd. Atmosphere Live: Keynote. 2014.

2. Google. Find an orphaned file - Google Apps Help. 2015; Available at: https://support.google.com/a/answer/6008339?hl=en. Accessed March 14, 2015.

3. LeBlanc D. Linux for Dummies. 7th ed. Indianapolis, Indiana: Wiley Publishing, Inc; 2006.

4. Google Ltd. Data Centers – Google. 2013; Available at: http://www.google.com/about/datacenters/inside/data-security/, 2015.

5. Google Ltd. View and manage file versions - Drive Help. 2015; Available at: https://support.google.com/drive/answer/2409045?hl=en, 2015.

6. Google Ltd. Recover a deleted file - Drive Help. 2015; Available at: https://support.google.com/drive/answer/2405957?hl=en, 2015.

7. Google. Google Drive REST API Overview - Drive REST API. 2015; Available at: https://developers.google.com/drive/web/about-sdk. Accessed March 14, 2015.

8. Boyd R. Getting Started with OAuth 2.0. : O'Reilly Media; 2012.

9. Pocatilu P, Boja C, Ciurea C. Syncing Mobile Applications with Cloud Storage Services. Informatica Economica 2013;17(2):96-108.

10. Google Ltd. Using OAuth 2.0 for Web Server Applications. 2015; Available at: https://developers.google.com/identity/protocols/OAuth2WebServer, 2015.

11. Lerdorf, Rasmus. PHP on Hormones. 2007 April 26.

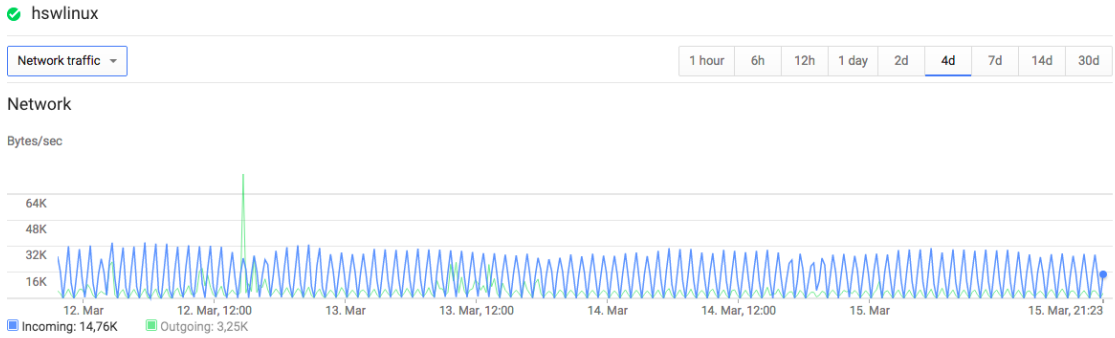
12. The PHP Group. PHP: Releases. 2015; Available at: http://php.net/releases/. Accessed March 13, 2015.

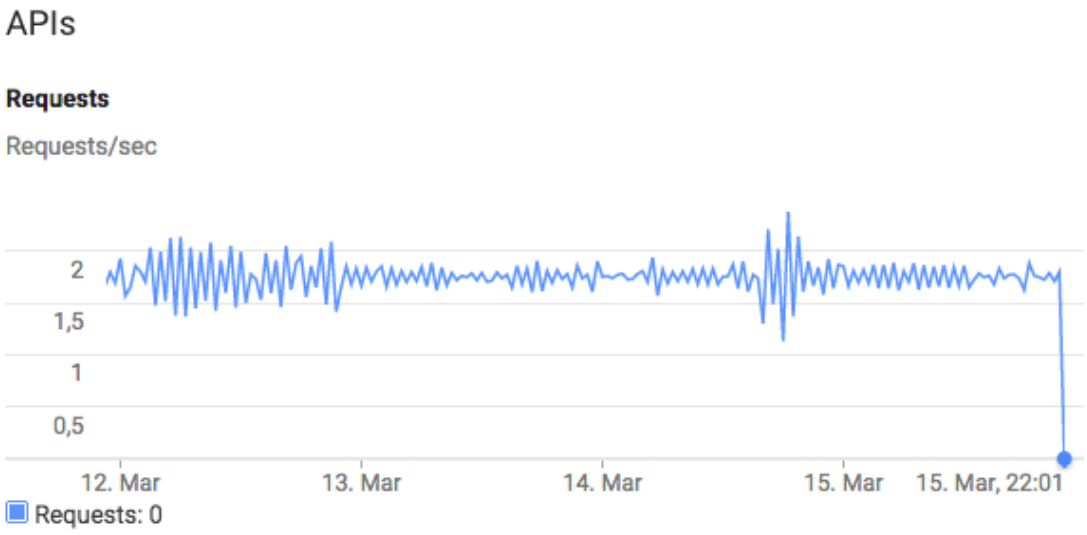
13. Oracle Corporation. MySQL | The Most Popular Open-Source Database | Oracle. 2015; Available at: http://www.oracle.com/us/products/mysql/overview/index.html, 2015.

