SCHOOL OF COMPUTING

UNIVERSITY OF TEESSIDE

PRAGUE COLLEGE

Computing Project 2019

Patrick Ethier

December 1st, 2019

Supervisor: Bohus Ziskal

Second Reader:Petr Svarny

World count: 8107

Computing Project 2019

Patrick Ethier

December 1, 2019

Abstract

This document presents the waterfall development lifecycle which produces a web application that enables the management and editing of ETL configuration files sourced from Github. Covered are the steps of requirements gathering, topic research, design, implementation and testing. The final evaluation demonstrates acceptance testing and reflects on lessons learned and further development.

Place: Prague College
Student ID: 18357644
Module Code: COM3051-N
Project Supervisor: Bohus Ziskal
Semester Code: 1904
Word Count: 8107

Contents

1 Introduction

Log management is crucial to managing Information Security (IS). According to (Kent and Souppaya 2006), "Log generation and storage can be complicated by several factors, including a high number of log sources; inconsistent log content, formats, and timestamps among sources; and increasingly large volumes of log data." This project will focus on helping to address the *"formats"* and *"inconsistent"* segments of these factors. As such, the customer has developed a solution to aid with this problem by attempting to provide a universal, Java-based technology to help with the Extraction, Transformation and Loading (ETL) of log entries that is compatible with multiple log management and big data technologies. This technology is available via an Open Source code repository situated on their GitHub at https://github.com/secureops/fieldextraction-rules.

The main problem with the customer's product is that the definition of each ETL *parser* is done by interacting with a complicated hierarchy of text files that require multiple configuration file formats and implies that the end-user has a considerable understanding of the file structure. It also assumes that the end-user has a certain amount of proficiency with GitHub's git interface in order to be able to contribute new *parsers* to the product's GitHub repository.

The *primary objective* to be achieved by this project is to produce a user environment that will reduce the dependency on manually manipulating these text files and provide a mechanism to abstract the use of the GitHub tools.

As will be discussed in the various sections of this report, the proposed solution includes the use of web-based technologies to provide a graphical user interface to address these goals. The project was executed using the *waterfall* approach with further details available in appendix A. A detailed log of progress entries (blog requirement) is provided via the commit logs of the various repositories associated with this project listed in appendix B. A test plan was provided to users and a copy is available in appendix C and a deployment guide was also provided and is available in appendix E.

2 Requirements

This project is driven by the customer requirements delineated in table 1. Further, the customer imposes certain technological constraints, listed in table 2, due to the fact that this project is part of a greater set of ambitions to offer multiple tools and products from a common web-based portal. It is important to point out that the customer's portal was not in production and was still evolving over the course of this project. Subsequently, the output of this project will likely serve as one of the first components to be hosted by this new portal.

Table 1: Customer Requirements

ReqID	Description	Requestor	Priority	Measure
R1	Provide a graphical user interface to author Morphline HCON files	Customer	Primary Feature	No operations outside of web browser
R1.1	User should be able to create new patterns	Customer	Primary Feature	File is created on Github
R1.2	User should be able to edit existing patterns	Customer	Primary Feature	File is changed on Github
R1.3	User should be able to test patterns using sample logs	Customer	Primary Feature	Function executes and returns parsed logs
R1.4	User should be able to mark existing patterns as deprecated	Customer	Secondary Feature	New parser definition ignores deactivated parsers
R2	Interface should be presented in a workbench style presenting all the parsers available	Customer	Primary Feature	customer acceptance testing
R3	Architecture must manage the lifecycle each user's contribution	Customer	Primary Feature	customer acceptance testing

Table 2: Customer Constraints

ContraintID	Description
C1	Application must be hosted by SecureOps' AWS account using existing Cognito, CloudFront and APIGWW technologies
C1.1	Any user signed up via Cognito should be able to use the application
C1.2	Any user related data collected beyond what is currently stored in Cognito must not be persisted inside the application infrastructure permanently
C2	Use current definitions distributed via github.io/secureops/fieldextraction-rules as a starting point for each user
C3	Changes must be submitted as commits to Github

The intention was to follow the "SMART" principles as described in (Mannion and Keepence 1995) defined as follows:

> in specifying software requirements we define SMART to be:
>
> **S** pecific
>
> **M** easurable
>
> **A** ttainable
>
> **R** ealisable
>
> **T** raceable.

Unfortunately, the requirements provided by the customer were mostly vague in terms of measurement and resulted in the customer setting a boundary of using only a web browser to replace their current workflow. Hence, this goes against the *Specific* nature of SMART requirements but the customer did not want to elaborate further than describing their workflow and specifying they wanted a web browser/workbench environment as a replacement tool. This makes it very difficult to apply the aforementioned SMART principles. Nonetheless, the testing provided in table 6 attempts to trace and measure each requirement in order to provide some indication of project success independent of customer acceptance testing.

3 Research and Methodology

As brought forward by the project specification available in appendix A, the successful delivery of the final feature-set is strongly dependent on expanding the understanding of a few key technologies explored in the following sections. Research was conducted in providing a graphical user interface to facilitate the management of parser definition files resulting in the presentation layer. This will be coupled by researching the business logic and persistence layers, provided by GitHub, which will result in enabling collaboration and version control. Lastly, research for an infrastructure stack, provided by Amazon AWS is required to provide an ecosystem upon which to implement the result.

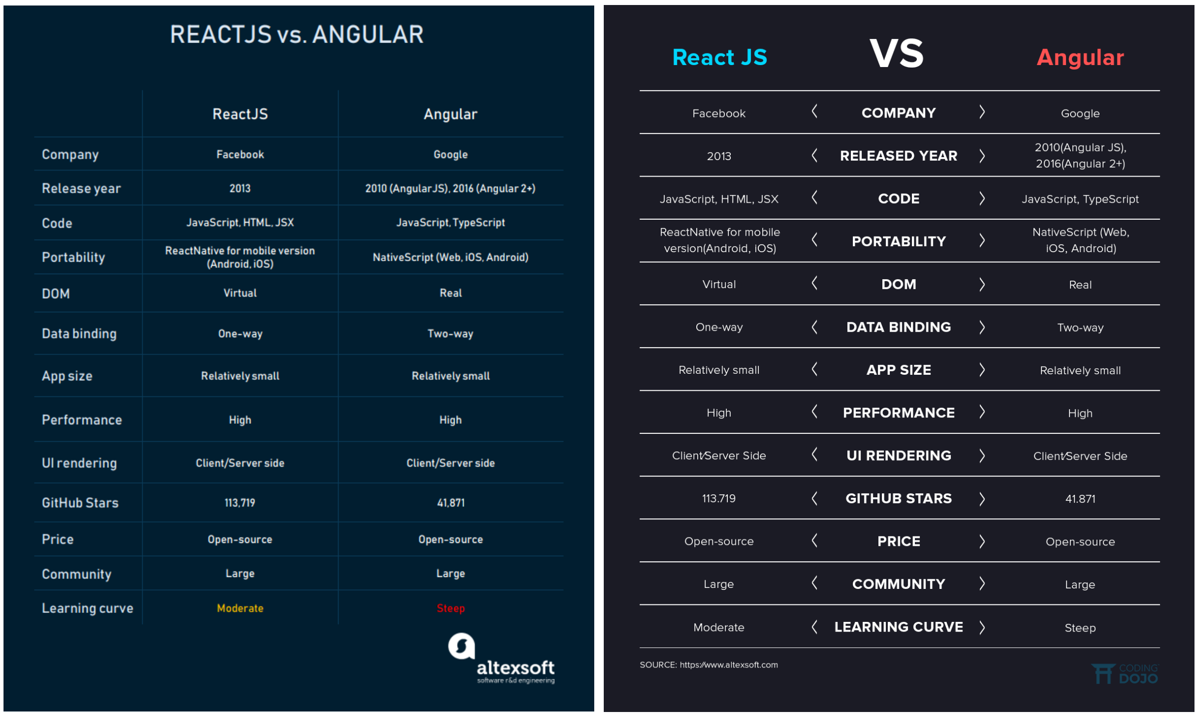
The research below is conducted by using the methodologies of Strength, Weaknesses, Opportunities, Threat (SWOT) analysis, prototyping/discovery and gap analysis in order to specifically identify challenges or options that will need to be resolved to further design and implementation.

3.1 UI Technology review (SWOT Analysis)

Constraints imposed by the Amazon AWS serverless infrastructure and detailed in section 3.3 imply that the web application cannot use a completely traditional server-side programming model as described in (MDN Contributors 2019b) where "sites display static templates (built using HTML, CSS, and JavaScript), and then dynamically update the data displayed inside those templates when needed". In the case imposed by the combined usage of Amazon Cloudfront and Amazon API Gateway, the site's content will need to be statically served from Amazon S3. Hence, there is no web server that can modify the contents of the web page upon being requested and it is up to the JavaScript engine inside the web browser to obtain the dynamic data and render the web page as described in (MDN Contributors 2019a). It also means that using a traditional Multi Page Application (MPA) approach would also push the responsibility to keep the state between pages onto the web browser.

The alternative is to build the entire website into a Single Page Application (SPA) where the JavaScript is used to dynamically modify a single static page served from the serverless infrastructure. As described in the blog posting at (Macquin 2018), SPAs come with some drawbacks which are not of consideration for this project such as "perform poor on the search engine".

The deciding factor to proceed with a SPA is resumed in the "Drawbacks of Multi Page Applications" section of (Macquin 2018). They state that a "Coupled backend and frontend" is a drawback to MPA. This describes our biggest constraint since we cannot "couple" the rendering of content on the backend to alleviate the front-end. This justifies forgoing a MPA approach in favour of a SPA methodology.



REACTJS vs. ANGULAR

	ReactJS	Angular
Company	Facebook	Google
Release year	2013	2010 (AngularJS), 2016 (Angular 2+)
Code	JavaScript, HTML, JSX	JavaScript, TypeScript
Portability	ReactNative for mobile version (Android, iOS)	NativeScript (Web, iOS, Android)
DOM	Virtual	Real
Data binding	One-way	Two-way
App size	Relatively small	Relatively small
Performance	High	High
UI rendering	Client/Server side	Client/Server side
GitHub Stars	113,719	41,871
Price	Open-source	Open-source
Community	Large	Large
Learning curve	Moderate	Steep

altexsoft
software r&d engineering

React JS	VS	Angular
Facebook	COMPANY	Google
2013	RELEASED YEAR	2010(Angular JS), 2016(Angular 2+)
JavaScript, HTML, JSX	CODE	JavaScript, TypeScript
ReactNative for mobile version(Android, iOS)	PORTABILITY	NativeScript (Web, iOS, Android)
Virtual	DOM	Real
One-way	DATA BINDING	Two-way
Relatively small	APP SIZE	Relatively small
High	PERFORMANCE	High
Client/Server Side	UI RENDERING	Client/Server Side
113.719	GITHUB STARS	41.871
Open-source	PRICE	Open-source
Large	COMMUNITY	Large
Moderate	LEARNING CURVE	Steep

SOURCE: https://www.altexsoft.com

CODING DOJO

(a) (AltexSoft 2018) (b) (Balog 2019)

Figure 1: SWOT, ReactJS vs. AngularJS

A survey of SPA related technologies reveals many frameworks. Multiple internet searches were done to identify potential SPA technologies and many blog posts reflect the conclusions provided by (Patel 2019) that, "Angular, React, Vue.js and Node.js are today the most used JavaScript frameworks in the market." The related surveys and most research on the Internet argue many points such as "Angular and React have many similarities and many differences. One of them is that Angular is a full-fledged MVC framework and React is merely a JavaScript Library (just the view)", cited from (Hamedani 2018). Most of these posts consist of various subjective opinions on the efficiency, difficulty and proliferation of each solution/approach based on facts that are so similar that there is no clear consensus as to which tool is best to learn and proceed with. For example, the SWOT analysis listed in figure 1 isolates the two most popular SPA technologies as developed by Facebook and Google and provide no clear winner for the purposes of this project.

Given the fact that there are no clear winners, the following considerations for this project were used to choose a technology:

- Familiarity and availability of reference material
- Due to a personal limitation of limited web-page design experience, the technology should provide a good ecosystem of pre-built visual components
- Due to limited knowledge of build and compilation environments, the technology should provide a flexible, pre-built packaging and debugging environment that doesn't require major modifications.

ReactJS was chosen based on the above for the following reasons. This list is summarised in tables 3 & 4 below:

- The customer of this project already has portions of their portal using the ReactJS framework that can be used as a starting point
- Angular strongly recommends learning TypeScript, which adds further complexity above and beyond all the other new concepts that would need to be learned in order to execute this project.
- Neither of the technologies, including cursory research done on other frameworks such as VueJS and Svelte, show any weaknesses or benefits that would, at first glance, prevent or enhance the chance of project completion.
- *Getting Started* and other tutorials for each framework all allude to pre-packaged build environments and seem fairly comprehensive in building a scalable web application.
- ReactJS seems to have a fair amount of pre-existing web components as per https://npm.org.
- Most importantly, ReactJS seems to be the platform adopted by Amazon AWS as the default in most of the reference material which is not pure JavaScript, such as their Amazon Amplify library that deals with most of their mobile and web-app build tools.

