Designing a Virtual Security Layer for Cloud Content

Lukas Klingsbo

September 29, 2015

Abstract

TODO: Abstract
Keywords:

Contents

1 Introduction

Developing larger projects containing static content usually involves using a Content Distribution Network to be able to scale to a large user base. The commercial Content Distribution Networks are usually fairly easy to use, the content that is to be used in a project is usually simply uploaded and then directly available to the public. For secret content this can be a problem and that is what this thesis is about. This work examines ways of enforcing access control on content and groups of content in the form of views. A system was developed to make the underlying theory work in practice.

2 Related Terminology

2.1 Technologies

2.1.1 React

React is a JavaScript library for building user interfaces. React uses both its own virtual DOM and the browser's, this makes it able to efficiently update dynamic web pages after a change of state through comparing the old virtual DOM with the resulting virtual DOM after the state change and then only update the browser's DOM according to the difference between the virtual DOMs [1].

2.1.2 Flux

2.1.3 Scala

Scala is a multi-paradigm programming language. It most commonly runs on the JVM and compared to Java it supports most functional programming features at the same time as it supports object oriented programming [2].

2.1.4 REST

Over http? WebSockets?

2.1.5 TODO: Insert persistent storage here

TODO: Write down related terminology, if any

2.2 Abbreviations

2.2.1 CDN

Content Distribution Network - Replicates content to several servers, usually spread out geographically. Once a request is made, the network serves content from the server closest to the requester.

3 Related Work

3.1 Copy-on-Write

This work relies heavily on the Copy-on-Write principle which was founded and used in the Mach kernel [3]. Today Copy-on-Write is used in everything from file systems [4] to desktop compositors [5]. Its principle is that when processes share

data in between each other it is copied first when one of the processes does changes to it. This is an optimisation as the processes does not have to send all of the related data that is in memory, rather they only have to send pointers to the data. After many Copy-on-Write's a complex structure can be built up, but it is possible to solve that structure [6].

4 Background

4.1 About Uprise

Uprise is a company based in Uppsala, Sweden.

4.2 The current system

Maybe write about battlebinary?

4.3 Problem description

Having

Battle Binary

What is needed * Security layers * Views * Virtual file structure * Versioning of content * Multi project support * Auth and audit logs * Users

5 Model

5.1 Entities

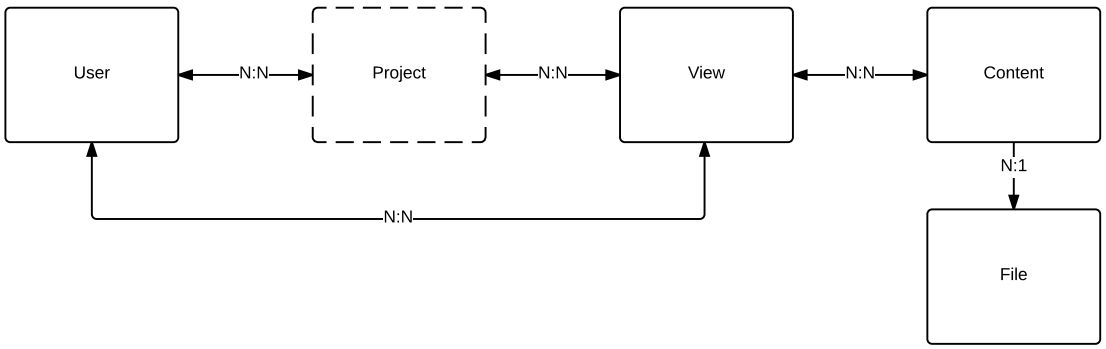

Figure 1: Entity Relationships

5.1.1 View

A view is a group of files which can be changed independently of the view's parent. A view can have subviews which are either automatically dependant on its parent and updated when the parent is updated or optionally updated or forked from its parent once their content diverge.

5.1.2 Project

A project is just a name for the view which is a root of a tree of views. The system can contain several projects and their trees are completely disjoint.

5.1.3 File

A file refers to an actual physical file in the file system.

5.1.4 Content

Content is an asset contained in a view, it is a virtual file. It could be described as a link between the view and the real file. It can be for example an image, video or binary blob combined with meta-data.

5.1.5 User

A user is the structure that handles people who has been granted access to the system. Access to the system is handled by a separate service, like LDAP.

6 Copy-on-Write

7 Methods for determining implementation details

This chapter introduces the different methods used to determine how the new system should be implemented, which DBMS it should use and how the estimation of long term scaling was done.

8 Security of the system

8.1 Authorization

8.2 Audit logs

9 Resulting system

9.1 REST Endpoints

For the frontend to communicate with the backend REST is used, the following endpoints were configured:

- projects

  GET - list all projects

  POST - create new project

- projects/{id}

  GET - get specific project

  PUT - update existing project

  DELETE - delete existing project

- projects/{id}/content

  GET - list all content in a specific project

  POST - create new content in a specific project

- projects/{id}/content/{id}

  GET - get specific content in a specific project

  PUT - update existing content in a specific project

  DELETE - delete existing content in a specific project

- projects/{id}/views

  GET - list all views in a specific project

  POST - create new view in a specific project

- projects/{id}/views/{id}

  GET - get specific view in a specific project

  PUT - update existing view

  DELETE - delete existing view

- projects/{id}/views/{id}/content

  GET - list all content in a specific view

  POST - create new content in a specific view

- projects/{id}/views/{id}/content/{id}

  GET - get specific content in a specific view

  PUT - update existing content in a specific view

  DELETE - delete existing content in a specific view

9.2 Scalability

10 Discussion

11 Summary

11.1 Conclusions

11.2 Future work

Stuff to write about: Modular design, every piece should be interchangable LDAP - why it was used as standard AUTH

References

[1] A. React, "Javascript library for building user interfaces," 2014.

[2] M. Odersky, P. Altherr, V. Cremet, B. Emir, S. Micheloud, N. Mihaylov, M. Schinz, E. Stenman, and M. Zenger, "The scala language specification," 2004.

[3] M. Accetta, R. Baron, W. Bolosky, D. Golub, R. Rashid, A. Tevanian, and M. Young, "Mach: A new kernel foundation for unix development," 1986.

[4] O. Rodeh, J. Bacik, and C. Mason, "Btrfs: The linux b-tree filesystem," *ACM Transactions on Storage (TOS)*, vol. 9, no. 3, p. 9, 2013.

[5] P. Sabella and N. Wilt, "Desktop compositor using copy-on-write semantics," June 28 2005. US Patent 6,911,984.

[6] F. J. T. Fábrega, F. Javier, and J. D. Guttman, "Copy on write," 1995.