14. Google. Publishing Your App | Apps Marketplace | Google Developers. 2015; Available at: https://developers.google.com/apps-marketplace/listing, 2015.

Initial Google Drive API measurements

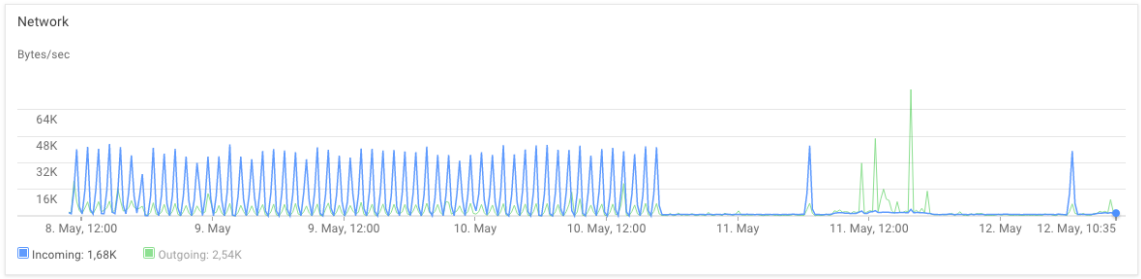
Initial Google Drive API measurements of 1 shared folder to monitor for the last 4 days. Measurements taken on March 15th, 2015 at 9:25 pm by Harry Sileoni.

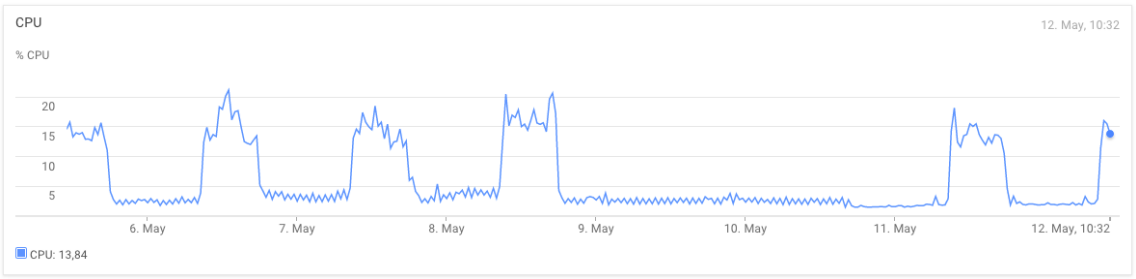




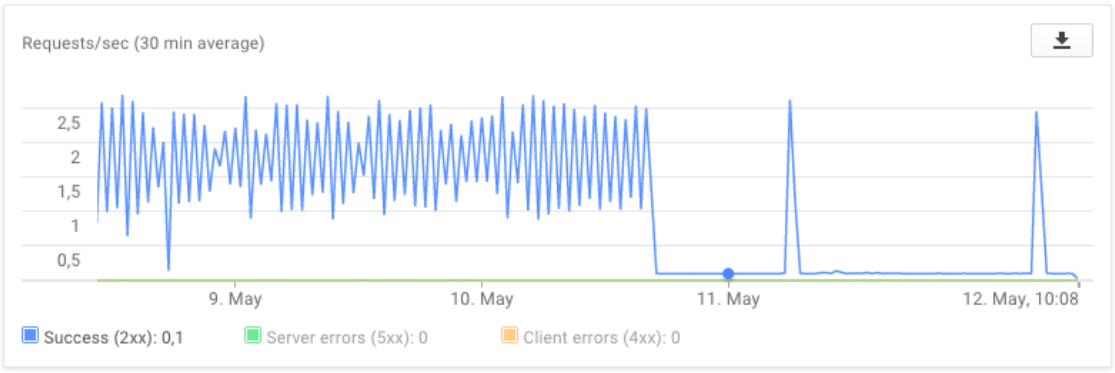
Google Drive API measurements after changes optimization

Google Drive API measurements for the last 7 days with 1 shared folder to monitor. The scanning logic was changed on May 10th at 5:20pm. Measurements taken on May 12th, 2015 at 10:30 am by Harry Sileoni. A clear reduction in resource usage can be observed.