Table 3: Angular SWOT

Strength	**Weakness**
	Not currently used in customer's portal environment
	Recommends learning TypeScript
No apparent shortcomings that would affect project outcome	
Has a good *Getting Started* and *pre-packaged* coding environment	
Seems to have a good visual component library on Node Package Manager	
	Not used natively by Amazon Amplify

Table 4: ReactJS SWOT

Strength	**Weakness**
Currently used in customer environment	
Uses JSX which seems like a simple HTML extension	
No apparent shortcomings that would affect project outcome	
Has a good *Getting Started* and *pre-packaged* coding environment	
Seems to have a good visual component library on Node Package Manager	
Used natively by Amazon Amplify	

3.2 Github API (Prototyping)

Constraint C3 and Requirement R3 imply that the customer's GitHub repository needs to be used as a basis for each user's environment. This entails the use of the GitHub API inside the application. GitHub provides a pure HTTP API located at https://developer.github.com/v3/git/. GitHub also provides various JavaScript libraries that deliver an abstraction layer to this API along with specific interfaces to the chosen language environment. A SWOT analysis is not necessary because the number of libraries compatible with ReactJS is limited to Octokit located at https://octokit.github.io/rest.js/. It provides either REST or GraphQL interface meant to be used within a JavaScript engine such as a web browser. The documentation for Octokit provides most of the major functionality that will be required based on the use-cases in table 1 which includes:

- Authenticate to Github
- Fork the repository
- Read a file or files
- Create a new file
- Write a file or files with versions
- Delete a file or files

Therefore, a small prototype was developed to explore the API for the functionality below and was later merged and used inside the SPA.

3.2.1 Authentication

Further inspection of the API at (Github Inc. 2019, ch. #authenticate) reveals that there are multiple choices for authentication such as:

> The string can be one of
>
> 1. Personal access tokens
>   You can create personal access tokens in your account's developer settings or use the "Create a new authorization" endpoint.

2. OAuth access tokens
   Retrieved using the web application flow. See also: octokit/oauth-login-url.js.
3. GitHub App bearer tokens
   See octokit/app.js to retrieve a JSON Web Token (JWT) in order to authenticate as a GitHub App.
4. GitHub App installation tokens
   See octokit/app.js to retrieve an installation access token in order to authenticate as a GitHub App installation.

Due to constraint C1.2, which requires not to store any long term credentials for the user, the choice of using username and password is not recommended because it would add complexity in storing the credentials locally and break this requirement. Obtaining a personal token, bearer token and installation token would complicate the user experience as the user would need to navigate the GitHub advanced configuration interface to obtain these values and potentially cause a technical support issue for the more novice users. This leaves the OAuth2 option, which, in the Amazon AWS serverless environment will require further consideration as discussed in section 4.1.

3.2.2 File manipulation

Manipulating files with Git is a little unconventional due to how it handles versioning. As described in (Chacon and Straub 2019-11-01, ch. 5), git stores the contents of each *version* of a file as an object using its SHA-1 hash as a filename. All the metadata about the object is stored in a *tree* object which brings together all the files and subtrees together. When a *commit* is executed, the file is added to the object store and a new *tree* is built updating the reference to the object. Each *commit* object points to a particular tree. In order to keep track of a specific version, git uses the concept of *ref* to represent a specific tree or a specific object at a particular commit. The *latest ref*, called *head* usually points to the last commit. These relationships are represented in figure 2

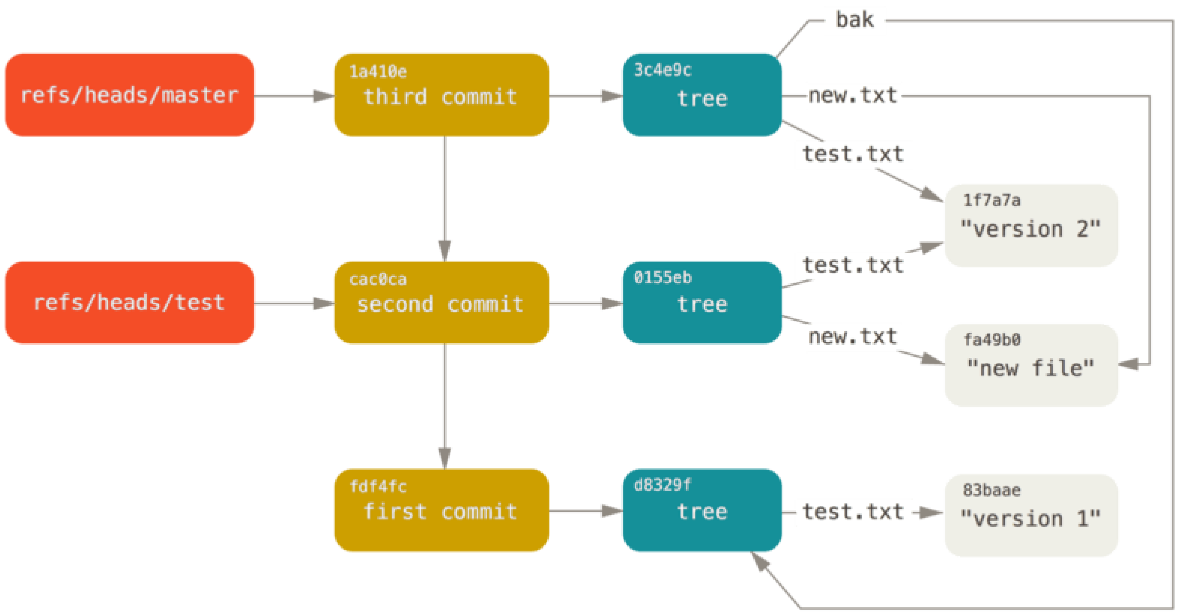


Figure 2: Git directory objects with branch head references included (**pp.449**; Chacon and Straub 2019-11-01)

The Octokit API provides the *createOrUpdateFile* function which handles managing these *trees* and *refs*. Since the entirety of the repository is not dowloaded into the web browser, the design will need to manage these references and trees across versions. As well, this function offers no option to combine multiple files at once. In this case, there are two options. The first involves manually rebuilding the tree with all the file changes after uploading the new objects in order to create a single commit for all the files, the second is to adopt a single file per commit approach. There is a preference for the first approach because it will be easier to *revert* a change within the UI in cases, like when a new Morphlines parser is created, where an atomic action in the final application results in changing multiple files.

3.2.3 File contents

While researching the Octokit API during the task #9 of the project timeline, it was noticed that simply parsing the contents of a file into a manipulatable structure brought up an issue. An easy way to illustrate the problem is demonstrated in listing 1 with the results in listing 2.

Listing 1: Text obfuscation code

```
const originaltext = `
{
  "bkey": "hello",
  "akey": "world"
}`;
console.log("Original File Contents\n---");
console.log(originaltext);
console.log("---");
const jsonobject = JSON.parse(originaltext);
console.log("New File Contents\n---");
console.log(JSON.stringify(jsonobject));
```

Listing 2: Text obfuscation output

```
Original File Contents
---
{
  "bkey": "hello",
  "akey": "world"
}
---
New File Contents
---
{"bkey":"hello","akey":"world"}
```

As can be observed, the *JSON.parse()* and *JSON.stringify()* do not preserve whitespaces and newlines. This was further observed while parsing *.properties* files using the library provided by https://github.com/npm/ini, where parsing the *.properties* files obtained from GitHub would result in a *commit* even though the user made no changes. This must be addressed to prevent users from *committing* files they didn't modify but were parsed in order to drive a visual component.

3.3 AWS integration (Gap Analysis)

In order to fulfill constraint C1, a gap analysis of the Amazon AWS serverless infrastructure is needed to determine what dependencies and what deficiencies exist with the customer's current web application and what is necessary to complete this project. Table 5 summarises the results of this gap analysis.

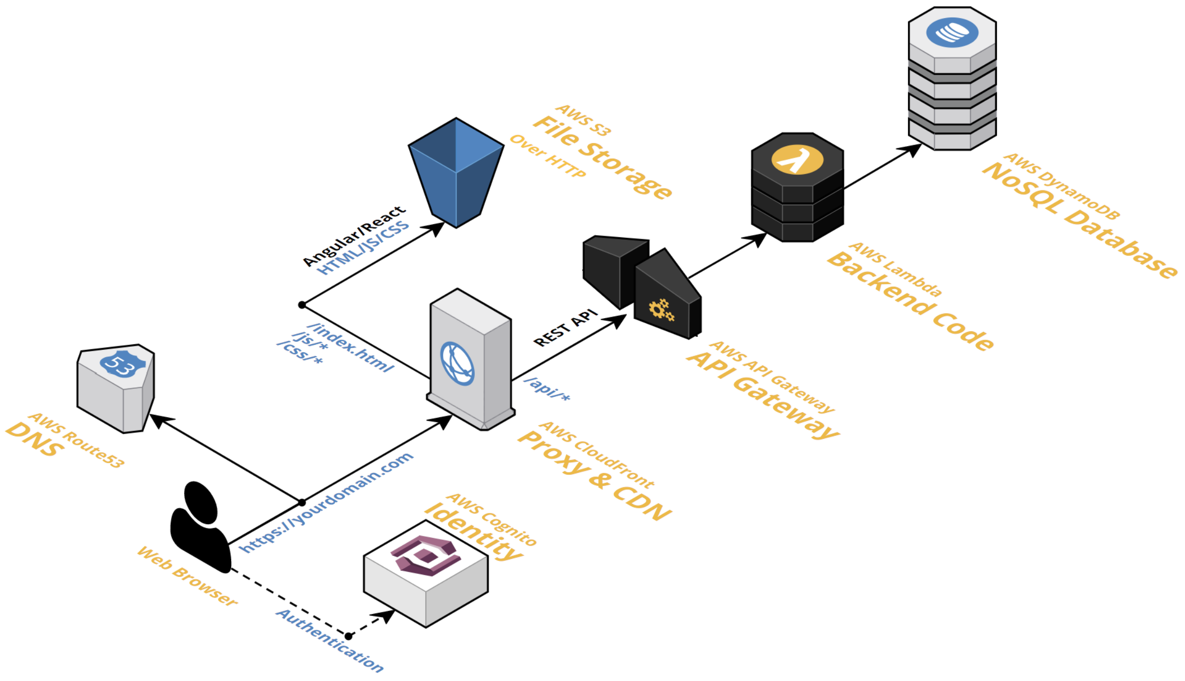


Figure 3: AWS Serverless Infrastructure Overview (Fernando 2018)

The architecture illustrated in figure 3 displays the components currently used by the customer. The Amazon Cloudfront component offers the capability to intercept the incoming arrow from the web browser *(viewer-request and viewer-response)* as well as the arrows pointing to Amazon API Gateway and Amazon S3 *(origin-request and origin-response)* via a mechanism called Amazon Lambda@Edge which will execute Python or NodeJS based functions to manipulate the various request/responses. A previous project used Amazon Lambda@Edge to implement OAuth2 but it was using the *implicit* authentication flow of the standard. Adding GitHub requires to use OAuth2 as well but requires the *authorisation* flow of the standard. In light of this finding, the customer alluded to the fact that they would like to convert their current *implicit* flow to *authorization* flow as well. An explanation of the OAuth2 standard can be found at https://tools.ietf.org/html/rfc8252. Further details will be covered in section 4.1.

The use of the SPA approach also implies that the entirety of the customer portal be done within a single page structure served by the Amazon S3 component in figure 3. Ideally, independent functionalities of the portal would be isolated into their own Amazon S3 bucket and configured as their own *origin*. As explained in (Amazon AWS 2019), each *app* can reside at their own path (/app1, /app2, ..., /appn) and would be mapped to their own "Behavior" which retrieves the referenced object from the origin designated by the behaviour. At first glance, this looks obvious and straightforward, but an important issue arises when invoking each app's *path* directly without explicitly specifying the *index.html* object in the requesting url. To clarify, fetching *https://app.com/app1* will not return the file index.html inside the Amazon S3 bucket because the Amazon Cloudfront "Behavior" doesn't support a *default object* to return in the case where no object is specified. There is an exception to this for the the root / object where a default object such as index.html can be specified. A mechanism will need to be developed in order to allow Amazon Cloudfront to detect when a URL is requesting a *path* and then return *index.html*. This will allow Amazon Cloudfront to present the entire web portal as a MPA while building itself from multiple SPA sub-deployments.

Another observation is that the deployment of the various serverless components on Amazon AWS via the web portal requires the assimilation of many different products and concepts that would make deployment and installation a complex affair to explain. The administrator's manual would literally need to reference the user back to multiple dozens of screenshots and walk-throughs in order to proceed to operations that only need to be done once and that are generally of no concern to the administrator. Amazon AWS offers some automation tools to simplify deployments using a tool called Amazon CloudFormation. During the research phase of this project, it quickly becomes apparent that using this tool greatly increases the efficiency of administration as well as the development process. Deployment/redeployment can be easily achieved from running a simple script versus having to click through multiple wizard-styled web pages.

Current	Desired Target
Single, Implicit OAuth2 flows using Amazon Cognito	Multiple OAuth2 authentication flows based on Amazon Cognito and GitHub
Single application/origin	Multiple application/origin
Manual deployment using web UI	Scripted deployments using Amazon CloudFormation

Table 5: Amazon AWS gap analysis results

4 Design

The design activities discussed in this section are essentially the results of prototyping and gap-analysis done in section 3. Once the overall technology landscape was stable, finalising the design was discussed with the customer in order to achieve a Minimum Viable Product (MVP) for most components. For both the *backend* system and the static *web UI* application, the proposed components will probably need to be reviewed in further iterations since the customer may wish to change their overall approach to delivering their portal. Hence, this app is modular and can be deployed independently in a staging environment using the provided Amazon CloudFormation scripts discussed in section 5.

4.1 Backend Design

The design of this project involves placing together many components that are deployed within the confines of the Amazon AWS environment illustrated in figure 3. Figure 4 provides a UML package diagram of the components associated with this project as well as their GitHub repositories. Links to these repositories are available in appendix B.