Response codes



HSWDrive logic table for file changes

Is root folder	Deleted	Existing in DB	Parents	Parents in root	Owned by storage user	Owned by same company	Trashed	Parents unchanged	File name changed	Task	New DB-status	Old DB-status
-	true	true	-	-	-	-	-	-	-	update DB	deleted	-
-	false	true	false	-	true	-	false	-	-	update DB, add to orphaned-folder	orphaned	!= 'orphaned'
-	false	true	true	true	true	-	false	-	-	update DB	ok	!= 'ok'
-	false	true	true	false	true	-	false	false	-	update DB, add to escaped-folder (keep existing parents)	escaped	!= 'escaped'
-	false	true	false	-	false	-	false	false	-	update DB, add to released folder	released	!= 'released'
-	false	true	true	false	false	true	false	-	-	take ownership & update DB	ok	!= 'ok'
-	false	true	true	false	false	-	false	false	-	update DB, add to released folder	released	!= 'released'
-	false	true	-	-	true	-	true	-	-	update DB	trashed	!= 'trashed'
-	false	true	true	false	true	-	false	true	-	update DB	escaped	!= 'escaped'
-	false	true	true	-	false	-	false	true	-	update DB	released	!= 'released'
-	false	true	true	false	false	-	false	false	-	update DB	released	!= 'released'
-	true	false	-	-	-	-	-	-	-	do nothing	-	-
-	false	false	false	-	true	-	false	-	-	do nothing	-	-
-	false	false	true	true	true	-	false	-	-	add to DB	ok	-
-	false	false	true	false	true	-	false	-	-	do nothing	-	-
-	false	false	false	-	false	-	false	-	-	do nothing	-	-
-	false	true	true	true	-	true	false	-	-	take ownership & add to DB	ok	-
-	false	true	true	false	-	-	false	-	-	do nothing	-	-
-	false	true	true	true	true	-	false	-	-	add to DB	trashed	-
-	false	false	true	true	false	false	false	-	-	add to DB	external	-
-	false	true	true	true	false	false	false	-	-	do nothing	-	-
-	false	true	-	-	-	-	-	-	yes	update name in DB	-	-
-	false	true	-	-	-	-	false	false	-	update parents in DB	-	-
true	false	false	-	-	false	true	false	-	-	add to DB	ok	-

Same color = same action + DB status

Border over status = update files_history

Quick guide for users (in Finnish)

HSWDrive -ohje

Kiitos kun kokeilet HSWDrive -sovellusta. Kyseessä on palvelusovellus, joka pitää huolta siitä, että jaetun kansion tiedostoja on helppo palauttaa, mikäli niitä poistetaan. Yrityksesi tulee päättää käyttäjätili, jonka omistukseen kaikki tiedostot jatkossa menevät (**jatkossa "tiedostovarastokäyttäjä"**). On erittäin suositeltavaa, ettei kyseisestä käyttäjätunnusta käytetä tavallisesti kirjautumalla lainkaan. Delegoitu käyttö on hyväksyttävää. Asentamista varten tarvitset vähintään yhden Google Apps -käyttäjätilin, jolla on **Super Administrator** (**jatkossa nimellä "järjestelmänvalvoja"**) -tason oikeudet Google Apps -tiliinne.

Asentaminen

1. Rekisteröidy HSWDrive-palveluun osoitteesta https://www.hs-works.fi/hswdrive/ ja valitsemalla **Register as a new user**. Syötä järjestelmänvalvojatason käyttäjän tiedot.

2. Saat nyt englanninkieliset ohjeet jatkamiseen sähköpostitse. Tässä vastine suomeksi:

    a. Kirjaudu järjestelmänvalvojatason käyttäjällä osoitteeseen http://admin.google.com
    b. Mene Tietoturva → Näytä lisää → Lisäasetukset → Hallinnoi OAuth-verkkotunnusavainta.
    c. Varmista että **Ota tämä kuluttaja-avain käyttöön** on valittuna ja tallenna muutokset.
    d. Palaa etusivulla ja mene Tietoturva → Näytä lisää → Lisäasetukset → Hallinnoi sovellusliittymäasiakkaan käyttöoikeuksia.
    e. Kopioi sekä liitä seuraavat tiedot oikeisiin kenttiin ja klikkaa lopuksi **valtuuta**:

        i. Asiakkaan nimi:
        559904892732-3cdptb19e19jk86se8iact01jkdp3ues.apps.googleusercontent.com

        ii. Sovellusliittymän ala:
        https://mail.google.com/,https://www.googleapis.com/auth/admin.directory.user,https://www.googleapis.com/auth/admin.directory.user.readonly,https://www.googleapis.com/auth/drive,https://www.googleapis.com/auth/gmail.compose,https://www.googleapis.com/auth/gmail.insert,https://www.googleapis.com/auth/gmail.labels,https://www.googleapis.com/auth/gmail.modify,https://www.googleapis.com/auth/gmail.readonly