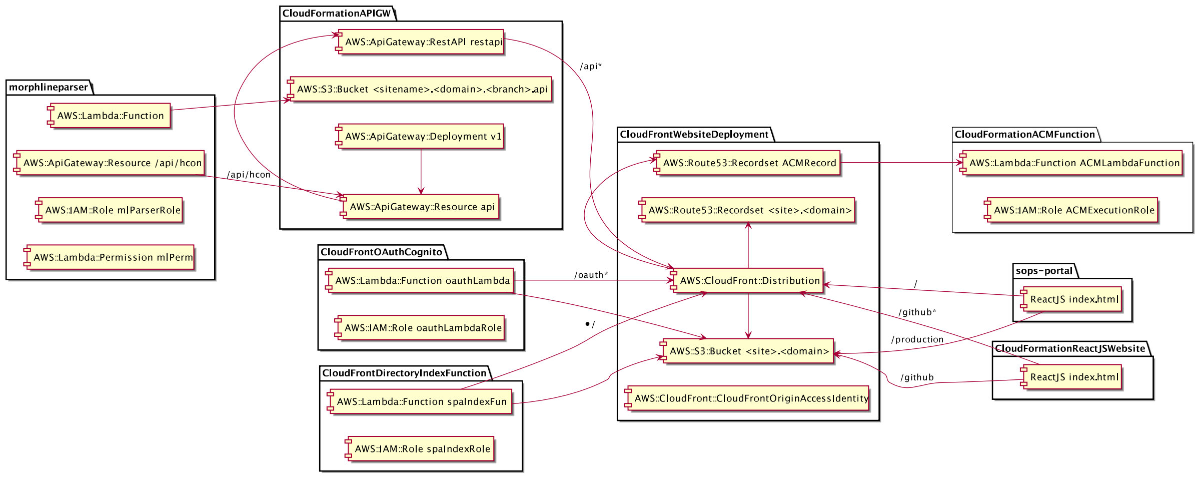
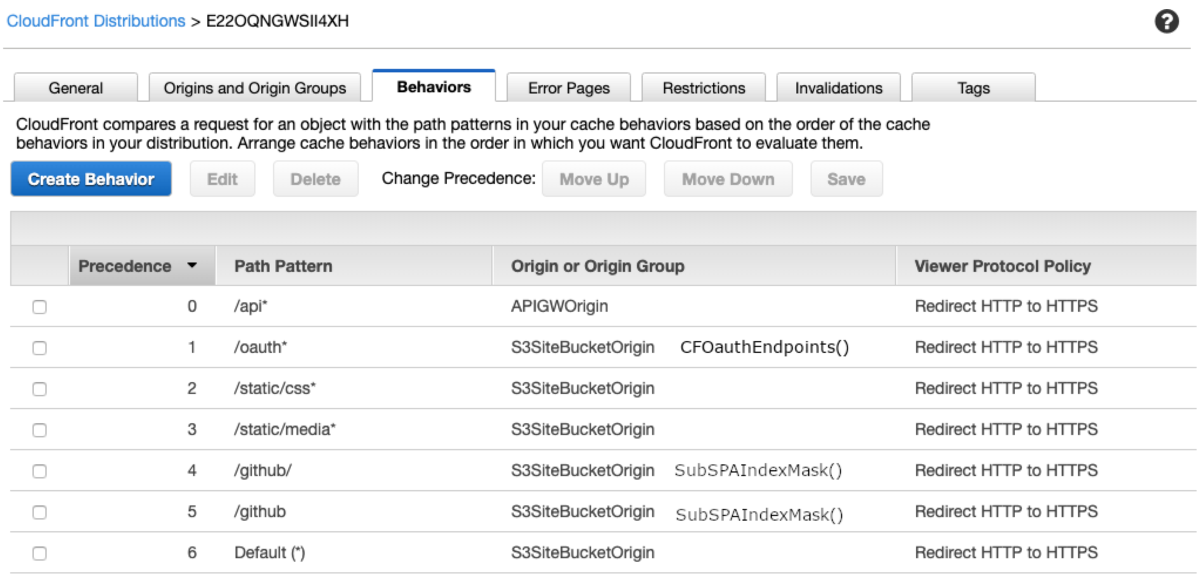

CloudFormationAPIGW
AWS::ApiGateway::RestAPI restapi
AWS::S3::Bucket <sitename>.<domain>.<branch>.api
AWS::ApiGateway::Deployment v1
AWS::ApiGateway::Resource api
morphlineparser
AWS::Lambda::Function
AWS::ApiGateway::Resource /api/hcon
AWS::IAM::Role mlParserRole
AWS::Lambda::Permission mlPerm
/api/hcon
/api*
CloudFrontWebsiteDeployment
AWS::Route53::Recordset ACMRecord
AWS::Route53::Recordset <site>.<domain>
AWS::CloudFront::Distribution
AWS::S3::Bucket <site>.<domain>
AWS::CloudFront::CloudFrontOriginAccessIdentity
CloudFormationACMFunction
AWS::Lambda::Function ACMLambdaFunction
AWS::IAM::Role ACMExecutionRole
CloudFrontOAuthCognito
AWS::Lambda::Function oauthLambda
AWS::IAM::Role oauthLambdaRole
/oauth*
*/
CloudFrontDirectoryIndexFunction
AWS::Lambda::Function spaIndexFun
AWS::IAM::Role spaIndexRole
sops-portal
ReactJS index.html
/
/github*
/production
/github
CloudFormationReactJSWebsite
ReactJS index.html

Figure 4: Project package Overview

The entire project is laid out according to the capabilities offered by Amazon Cloudfront. Each path on the web application serves either an API function, hosted via Amazon API Gateway and Amazon Lambda, a static web page, generated by ReactJS or a pseudo-path that handles requests and responses with a function defined within Amazon Lambda@Edge. These are listed in figure 5.



CloudFront Distributions > E22OQNGWSII4XH

CloudFront compares a request for an object with the path patterns in your cache behaviors based on the order of the cache behaviors in your distribution. Arrange cache behaviors in the order in which you want CloudFront to evaluate them.

Precedence	Path Pattern	Origin or Origin Group		Viewer Protocol Policy
0	/api*	APIGWOrigin		Redirect HTTP to HTTPS
1	/oauth*	S3SiteBucketOrigin	CFOauthEndpoints()	Redirect HTTP to HTTPS
2	/static/css*	S3SiteBucketOrigin		Redirect HTTP to HTTPS
3	/static/media*	S3SiteBucketOrigin		Redirect HTTP to HTTPS
4	/github/	S3SiteBucketOrigin	SubSPAIndexMask()	Redirect HTTP to HTTPS
5	/github	S3SiteBucketOrigin	SubSPAIndexMask()	Redirect HTTP to HTTPS
6	Default (*)	S3SiteBucketOrigin		Redirect HTTP to HTTPS

Figure 5: Amazon Cloudfront Origin Behaviours

Notice the call to step 2 in figure 6 corresponds to the pattern defined in behaviour #4 in figure 5. This elaborates the solution of using a Amazon Lambda@Edge function to compensate for Amazon Cloudfront's limitation of not recognising when a request consists of a path and not an actual object discussed in section 3.3. Steps 13 and 19 in figure 6 demonstrates the process required to obtain a GitHub OAuth2 token discussed in section 3.2 in order to use the API and maps to origin behaviour #1 in figure 5.

All calls to behaviour #0 in figure 5 will be redirected to their corresponding paths in Amazon API Gateway, calls to #2 and #3 will map to the customer's original website and calls the #4 and #5, when specific objects are specified, will map to the contents generated by our ReactJS static pages in the *CloudFormationReactJSWebsite* repository as shown in figure 4.

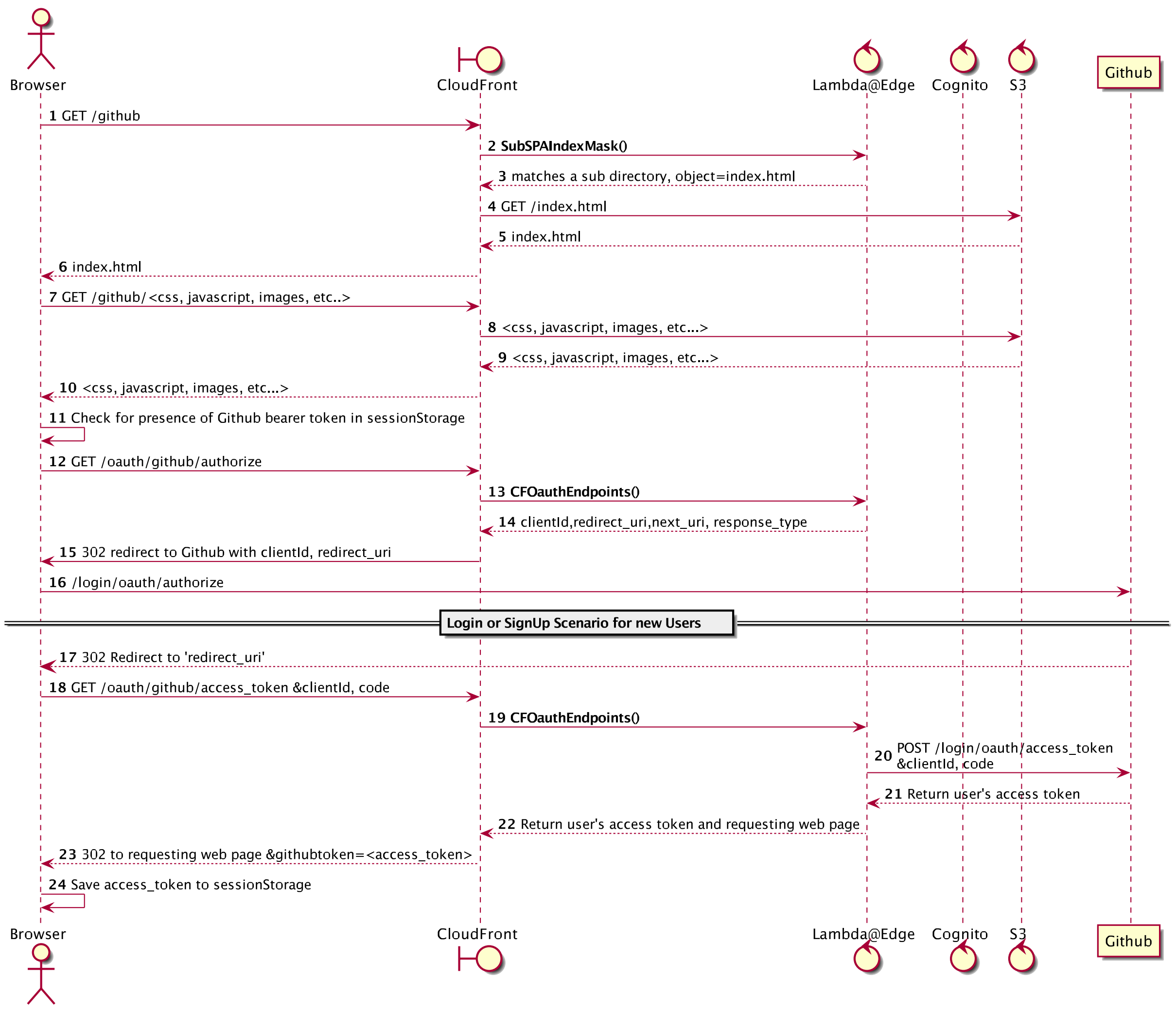
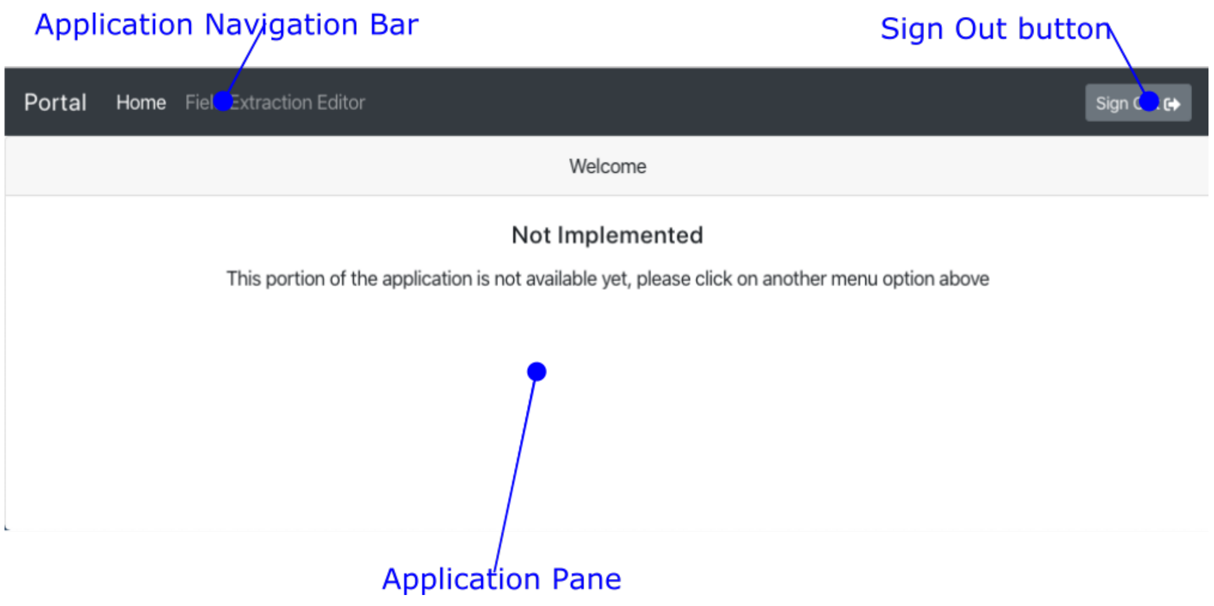

Browser
CloudFront
Lambda@Edge
Cognito
S3
Github
1 GET /github
2 SubSPAIndexMask()
3 matches a sub directory, object=index.html
4 GET /index.html
5 index.html
6 index.html
7 GET /github/<css, javascript, images, etc..>
8 <css, javascript, images, etc...>
9 <css, javascript, images, etc...>
10 <css, javascript, images, etc...>
11 Check for presence of Github bearer token in sessionStorage
12 GET /oauth/github/authorize
13 CFOauthEndpoints()
14 clientId,redirect_uri,next_uri, response_type
15 302 redirect to Github with clientId, redirect_uri
16 /login/oauth/authorize
Login or SignUp Scenario for new Users
17 302 Redirect to 'redirect_uri'
18 GET /oauth/github/access_token &clientId, code
19 CFOauthEndpoints()
20 POST /login/oauth/access_token &clientId, code
21 Return user's access token
22 Return user's access token and requesting web page
23 302 to requesting web page &githubtoken=<access_token>
24 Save access_token to sessionStorage
Browser
CloudFront
Lambda@Edge
Cognito
S3
Github

Figure 6: Sequence Diagram showing use of Amazon Cloudfront as a web server

4.2 User Interface

The focus of this application in its first iteration is not focused on the outcome of the graphical user interface or based on optimising user experience. The objective was to set up the back-end environment as well as solve the issues with users having to use the git tool. As such, the challenge in the web browser, as outlined in section 3.2, is to focus on file manipulation (creating, saving, updating, deleting) and to abstract the file structure as per R1 in table 1. This results in an interface that looks a lot like a standard Integrated Development Environment (IDE).

The customer provided a rudimentary layout demonstrated in figure 7



Not Implemented

This portion of the application is not available yet, please click on another menu option above

Figure 7: Customer Provided template

The first consideration is understanding the file navigation portion. As per figure 8, the concept of library *version* must be accounted for. There will also need to be a logout button to manage the GitHub authentication.

/fieldextraction.properties
/versions/**<version>**/vendor/**<vendor>**.properties
/versions/**<version>**/morphlines/**<morphline_config>**.morphlines

Figure 8: File Paths in GitHub Repository

In order to accomplish R1 in table 1, the user must be able to create a new vendor file, save a vendor file, delete a vendor file, add a pattern to a vendor file, create a morphlines file associated with a pattern, edit any of these files and then test the selected pattern. The following basic layout, as mocked up in figure 9, was proposed to the customer to accomplish these tasks. Along with the version and logout buttons discussed above, a navigation pane to add/remove vendors and patterns and an editor pane are being proposed. The editor pane will have three (3) tabs. The first will allow to edit the vendor properties file. The second will appear when a Morphlines pattern is being defined in order to allow editing the Morphlines file. The third will appear when a Morphlines file is selected in order to allow testing of the Morphlines definition.

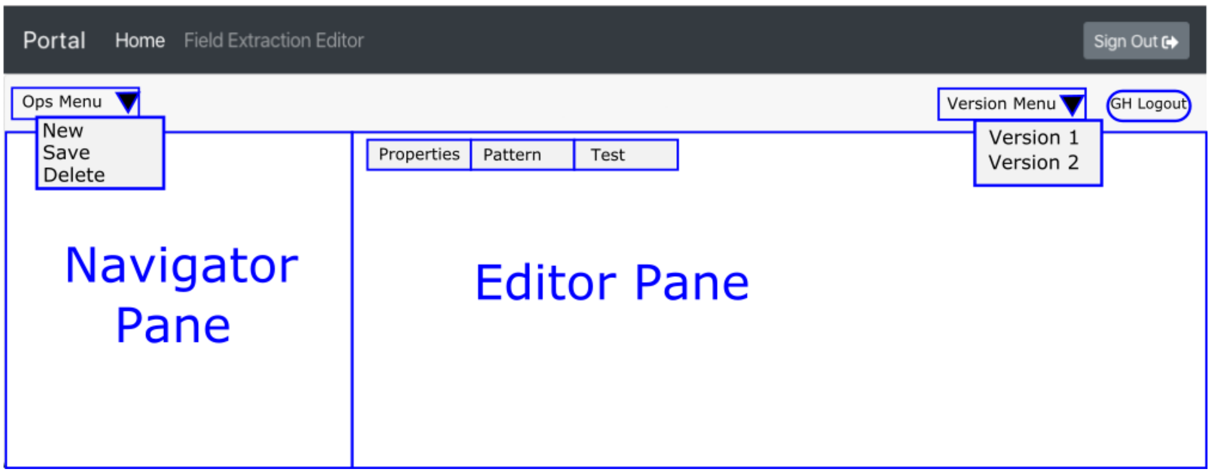


Figure 9: Proposed Layout

4.2.1 Navigation Pane

The navigation pane needs to take into account the relationships between the files being managed as shown in figure 10

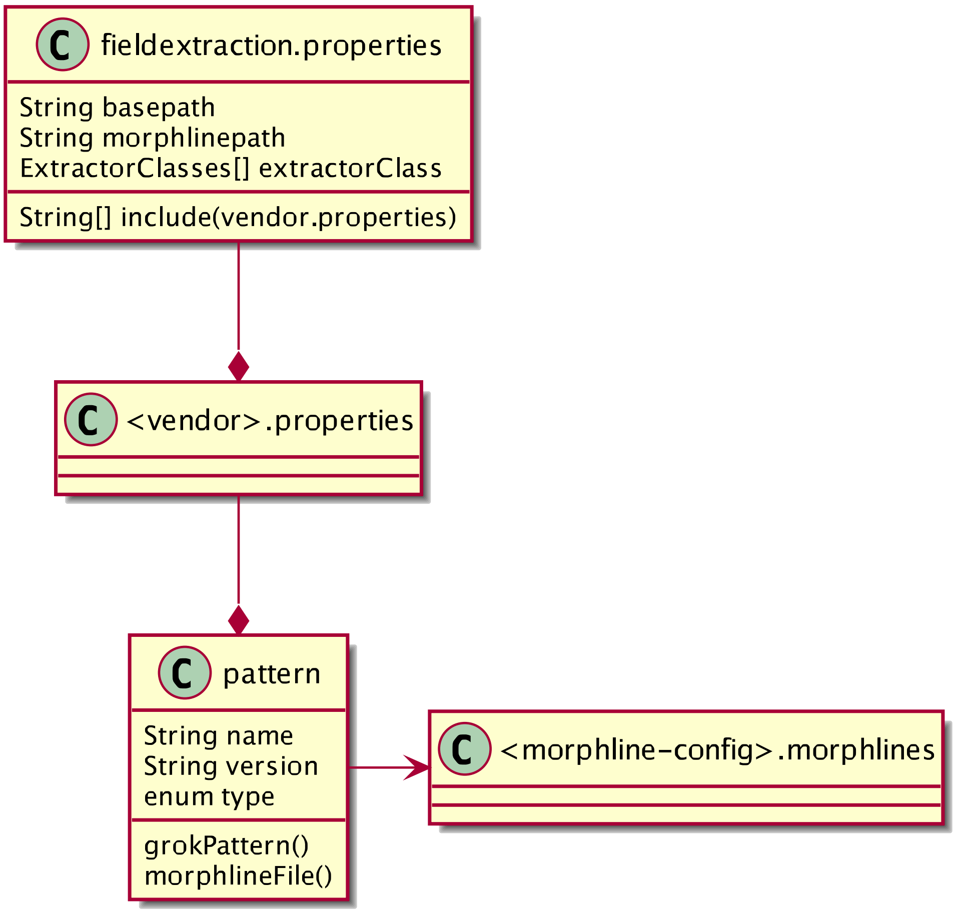

fieldextraction.properties
String basepath
String morphlinepath
ExtractorClasses[] extractorClass
String[] include(vendor.properties)
<vendor>.properties
pattern
String name
String version
enum type
grokPattern()
morphlineFile()
<morphline-config>.morphlines

Figure 10: File Object Relationship

A tree structure will be used with a node created for each vendor file and then children nodes for each pattern defined in the vendor file. When the root node is selected, the *add* and *delete* buttons will be available in the **ops** menu. When a *vendor* node is selected, the *add* option in the **ops** menu will be available to add a new pattern. When a *pattern* node is selected, the *delete* option will appear in the **ops** menu. The *properties* file also defines an *activeExtractors* variable that enables or disables the pattern. So when the *activeExtractors* defines the pattern as active, selecting the pattern will show *deactivate* and when *activeExtractors* defines the pattern as inactive (or not defined), selecting the pattern will show *activate* in the **ops** menu.

4.2.2 Editor Pane - Properties file tab

This pane is simple. It should display the text for the *vendor properties* file. Leveraging a pre-built component, such as the Monaco Editor component, should also allow to do code highlighting as well as *select* the patterns being edited. A *Save* button can optionally appear at the top of the editor pane if the file has been modified.

4.2.3 Editor Pane - morphlines config tab

If the pattern selected is a Morphlines, a tab should appear with the definition of the Morphlines file. If the file changed, a save button should appear at the top of the editor pane.

4.2.4 Editor Pane - morphlines test tab

The particularities of the test tab involve managing what can potentially be a large amount of log files. These can easily be stored into the web browser's "localStorage" or "sessionStorage" as explained in (MDN Contributors 2019c) but the amount of information can lead to a very busy interface.

Requirement R1.3 implies that the user is to supply and use "sample" logs. The approach chosen should involve reducing batches of log entries into a single item. Unfortunately, this approach has the drawback of hiding the underlying logs from the user which could lead to a user specifying log entries more than once. Nevertheless, the customer was not able to predict the actual workflow of the end-users in such a way that clarified the decision making process so *neat and tidy* was chosen as the driving factor. This pane is implemented as a component and can be easily swapped out or re-engineered per user feedback later if necessary.

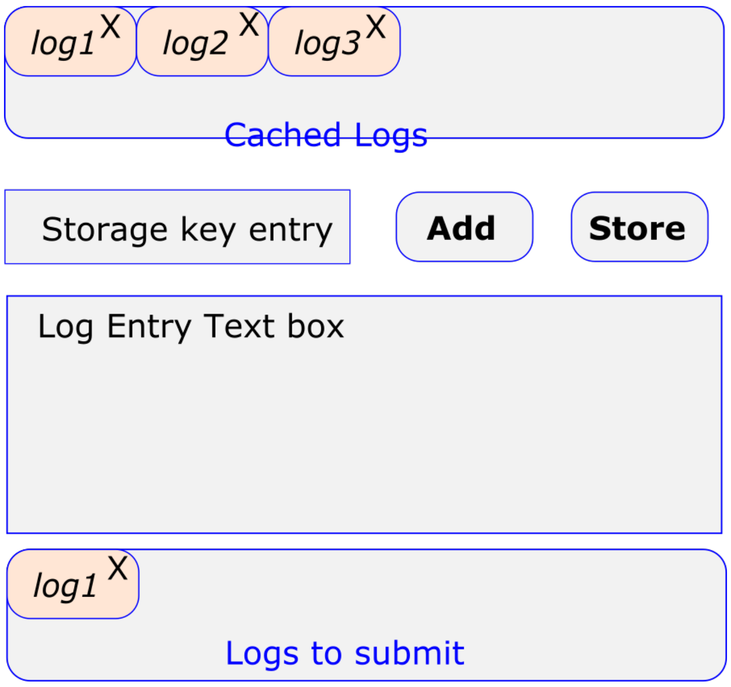
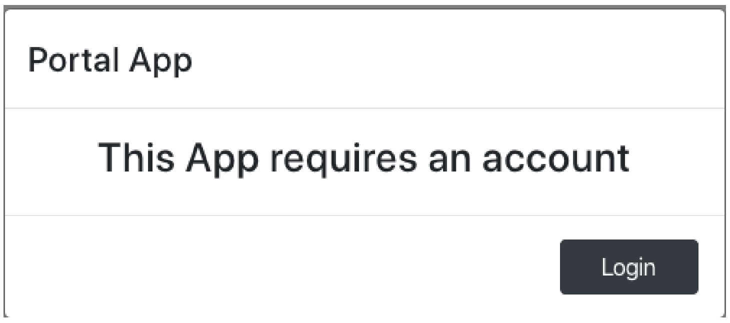


Figure 11: Log Manager widget

The log manager widget in figure 11 implements the required functionality. A user can input a label in the *key* input and place any number of logs to be submitted for testing inside the log entry box. To submit these logs, the user can *add* **log1** to the list of logs to submit or *store* the log to be used at a later date. This interface is compact and allows to easily vary the combination of logs being submitted for testing since this combination can vary from pattern to pattern.

4.2.5 Modals and dialogs

Many of the operations require providing some sort of wizard or form to the user. In a typical MPA this would be done using a *form view*. Because this project employs a SPA approach, using **modal** or **pop-up** dialogs is easier. Additionally, this is consistent with the approach used by the customer as part of their template which uses a *modal* dialog illustrated in figure 12.


Portal App

This App requires an account

Figure 12: Customer's Login modal dialog

These *modals* consist of a pre-defined *Card* offered by the React Bootstrap package. Modals will be required for the following use-cases:

- Login to Github (prompt)
- Fork the SecureOps Github repository into the user's account (wizard)
- Commit files to GitHub (wizard, save/delete multiple files)
- Create a new vendor (wizard)
- Create a new pattern (wizard)

For consistency, these *modals* will all use the layout provided by the customer in figure 12.