3. Varmista että haluamasi jaettu kansio näkyy sekä järjestelmänvalvojakäyttäjälle että tiedostovarastokäyttäjälle. Kullakin käyttäjällä tulee olla muokkausoikeudet.

4. Luo uusi seurantainstanssi menemällä osoitteeseen https://www.hs-works.fi/hswdrive/ ja valitsemalla **Create a HSWDrive instance to monitor your folder**. Kirjaudu sisään järjestelmänvalvojatason käyttäjällä jonka rekisteröit kohdassa 1, klikkaamalla **Connect Me!**.

Hyväksy lupapyynnöt. Syötä aukeavaan lomakkeeseen seuraavat tiedot:

a. **Shared folder description** -kohtaan kansion lyhyt nimi tai seloste.
b. **Shared folder** -kohtaan valitaan jaettu kansio, klikkaamalla **Select folder**.
c. **Domains** -kohtaan syötetään kaikki organisaation domainit pilkulla eroteltuna (esim. yritys.fi,yritys.com,yritys.net. Jos domaineja on vain yksi, syötetään pelkkä domain ilman pilkkuja.
d. **Data owning users email** -kohtaan syötetään tiedostovarastokäyttäjän ensisijainen sähköpostiosoite.
e. **Company name** -kohtaan syötetään yrityksen nimi.
f. **Company contact email** -kohtaan syötetään yrityksen sähköpostiosoite yhteydenpitoa varten.

Syötä lomake klikkaamalla **Submit**. Jos kaikki meni oikein, pitäisi jaetun kansion tiedostojen siirtyä tiedostovarastokäyttäjän omistukseen automaattisesti. Huomaa että ensimmäisessä käyttöönottosiirrossa saattaa kestää tiedostojen määrästä riippuen monta tuntia tai jopa vuorokausia.

Seurantainstanssin muokkaaminen

Mikäli seurantainstanssia halutaan muokata tai ottaa pois käytöstä, toimitaan seuraavasti:

1. Mene osoitteeseen https://www.hs-works.fi/hswdrive/ ja klikkaa **Modify your current HSWDrive instance**.
2. Kirjaudu sisään järjestelmänvalvojatason käyttäjällä jonka rekisteröit asennusohjeen kohdassa 1, klikkaamalla **Connect Me!**.
3. Valitse instanssisi luettelosta ja klikkaa **Refresh**.
4. Päivitä haluamasi tiedot ja klikkaa **Submit**. Huomaa ettei kansiota pysty enää muuttamaan jälkikäteen. Jos haluat vaihtaa kansiota, joudut luomaan sille uuden instanssin.

Poistettujen tiedostojen palauttaminen

Jos käyttäjä poistaa tiedoston tai tiedostoja vahingossa, tulee palautus tehdä järjestelmänvalvojakäyttäjän toimesta seuraavasti (palauttaa kaikki kansion tiedostot halutun päivän kansiorakenteisiin - tiedostojen sisältöön ei tule muutoksia):

1. Mene osoitteeseen https://www.hs-works.fi/hswdrive/ ja klikkaa **Restore your instance to an earlier state**.
2. Kirjaudu sisään järjestelmänvalvojatason käyttäjällä jonka rekisteröit asennusohjeen kohdassa 1, klikkaamalla **Connect Me!**.
3. Valitse haluamasi instanssi sekä palautusaika (UTC-aikavyöhykkeessä) ja klikkaa **Refresh**. Tämä tuo esiin. tiedostolistauksen, josta näet kyseisen kansion tiedostorakenteen haluttuna päivämääränä. Kun löydät päivämäärän, jolloin tiedostot olivat oikeassa paikassa, voit palauttaa ne alareunan **Restore files to...** -napilla. Palautuspäivän jälkeen luotuja tiedostoja ei poisteta. Saat sähköpostiin kuittauksen kun tiedostot on palautettu.