5 Implementation

Provided below are summary details of the implementations used to solve some of the issues researched in section 3. Most of the tasks were done in parallel with an attempt to follow the sequence outlined in section A.1. Above and beyond the research sections, most of the implementation also focused on the development of a Continuous Integration/Continuous Delivery (CI/CD) workflow to keep the development tasks

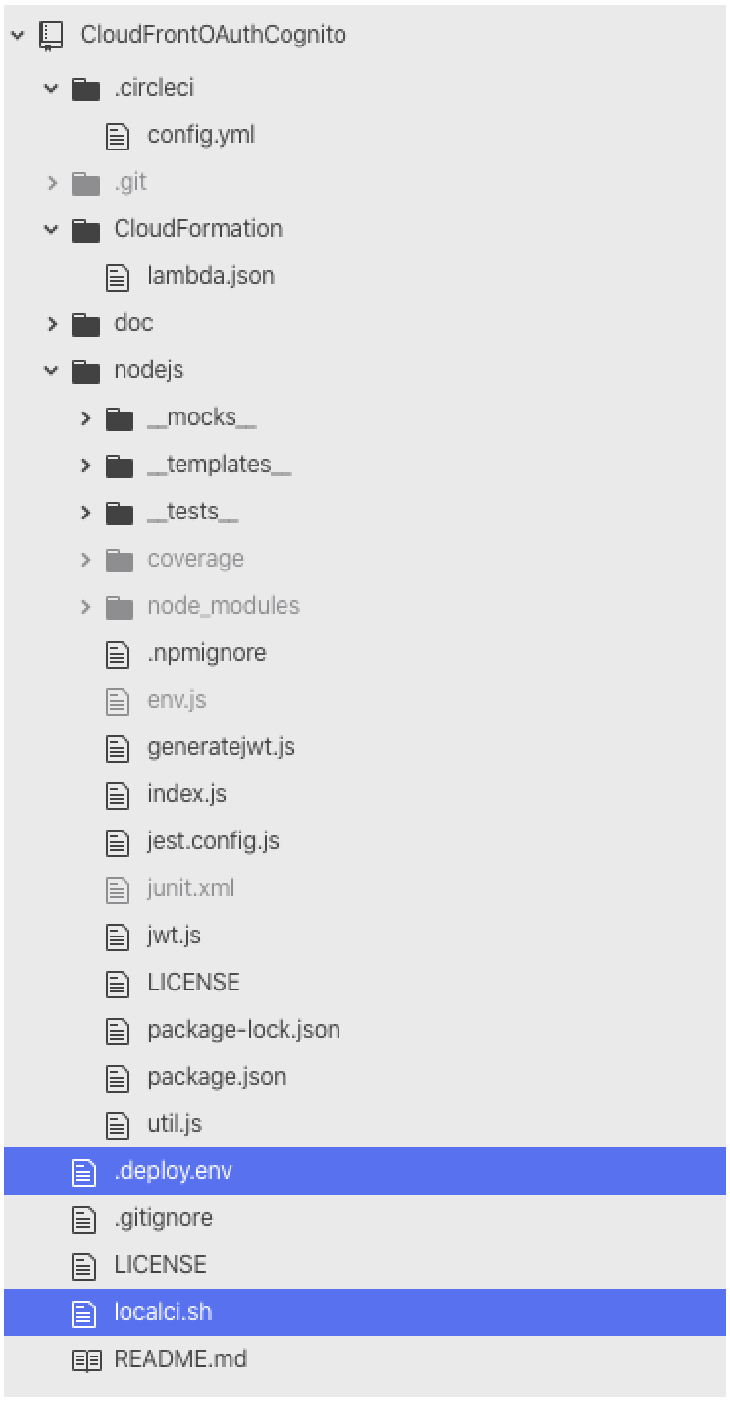
5.1 Amazon AWS

5.1.1 Continuous Integration/Continuous Delivery

The complexity of the Amazon AWS serverless environment, especially the repeated deployment of artefacts such as Amazon Lambda functions via Amazon AWS's Web Console can be time consuming and distracting since it requires leaving the development environment to go click buttons and fill out forms on a repeated basis. Further, having administrators keep track of each individual step and permutation can be complicated to communicate as well as maintain.

Since the manual method can also be error prone, a lot of time was taken to develop, in parallel with the actual code, the CircleCI scripts and the related Amazon CloudFormation scripts for each component. The result is that deploying iterations of each component on a test environment is relatively quick and reduces the impact of having limited unit tests because the developer can manually test in a development sandbox. This also helped with prototyping and gap-analysis because quick changes could be made, deployed and evaluated as a result.

Each module in this project uses a variation of the files listed in figure 13 and further detailed in appendix E. CircleCI and Amazon CloudFormation always reside in the *.circleci* and *CloudFormation* directories. The variables for the project are defined in *.deploy.sh* with an example in listing 18.



```
CloudFrontOAuthCognito
  .circleci
    config.yml
  .git
  CloudFormation
    lambda.json
  doc
  nodejs
    __mocks__
    __templates__
    __tests__
    coverage
    node_modules
    .npmignore
    env.js
    generatejwt.js
    index.js
    jest.config.js
    junit.xml
    jwt.js
    LICENSE
    package-lock.json
    package.json
    util.js
  .deploy.env
  .gitignore
  LICENSE
  localci.sh
  README.md
```

Figure 13: Filesystem for OAuth module

Note, listing 15 also does a lot of normalising of the environment. It abstracts many details from the developer and streamlines the output to provide metrics such as test coverage, library versions, etc. The typical developer experience can be summarised as:

1. The *.deploy.env* script defines the variables required to actually deploy the project.
2. The *localci.sh* transforms those variables to input parameters that can be read by the

.circleci/config.yaml listed in listing 15.

3. CircleCI defined in listing 15 then proceeds to (this varies slightly from component to component because some steps may not be required)
   (a) (line 12) Check out the code from git
   (b) (line 33) Download and install the dependencies of the module
   (c) (line 42) Run any local unit tests
   (d) (line 61) Generate any dynamic configuration files
   (e) (line 67) Package a .zip to upload to the environment
   (f) (line 81) Upload the .zip to the environment
   (g) (line 94) Execute the Amazon CloudFormation script to update the environment
   (h) (line120) clean up
4. Amazon CloudFormation defined in listing 16, does the following steps on subsequent runs
   (a) (line 22) Validate the permissions required by the function to operate
   (b) (line 40) Replace the JavaScript files and packages with the value uploaded by CircleCI above
   (c) (line 60) Export a pointer to the function to be used by other configurations.

The result of this work is that if unit tests fail, or logic is not implemented properly, all the programmer needs to do is correct the code and run the *localci.sh* script again. This also takes care of doing proper house-keeping, making sure version numbers are kept in sync and that labels and function names are consistent across the deployment. Since Amazon CloudFormation also defines the deployment as a *stack*, this can easily be referenced by other Amazon CloudFormation scripts, duplicated or deleted.

5.1.2 OAuth2

The integration of OAuth2 authentication implements the sequence referenced in figure 6. The lambda function in listing 17 uses the Amazon Lambda@Edge capabilities of Amazon Cloudfront in order to capture the URI of the request and attempts to match it with the appropriate step in the OAuth2 authorisation flow. The same function is used for both Amazon Cognito and GitHub. It could also be extended for any other OAuth2 *authorisation* code flow simply by adding details to the configuration file provided to the function. Note that the logic at line 44 is not required for our use-case but this function could also be used to mask every single *origin behavior request* to evaluate a Javascript Web Token (JWT) and provide authentication for the entire Amazon Cloudfront site.

The function isolates the OAuth2 provider at line 56 and then exits with an authentication failure at line 64 if the request doesn't exist. Line 75 checks if the request consists of an initiating request, at which point it returns a *302* redirect to the provider with the necessary query string parameters to authenticate the web app with the OAuth2 provider. Line 82 then matches the condition where the request is a redirect from the OAuth2 provider, including the *code* supplied by the OAuth2 provider in order to *fetch* the JWT on behalf of the user at line 127. Using the JavaScript *promise*, the function waits for the fetch function to return and then redirects the user to the web application with the JWT included in the redirect query string. The browser can then store this token and use it to authenticate, in the case of the Amazon Cognito token, to the */api* function implemented by Amazon API Gateway and in the case of the GitHub token, using the Octokit api.

5.1.3 Amazon Cloudfront path / index.html

The Amazon Lambda function in listing 3 is a little more concise. Since the function is mapped to the Amazon Lambda@Edge of a specific *origin behavior* as shown in figure 5, it only needs to validate whether the *url* ends with a / character and then modify the origin request to append index.html to the path in lines 14 and 17.

Listing 3: Amazon Lambda function that returns index.html when a path is detected

```
1  'use strict';
2  const util = require('util');
3  const {
4    URLSearchParams
5  } = require('url');
6
7  exports.handler = (event, context, lambda_return_cb) => {
8    console.log(util.inspect(event, {
9      showHidden: false,
10     depth: null
11   }));
12   const cfrequest = event.Records[0].cf.request;
13   if(cfrequest.uri.endsWith('/')){
14     cfrequest.uri = cfrequest.uri + 'index.html'
15   }
16   else {
17     cfrequest.uri = cfrequest.uri + '/index.html'
18   }
19   console.log(util.inspect(cfrequest, {
20     showHidden: false,
21     depth: null}));
22   lambda_return_cb(null, cfrequest);
23   return true;
24 };
```

5.2 Github API ReactJS Components

GitHub provides an HTTP API of the git environment. The relationships in figure 2 are also maintained by Octokit which is a NodeJS specific library to interact with this Application Programming Interface (API) using REST. Given the prototyping executed in section 3.2, the following wrappers were developed in order to interact with the persistence layer.

5.2.1 Retrieving a file

The code in listing 4 uses the Octokit library to retrieve a file. Note that the function is *asynchronous*, which means it returns a JavaScript *promise*. The function first checks, in line

7, if there's already a local version of the file stored in the *data* object. In line 11, the function essentially retrieves a *reference* to the **HEAD** of the current *branch* in order to subsequently retrieve the contents of the file. GitHub stores the content of the file in *base64* encoding, so line 17 decodes the content *blob* and then adds the file to the *data* object in lines 22 and 23.

Listing 4: Using GitHub Octokit API to retrieve a file

```
1   // Loads files from Github into the data state variable
2   getContents(path, type) {
3     const { client, reponame, owner, branch } = this.props;
4     const { data } = this.state;
5     return new Promise((resolve, reject) => {
6       // Check if the ini file is already in the data structure
7       if(data[path]) {
8         return resolve({decoded: data[path].decoded, raw: data[path].raw, path: path})
9       }
10      if(reponame && branch) {
11        client.repos.getContents({owner: owner, repo: reponame, ref: branch, path: path})
12        .then(raw => {
13          var content = null;
14          var decodedContent = null;
15          if (raw.data.size < 100000 && raw.data.encoding === 'base64'){
16            content = raw.data.content;
17            decodedContent = new Buffer(raw.data.content, 'base64').toString('ascii');
18          }
19          else {
20            console.log("Not decoding, size too big: " + raw.data.size );
21          }
22          data[path] = { raw: content, decoded: decodedContent, sha: raw.data.sha, type: type, changed: false, vcs: 'github'}
23          this.setState({data: data});
24          return resolve({ decoded: decodedContent, raw: raw.data.content, path: path});
25        }).catch(error => {
26          return reject(error);
27        });
28      }
29    });
30  }
```

5.2.2 Deleting files

Unlike file creation, file deletion is achieved by reconstructing a *tree* consisting of only objects that need to be kept, not deleted. This would require retrieving all the files in the repository in order to process a commit. As such, using individual commits for each file deletion and invoking the Octokit *deleteFile* was chosen as a compromise. The code in listing 5 executes this operation at line 15 but this entails a slight subtlety with regards to the *sequential* deletion of a file. Firstly, the deletion needs to occur on the **HEAD** *reference* on the git repository. This means that before each deletion, the **HEAD** *reference* must be retrieved which is done at line 10. In order to force JavaScript's asynchronous promises to execute each deletion sequentially, the files to be deleted are stored inside an array and each file is *popped* in line 9 and the remaining files are recursively passed to *deleteContents()* at line 25 within the *then* block of the *deleteFile* which ensures that the subsequent *deleteContents()* is not called until the previous invocation is completed.

Listing 5: Using GitHub Octokit API to delete a file

```
1   deleteContents(files, message) {
2     const { client, reponame, owner, branch } = this.props;
3     const { data } = this.state;
4
5     if (!files || files.length === 0) {
6       return new Promise((resolve) => resolve());
7     }
8     return new Promise((resolve, reject) => {
9       const path = files.pop();
10      client.repos.getContents({
11        owner:owner,
12        repo: reponame,
13        path: path
14      }).then((file) => {
15          client.repos.deleteFile({
16            owner: owner,
17            repo: reponame,
18            path: path,
19            branch: branch,
20            sha: data[path].sha,
21            message: message,
22            committer: { name: 'BobSquarePants', email: 'scooby@do.tv'}
23          }).then(() => {
24            if(files.length > 0) {
25              this.deleteContents(files, message);
26            }
27            resolve();
28          })
29      })
30    });
31  }
```

5.2.3 Creating and Editing files

Creating and editing files is not a straightforward affair. Listing 19 shows that this operation requires to retrieve the last commit tree (line 11). Then a *base64 blob* needs to be created for each file in the tree (line 18) that was added or modified. A new *tree* must then be created and updated with all the files to commit (line 33) and once this tree is created, the *commit* can proceed (line 39).

5.3 Properties files serialisation and deserialisation

This implementation represents the biggest challenge of this project. Many *ini* or *properties* libraries exist for JavaScript but none of them implement the functionality to *include* files. The issue of preserving whitespace and comments, as per section 3.2, is also not included. The library supplied by Node Package Manager at https://github.com/npm/ini initially contributed by Isaac Z. Schlueter was used as a base to further implement the missing features. Note that this library is not completely *object oriented* and no attempt was made to refactor the code to accomplish this. This leads to some difficult code as the environment and options of the *ini* object must be kept inside the parent component and passed down into the object on each use of a function.

Listing 6: Original .ini object as provided by https://github.com/npm/ini

```
{
        "key": "value",
        "sectionkey": {
                {
                        "key" : "value"
                } ,
                "arraykey": [
                        {  "value1", "value2", ... , "valuen" }
                ],
                "subsectionkey": {...}
        }
}
```

The *ini* function initially serialises the *ini* file into the structure listed in 6. The problem with this structure is that it doesn't relate the initial text state of the object. It also doesn't hold into account the original value and the substituted value of each key. Finally this structure also doesn't allow to contain *ini* configurations that spawns multiple files. To accommodate all these shortcomings, the library was modified to produce the structure in listing 7.

Listing 7: Modified ini library

```
1  {
2    ".": // key that contains all the lines or positions of text not associated with actual ini objects
3    {
4      "type": "section",
5      "values": [
6        {
7          "content": "line of text from file, null for start and end of document",
8          "nextline" : "pointer to next line",
9          "prevline" : "pointer to previous line",
10         "source" : "filename that contains the line",
11         "value": null,
12         "virtualvalue": null
13       }
14     ],
15     "children": null
16   },
17   "toplevelattributekey": {
18     "type": "attribute",
19     "values": [
20       {
21         "content": "line of text from file",
22         "nextline" : "pointer to next line",
23         "prevline" : "pointer to previous line",
24         "source" : "filename that contains the line",
25         "value": "value of the key in the case where the type is attribute as represented in text file",
26         "virtual-value": "value of the key with variable substitutions where necessary"
27       }
28     ],
29     "children": null
30   },
31   "sectionkey": {
32     "type": "section",
33     "value": null,
34     "children": {
35       "sectionattributekey": {
36         "type": "attribute",
37         "values: [
38           {
39             "content": "line of text from file",
40             "nextline" : "pointer to next line",
41             "prevline" : "pointer to previous line",
42             "source" : "filename that contains the line",
43             "value": "value of the key in the case where the type is attribute as represented in text file",
44             "virtual-value": "value of the key with variable substitutions where necessary"
45           }
46         ]
47       }
48     }
49   }
50 }
51 }
```

Lines 8,9,22,23,40,41 implement a **linked list** that represents the sequence of lines inside the files denoted by the related *source* attribute in lines 10,24,42 of listing 7. The beginning of each file can be found by searching the "." section for *prevline* with a value of *null* for this given *source* and the end of the file can also be found by searching for *nextline* of *null*.

The original library also needed to be modified to allow retrieval and processing *include* files using a callback and then to merge all the resulting objects together to produce the overall structure.

Finally, this new structure allows to quickly locate the position of a section or attribute within a document and to add new documents. All these functions were added to the original library.

With these changes, it is now possible to track the hierarchy of the *ini* files inside the GitHub repository and decouple this from the user interface code by providing a unified interface.

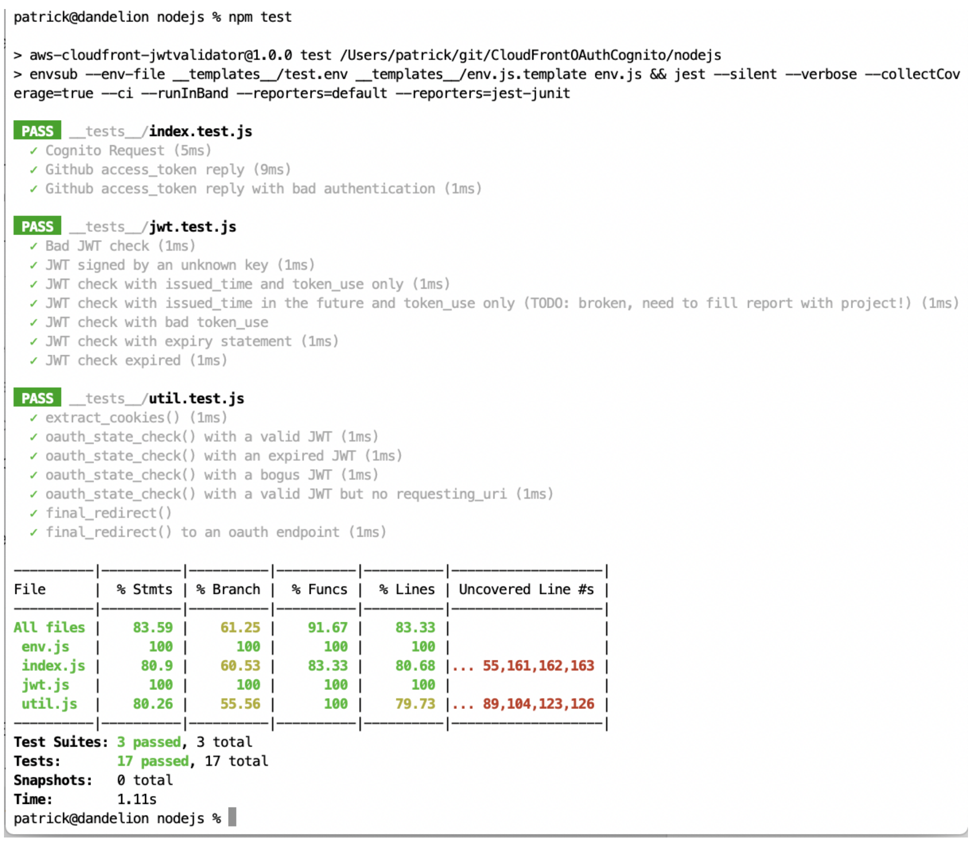
6 Testing and Customer Acceptance

The original project specification hinted that the use of *test-driven development* wasn't necessary. During the development of the *server-side* components, such as the OAuth2 component, it was found that using *unit tests* driven by *mock* implementations was more time-efficient in terms of validating proper functioning of the code versus deploying and manually testing these functions *in place*.

Customer acceptance was also important and so a formal plan was devised and sent to the customer covering all the use-cases listed in table 6.

6.1 Unit testing

Unit testing was executed on NodeJS server-side components. The results for the OAuth2 code are outlined in figure 14 as a sample of how unit testing was performed, showing all the different test cases as well as the code coverage.



```
patrick@dandelion nodejs % npm test

> aws-cloudfront-jwtvalidator@1.0.0 test /Users/patrick/git/CloudFrontOAuthCognito/nodejs
> envsub --env-file __templates__/test.env __templates__/env.js.template env.js && jest --silent --verbose --collectCov
erage=true --ci --runInBand --reporters=default --reporters=jest-junit

PASS  __tests__/index.test.js
  ✓ Cognito Request (5ms)
  ✓ Github access_token reply (9ms)
  ✓ Github access_token reply with bad authentication (1ms)

PASS  __tests__/jwt.test.js
  ✓ Bad JWT check (1ms)
  ✓ JWT signed by an unknown key (1ms)
  ✓ JWT check with issued_time and token_use only (1ms)
  ✓ JWT check with issued_time in the future and token_use only (TODO: broken, need to fill report with project!) (1ms)
  ✓ JWT check with bad token_use
  ✓ JWT check with expiry statement (1ms)
  ✓ JWT check expired (1ms)

PASS  __tests__/util.test.js
  ✓ extract_cookies() (1ms)
  ✓ oauth_state_check() with a valid JWT (1ms)
  ✓ oauth_state_check() with an expired JWT (1ms)
  ✓ oauth_state_check() with a bogus JWT (1ms)
  ✓ oauth_state_check() with a valid JWT but no requesting_uri (1ms)
  ✓ final_redirect()
  ✓ final_redirect() to an oauth endpoint (1ms)

----------|----------|----------|----------|----------|-------------------|
File      |  % Stmts | % Branch |  % Funcs |  % Lines | Uncovered Line #s |
----------|----------|----------|----------|----------|-------------------|
All files |    83.59 |    61.25 |    91.67 |    83.33 |                   |
 env.js   |      100 |      100 |      100 |      100 |                   |
 index.js |     80.9 |    60.53 |    83.33 |    80.68 |... 55,161,162,163 |
 jwt.js   |      100 |      100 |      100 |      100 |                   |
 util.js  |    80.26 |    55.56 |      100 |    79.73 |... 89,104,123,126 |
----------|----------|----------|----------|----------|-------------------|
Test Suites: 3 passed, 3 total
Tests:       17 passed, 17 total
Snapshots:   0 total
Time:        1.11s
patrick@dandelion nodejs %
```

Figure 14: Unit tests for OAuth2 component

Unit testing involves hard-coding the return values of external components in order to validate that the code handles and behaves as expected based on various situations. Doing this is commonly known as *mock*ing a function call. Line 11 of Listing 8 shows how to hard-code the return value of the *sign()* function of the *jsonwebtoken* library to a value that can be compared to an expected return value as demonstrated in line 5. Therefore, the *expect()* statement in line 22 can use a consistent value to test against.

Similarly, line 16 of listing 8 shows how an entire function can be mocked, where logic can be manipulated to return expected results from the external library.

Listing 8: Test case using a mock function from an external library

```
1  {
2  const returnEvent = {
3    "headers": {
4      "location": [{
5        "value": "/sendmehere?githubtoken=abcdef0123456"
6      }]
7    },
8    "status": 302,
9    "statusDescription": "Authenticated, redirecting..."
10 };
11   jsonwebtoken.sign.mockReturnValue("abcdef0123456789");
12
13   jsonwebtoken.decode.mockImplementation((token, params) => {
14     return jwt.decode(token, params);
15   });
16   jsonwebtoken.verify.mockImplementation((jwtToken, pem, params ) => {
17     return jwt.verify(jwtToken, pem, params);
18   });
19   const token_response = { body: { access_token: "abcdef0123456" }, status: 200};
20   fetchMock.post("begin:" + env.OAUTH_PROVIDERS.github.provider_url, token_response);
21   function callback(err, data) {
22     expect(data).toEqual(returnEvent);
23     done();
24   };
25   lambda.handler(requestEvent, context, callback)
26 });
```

Finally, the coverage report in figure 14 provides an indication of how much and which parts of the code has been executed as part of the testing. These metrics can be provided to quality assurance teams and also offers benefits in terms of change control to make sure older code components are still working with newer versions of libraries.

6.2 User Testing

User testing was executed by providing a step-by-step walkthrough in the form of a user manual provided in appendix C. The table was provided and approved by the customer to execute each step and report if the step worked as expected. The customer was encouraged to report bugs, using the test case/step number into the website's GitHub repository. A first review was done according to the project specification timeline in figure A.1 as demonstrated in section C.1.1 and then corrected and revised in section C.1.2

Table 6: Test Cases

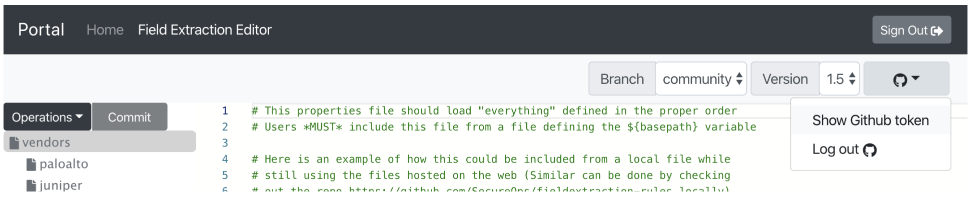
TestID	Description	Success
T2.1	Login button works	
T2.2	Signup works	
T2.3	New password works	
T2.4	Validation email received	
T2.5	Verification works	
T2.6	Landing page works	
T3.1	Github log successful	
T3.2	Github fork successful	
T3.3	Github repository verified	
T3.4	FieldExtraction page appears	
T3.5	Github menu appears	
T3.6	Github token displayed	
T4.1	paloalto properties file displays	
T4.2	morphline parser renders in config tab	
T4.3	Ini tab displays parser lines highlighted	
T4.4	Test tab shows Log Manager	
T5.1	Log text entry works	
T5.2	Log is added to logs to submit	
T5.3	Test shows waiting spinner and returns a value in the Results pane	
T5.4	Cache button adds log to logs to cache	
T6.1	Vendors hightlights	
T6.2	Add prompt for vendors selection appears when Operations clicked	
T6.3	New vendor dialog appears	
T6.4	Empty vendor file is presented	
T6.5	Operations->Add for a vendor file selection appears	
T6.6	New Section dialog appears	
T6.7	New morphlines results in a Config tab with a template morphline defined	
T6.8	Ini tab displays the morphlines parser definition and highlights it	
T6.9	Operations -> Activate is available	

Continued on next page

TestID	Description	Success
T6.10	activeExtractors= ini is added	
T6.11	Commit button is active and available	
T6.12	Commit dialog appears as described	
T6.13	.properties file is in Github repository	
T6.14	.properties file matches content in web application	
T6.15	.morphlines file is present on Github	
T6.16	Test tab contains cached log	
T6.17	Selecting log and pressing Add places log into logs to submit	
T6.18	Test button returns partial matches	
T6.19	N/A	N/A
T6.20	Operations -> Add shows new section dialog	
T6.21	New section with Grok type shows Grok Pattern input box	
T6.22	Activate/Deactivate adds and removes the activeExtractors= line	
T6.23	Commit modifies only the .properties file	
T7.1	Operations->Delete removes the parser definition from the .properties file	
T7.2	Commit shows only 1 .properties file	
T7.3	Operations -> Delete is available for the new vendor	
T7.4	Include line is removed from the top-level .properties file	
T7.5	Commit operation shows top-level file in changed files and vendor file in delete	
T8.1	Github logout is available and results in Github login screen	
T8.2	Cancel results in the landing page or login prompt	

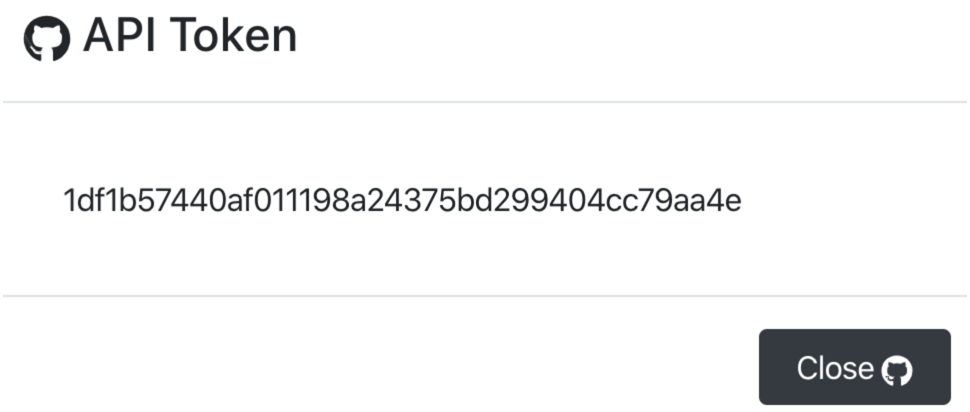
6.2.1 Simulation Environment

No unit tests were executed and developed in the ReactJS environment but it was quickly established, especially due to OAuth2 requirements from GitHub API calls, that some mechanism needed to be explored to run the NodeJS developer environment locally on the workstation. NodeJS allows to *proxy* certain paths as described at https://create-react-app.dev/docs/proxying-api-requests-in-development. The requests are proxied to an ExpressJS hosted site that uses PUG templates to emulate the Github login locally. This allows the developer to copy/paste the GitHub bearer token from a real session into a form to *trick* the web browser that it obtained the token and use it for GitHub API calls. This prompted to create a *show GitHub token* modal inside the user interface to facilitate this functionality. The process is shown in figures 17, 16, 18, 15 below.



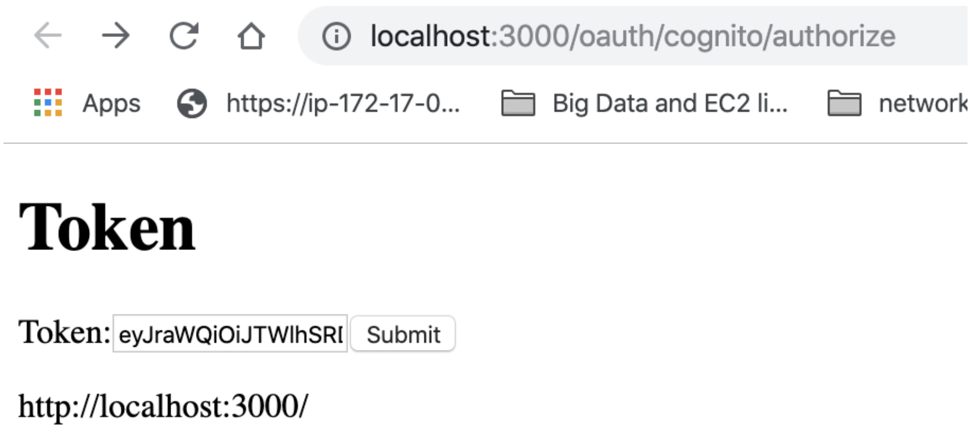
```
# This properties file should load "everything" defined in the proper order
# Users *MUST* include this file from a file defining the ${basepath} variable

# Here is an example of how this could be included from a local file while
# still using the files hosted on the web (Similar can be done by checking
# out the repo https://github.com/SecureOps/fieldextraction-rules locally)
```

Figure 15: Using the proxy to supply the OAuth2 token



1df1b57440af011198a24375bd299404cc79aa4e

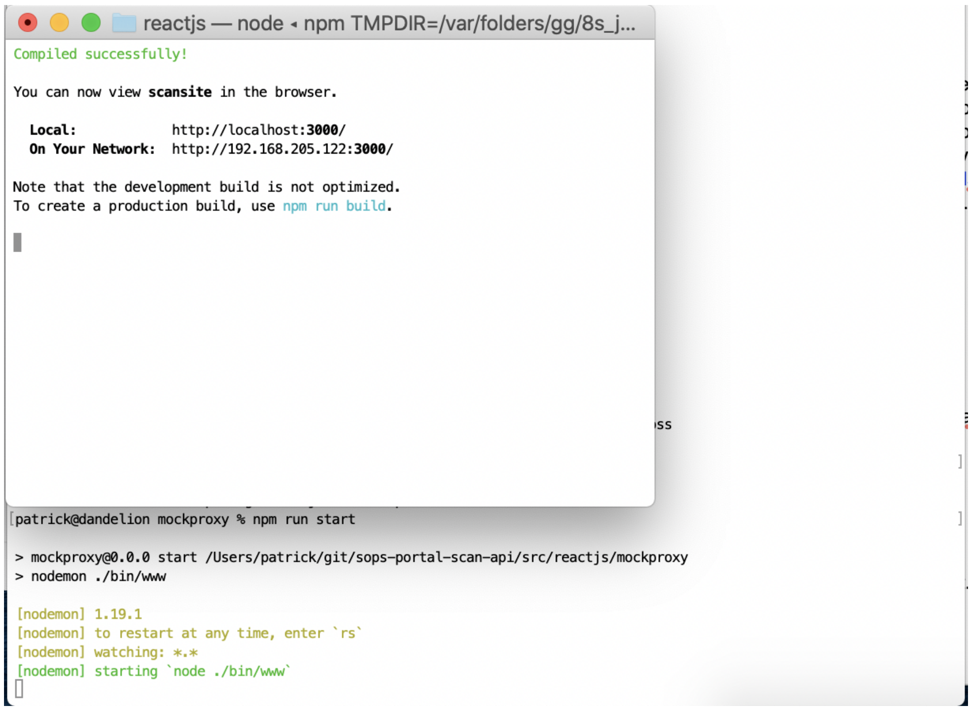
Figure 16: GitHub token dialog



Token: eyJraWQiOiJTWlhSRl

http://localhost:3000/

Figure 17: GitHub submit token menu



```
Compiled successfully!

You can now view scansite in the browser.

  Local:            http://localhost:3000/
  On Your Network:  http://192.168.205.122:3000/

Note that the development build is not optimized.
To create a production build, use npm run build.
```

```
patrick@dandelion mockproxy % npm run start

> mockproxy@0.0.0 start /Users/patrick/git/sops-portal-scan-api/src/reactjs/mockproxy
> nodemon ./bin/www

[nodemon] 1.19.1
[nodemon] to restart at any time, enter `rs`
[nodemon] watching: *.*
[nodemon] starting `node ./bin/www`
```

Figure 18: ReactJS development server proxy using two node servers

7 Evaluation

As described in this document, this project has successfully touched all the traditional phases of the waterfall model. The provided user environment completely replaces the customer's previous manual process that was required to manage the ETL artefacts and the testing and management of parsers has been greatly simplified. Consequently, the primary research objective from the project specification has satisfactorily been achieved.

Delving into the GitHub capabilities, the git protocol has also been successfully used to manage the versioning of parser definitions and using the *fork* and *commit* features prevents users from overwriting each other's changes and lends itself nicely to allowing end-users to use whichever repository they wish to customise for themselves while also providing the customer with a capability to assimilate parser contributions into their own public repository.

Many valuable experiences have been collected throughout this project as elaborated in section 7.2 below. The customer has successfully tested and accepted the deliverables and is looking forward to pursuing this project into the next iteration. This project provided a great experience in software development and has solidified the importance of requirements gathering, research, project planning and testing in the realisation of objectives. It is clear that in situations where the definition of success can mean different things to different stakeholders these phases of a project greatly increase the chances of success.

7.1 Requirements Review

Included in table 7 is a copy of table 1 with a status statement to indicate the progression of this project upon submission. All major objectives were completed.

Table 7: Customer Requirements Status

ReqID	Description	Test/Approval	Status
R1	Provide a graphical user interface to author Morphline HCON files	T4, T5, T6, T7	Complete
R1.1	User should be able to create new patterns	T6.5, T6.7	Complete
R1.2	User should be able to edit existing patterns	T4.1	Complete
R1.3	User should be able to test patterns using sample logs	T5.3, T6.16, T6.17, T6.18	Complete
R1.4	User should be able to mark existing patterns as deprecated	T6.8, T6.9	Complete
R2	Interface should be presented in a workbench style presenting all the parsers available	T4, T6	Complete
R3	Architecture must manage the lifecycle each user's contribution	T6.11, T6.12	Complete

7.2 Lessons learned

Multiple technical difficulties prompted fairly in-depth research and forced the understanding of technologies associated with this project. Specifically, the underpinning of the git operations were especially fascinating. Unfortunately, in this particular case, curiosity caused a loss of focus that took away from the concept of MVP and the added complexity of trying to accomplish an atomic commit lead to what could be perceived as a slightly erratic user experience. Although this is acceptable for a first iteration it requires to be refined further. It also introduced additional risk in taking time away from other activities that could have caused the project not to succeed.

Time management was also exceedingly difficult. The scope of the project, including the research, was defined loosely by the customer and greatly open to interpretation.The availability of human resources from the customer left limited time for feedback especially for confirming early artefacts. Luckily, the project specification established a sequence of tasks that were loosely respected but could have been scoped to include milestones that could have been explicitly approved by the customer in order to avoid the availability situation.

The project management could have been tracked in a more meticulous fashion tying the commit logs provided in appendix B with actual project plan items specified in section A.1. Perhaps an agile scrum/sprint approach would be more conducive to keeping this phenomenon under control rather than the selected waterfall model since the time-tracking would be updated on a weekly basis.

7.3 Ideas for next iteration

The progression of the overall application required many decisions and glossed over many discoveries that were deferred to *further iterations*. Focus needed to be maintained on the MVP perspective. Listed below are some items that can be considered for the next iteration of the product.

1. Convert the ini library to a proper object-oriented class. (This will simplify the interface code greatly)
2. Review the Octokit implementation to take into account new plugins that would allow for atomic commits for creation, delete and updates of multiple files at once.
3. Abstract CSS from all components to allow theming of the application
4. Correct navigation tree to have +/- buttons for items to make operations clearer to the end user
5. Revise the log submission interface based on user feedback
6. Add a panel to manage git *pull requests* back into the original repository.
7. Automated unit testing framework for the UI components.
8. Code highlight for HCON language
9. Serialisation/Deserialisation of HCON files to provide a programmatically manipulatable data structure

8 Conclusion

The final deliverable has successfully met the customer's objective to provide their users with a workbench style web-based application to author new parsers. The customer has expressed optimism in rolling this out into their production environment as part of the initial launch of their portal once a few other applications have been finalised early in the new year. The customer sees value in the final project as it provides an interface to produce community content that will help *normalise* the ETL capabilities of third-party products such as Elastic-Search as well as push for a *key/value* based **format** to be **standardised** upon within the log management community.

Personally, this project has exposed the importance of activities that go beyond coding and integration required to accomplish a successful product delivery. In this light, it is recognised that the product as it stands is still only suited for advanced ETL authors and would still require a certain amount of work, especially in the realm of serialising and deserialising HCON which would then open the possibility to provide a WYSIWYG component instead of a text-editing interface for Morphlines file authoring.

Despite this limitation, the product is complete according to the originally intended deliverable and the outlook is optimistic that the customer will want to push the feature-set further in the future.

A Project Specification

page left blank

System Log Tokenization/ETL Workbench

Patrick Ethier, April 3, 2019

1 Outline Description

The focus of this project is to conceive a free-to-use , web-based Extraction, Transformation and Loading (ETL) workbench aimed at parsing various system logs. The underlying motivation from the customer is to produce a tool which has the potential to enable the creation of an online community to share and distribute ETL patterns and provide the change control process required to ensure the integrity and functionality of community authored content.

The research will be focused on providing this functionality by using cloud-based technologies. The uncertainty to be addressed resides in how to leverage this infrastructure, which is already imposed by the target customer because they already have the basics of a services portal in place, to deliver the final product. Thus, the areas for investigation are expected to be heavily involved with the Amazon AWS environment, especially pertaining to identity management, databases, execution and workflow management.

2 Background Research and Methodology

The problem will be approached by developing a mechanism that allows to browse the available ETL definitions, to modify them and then provide the changes back to the community.

2.1 State of the Art

The following background research was executed by searching for recommended architectures and technologies in publications provided by the SANS reading room and blogs. Professional input was also gathered from discussions with subject matter experts employed by the customer with regards to what technologies they were familiar with was also used.

According to searches on the SANS reading room for terms like "log parse", "log processing" and "log transformation", (SANS 2019), most of the research papers and blog posts referenced tools that are either outdated or proprietary. For example, the prevalent search results reference a tool called "Log Lizard" from Lizard Labs which is uniquely geared towards Windows architecture (LizardLabs 2019).

Conversly, as stated by the customer's subject matter experts, most of the log aggregation tools supply an ETL type of functionality. Free tools, such as ElasticSearch, provide facilities to load plugins or define patterns but no actual comprehensive repositories of patterns exists (Elasticsearch 2019). Commercial tools, such as IBM QRadar, claim that, "QRadar can read and interpret events from more than 300 log sources", (IBM 2019), but this is contingent upon the product being purchased and used in the data processing pipeline to leverage.

2.2 Approach

The project planning phase will use a mostly "linear" methodology, as described in (**pp. 1**; CMS 2008) and as presented in figure 1. This choice was made due to the fact the project will be executed mostly by a single person with a few, very clear, interaction points with external entities such as the customer and beta-testers.

In the research phase, the primary evaluation method to determine the best design and technology decisions will include gap and S.W.O.T. analysis. This will have the benefit of providing a clear measurement to determine the feasibility of each major component with respect to the objectives as well as the compatibility of each major technology with respect to the existing components provided by the customer.

The overall development process should include using standard Continuous Integration/Continuous Delivery (CI/CD) environments which are undetermined at the time of this writing and may be subject to the customer's requirements upon further investigation. This will enable a test-driven approach that will help reduce the risk of components not working as intended and also minimize the amount of manual verification required in the testing phase.

The potential drawbacks to this approach is that the plan could drastically change at any moment due to the knowledge and experience gained in any of the phases. Careful consideration must be given to the scope of the problem and the resulting implementation. The lack of prototyping may also result in missing some key architectural problems that may appear later on in the implementation. As such, there is no guarantee that a gap-analysis or S.W.O.T. evaluation will ultimately provide the proper choice for implementing a requirement.

2.3 Research Questions

The following criteria were discussed with the customer and must be considered in order to address this problem:

Firstly, the user experience, especially with respect to developing ETL patterns, needs to be as graphically driven as possible. Ideally, the workbench should provide a visual logic/programming approach similar to what is provided by Cyberchef.

Secondly, change control over the pattern database is required. As such, the interaction between the community driven ETL pattern database, currently hosted and distributed via Github, and ongoing user modifications will require particular attention. There needs to be a mechanism which prevents a specific community member from overriding or damaging the ETL pattern database and ruin the experience for other users.

Tertiary issues may also affect user experience such as the ability for the user to submit or recall log entries, save temporary ETL definitions without submitting them, etc. The resulting solution will need to provide at least one mechanism to allow users to load and/or recall system logs into the interface.

2.4 Research Ethics

Due to the potential sensitivity of the information that may be contained within system logs, special consideration must be taken to provide for the security, especially the confidentiality, of logs that may be provided by individual users to the system. User information gathered and retained should be kept to a minimum. Users of the system should be able to participate by providing the minimum amount of information to confirm them as an individual.

3 Personal Objectives

The rationale for choosing this project stems from my previous experience as a security systems architect. Having encountered the need to integrate ETL definitions into various projects has always taken considerable time especially with respect to technologies that don't have predefined ETL patterns for the devices I wish to monitor. By contributing the capability to develop and share these patterns I am hoping that it provides an opportunity for my colleagues and I to finally have a tool that alleviates this issue. The learning outcome of this project will hopefully allow me to gain skills with respect to the user interface design and user experience. I have extensive knowledge of "back-end" development and programming using Python, Java and Structured Query Language (SQL). Applying this knowledge to a new platform, such as Amazon AWS and learning how to program using a Single Page Application (SPA) oriented design will allow me to add "front-end" development experience to my existing list of skills.

Further, the integration of Continuous Integration/Continuous Delivery (CI/CD) promised by Amazon AWS is interesting to me as it represents, in my opinion, not only the future of software development, but the future of IT technology in general.

By completing this project, I hope to accomplish the objective of adding a full-fledged UI design, including automated unit and integration testing , to my current skillset. I also hope to further my experience level pertaining with how to continuously develop, integrate, test and deploy applications in a cloud environment.

4 Technical Description

The foreseen layout of this project is to use Amazon AWS technologies in order to achieve the objectives. The graphical user interface will be a web-based user interface. Research will need to be done to choose a suitable web-application framework which will provide for the use and integration of underlying technologies. This interface must be supported by Amazon AWS's array of products. The environment provided by the customer already uses Amazon Cognito for user management, Amazon API Gateway for web API hosting and Amazon Lambda as a logic execution environment.

This existing cloud environment will need to be expanded to include further tools and technologies in order to provide the functionality required by the use-cases brought forth by this project. This is expected to interface with a storage strategy that will allow to store stateful information and require interaction with various data repositories and streaming functions to give access to end-user logs. As such, research related to the processing of this data will need to be performed to identify a suitable candidate strategy that can be used in a dynamic web application to achieve this goal.

The current use of Github as a distribution mechanism by the customer will also need to be re-evaluated and special consideration to Github's API must also be taken into account. It is the impression, at the time of this writing, that the solution to the change control problem will be somehow related to Github's branching and tagging technology.

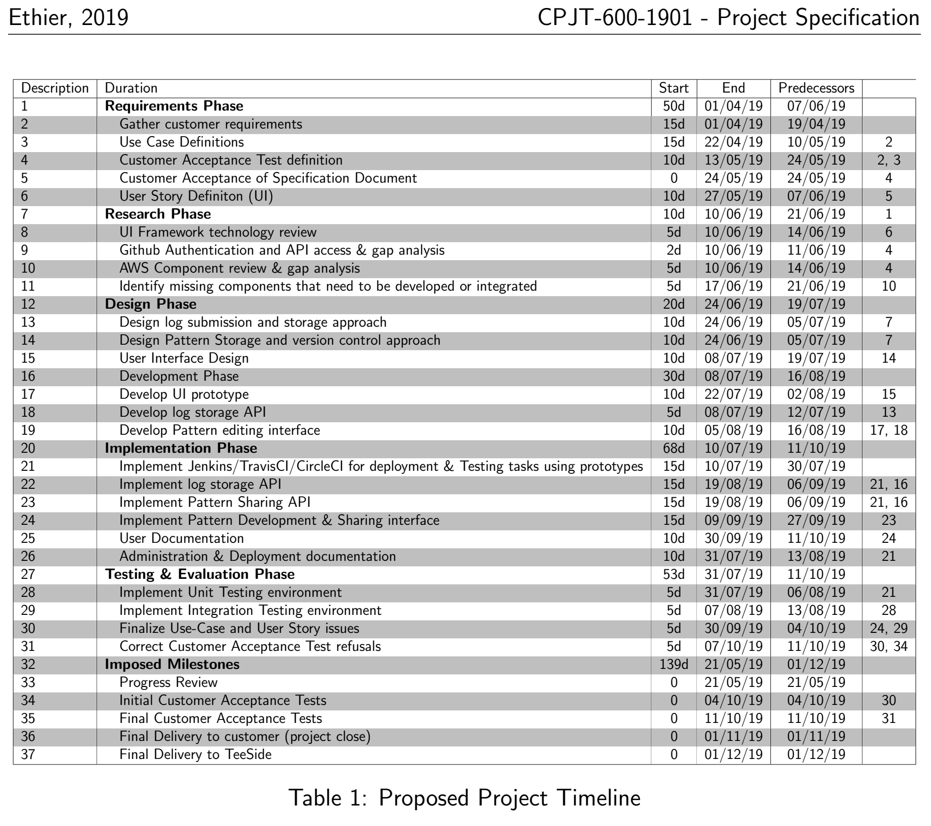
5 Intended Product and Project Deliverables

The physical deliverable will primarily be a web site. The supporting infrastructure will also be provided in the form of instructions to configure the various components of Amazon AWS' infrastructure services required to integrate the final solution. All source code will be hosted on Github either under the customer's account or a personal account.

A series of deployment and maintenance guides will also be provided to enable the deployment of the solution. Finally, a user tutorial will also be provided in order to demonstrate the intended usage of the workbench to end-users.

6 Project Plan

The proposed document timeline is listed in figure 1 and represented in the Gantt chart provided in the Appendix. Time allocated is measured in days, accounting for an average of a 4-hour workday and skipping weekends. Each time estimate was formulated by breaking down each major desired outcome , as described in the *Research Questions* and *Technical Description* sections of this project, and spreading them over the various project phases. At the time of this writing and as suggested by (MindTools 2018), a bottom-up approach has resulted in this list of tasks which align the project for completion shortly before the required deadline. The produced WBS also assumes tasks were overestimated by a factor of 2 due to the fact that many of them may change pending the research and design phases.

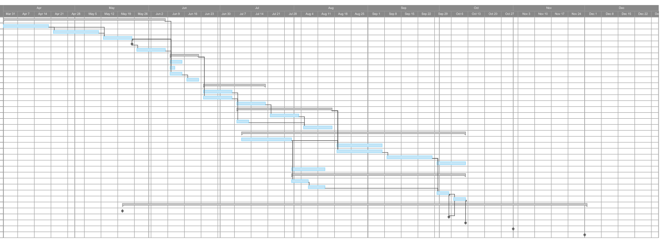


Description	Duration	Start	End	Predecessors	
1	**Requirements Phase**	50d	01/04/19	07/06/19	
2	Gather customer requirements	15d	01/04/19	19/04/19	
3	Use Case Definitions	15d	22/04/19	10/05/19	2
4	Customer Acceptance Test definition	10d	13/05/19	24/05/19	2, 3
5	Customer Acceptance of Specification Document	0	24/05/19	24/05/19	4
6	User Story Definiton (UI)	10d	27/05/19	07/06/19	5
7	**Research Phase**	10d	10/06/19	21/06/19	1
8	UI Framework technology review	5d	10/06/19	14/06/19	6
9	Github Authentication and API access & gap analysis	2d	10/06/19	11/06/19	4
10	AWS Component review & gap analysis	5d	10/06/19	14/06/19	4
11	Identify missing components that need to be developed or integrated	5d	17/06/19	21/06/19	10
12	**Design Phase**	20d	24/06/19	19/07/19	
13	Design log submission and storage approach	10d	24/06/19	05/07/19	7
14	Design Pattern Storage and version control approach	10d	24/06/19	05/07/19	7
15	User Interface Design	10d	08/07/19	19/07/19	14
16	Development Phase	30d	08/07/19	16/08/19	
17	Develop UI prototype	10d	22/07/19	02/08/19	15
18	Develop log storage API	5d	08/07/19	12/07/19	13
19	Develop Pattern editing interface	10d	05/08/19	16/08/19	17, 18
20	**Implementation Phase**	68d	10/07/19	11/10/19	
21	Implement Jenkins/TravisCI/CircleCI for deployment & Testing tasks using prototypes	15d	10/07/19	30/07/19	
22	Implement log storage API	15d	19/08/19	06/09/19	21, 16
23	Implement Pattern Sharing API	15d	19/08/19	06/09/19	21, 16
24	Implement Pattern Development & Sharing interface	15d	09/09/19	27/09/19	23
25	User Documentation	10d	30/09/19	11/10/19	24
26	Administration & Deployment documentation	10d	31/07/19	13/08/19	21
27	**Testing & Evaluation Phase**	53d	31/07/19	11/10/19	
28	Implement Unit Testing environment	5d	31/07/19	06/08/19	21
29	Implement Integration Testing environment	5d	07/08/19	13/08/19	28
30	Finalize Use-Case and User Story issues	5d	30/09/19	04/10/19	24, 29
31	Correct Customer Acceptance Test refusals	5d	07/10/19	11/10/19	30, 34
32	**Imposed Milestones**	139d	21/05/19	01/12/19	
33	Progress Review	0	21/05/19	21/05/19	
34	Initial Customer Acceptance Tests	0	04/10/19	04/10/19	30
35	Final Customer Acceptance Tests	0	11/10/19	11/10/19	31
36	Final Delivery to customer (project close)	0	01/11/19	01/11/19	
37	Final Delivery to TeeSide	0	01/12/19	01/12/19	

Table 1: Proposed Project Timeline

Gantt Chart

	Task Name	Duration	Start	Finish
1	Requirements Phase	50d	01/04/19	07/06/19
2	Gather customer requirements	15d	01/04/19	19/04/19
3	Use Case Definitions	15d	22/04/19	10/05/19
4	Customer Acceptance Test definition	10d	13/05/19	24/05/19
5	*Customer Acceptance of Specification Document*	0	24/05/19	24/05/19
6	User Story Definiton (UI)	10d	27/05/19	07/06/19
7	Research Phase	10d	10/06/19	21/06/19
8	UI Framework technology review	5d	10/06/19	14/06/19
9	Github Authentication and API access & gap analysis	2d	10/06/19	11/06/19
10	AWS Component review & gap analysis	5d	10/06/19	14/06/19
11	Identify missing components that need to be developed or integrated	5d	17/06/19	21/06/19
12	Design Phase	20d	24/06/19	19/07/19
13	Design log submission and storage approach	10d	24/06/19	05/07/19
14	Design Pattern Storage and version control approach	10d	24/06/19	05/07/19
15	User Interface Design	10d	08/07/19	19/07/19
16	Development Phase	30d	08/07/19	16/08/19
17	Develop UI prototype	10d	22/07/19	02/08/19
18	Develop log storage API	5d	08/07/19	12/07/19
19	Develop Pattern editing interface	10d	05/08/19	16/08/19
20	Implementation Phase	68d	10/07/19	11/10/19
21	Implement Jenkins/TravisCI/CircleCI for deployment & Testing tasks using prototypes	15d	10/07/19	30/07/19
22	Implement log storage API	15d	19/08/19	06/09/19
23	Implement Pattern Sharing API	15d	19/08/19	06/09/19
24	Implement Pattern Development & Sharing interface	15d	09/09/19	27/09/19
25	User Documentation	10d	30/09/19	11/10/19
26	Administration & Deployment documentation	10d	31/07/19	13/08/19
27	Testing & Evaluation Phase	53d	31/07/19	11/10/19
28	Implement Unit Testing environment	5d	31/07/19	06/08/19
29	Implement Integration Testing environment	5d	07/08/19	13/08/19
30	Finalize Use-Case and User Story issues	5d	30/09/19	04/10/19
31	Correct Customer Acceptance Test refusals	5d	07/10/19	11/10/19
32	Imposed Milestones	139d	21/05/19	01/12/19
33	Progress Review	0	21/05/19	21/05/19
34	Initial Customer Acceptance Tests	0	04/10/19	04/10/19
35	Final Customer Acceptance Tests	0	11/10/19	11/10/19
36	Final Delivery to customer (project close)	0	01/11/19	01/11/19
37	Final Delivery to TeeSide	0	01/12/19	01/12/19



Exported on April 1, 2019 1:36:09 o'clock PM MEST

List of Tables

References

CMS (Mar. 2008). *Selecting a Development Approach*. URL: https://www.cms.gov/Research-Statistics-Data-and-Systems/CMS-Information-Technology/XLC/Downloads/SelectingDevelopmentApproach.pdf.

Elasticsearch (2019). *Filter Plugins — Logstash Versioned Plugin Reference*. URL: https://www.elastic.co/guide/en/logstash-versioned-plugins/current/filter-plugins.html.

IBM (2019). *IBM QRadar SIEM V7.3.2 documentation*. URL: https://www.ibm.com/support/knowledgecenter/en/SS42VS_DSM/c_logsource_manage.html?cp=SS42VS_7.3.2.

LizardLabs (Mar. 2019). *Log Parser Lizard*. URL: http://www.lizard-labs.com/log_parser_lizard.aspx.

MindTools (2018). *Estimating Time Accurately*. URL: https://www.mindtools.com/pages/article/newPPM_01.htm.

SANS (2019). *SANS Institute: Reading Room*. URL: https://www.sans.org/reading-room/.

Glossary

Amazon API Gateway A web-api hosting gateway that is part of Amazon's AWS offering. 4

`https://aws.amazon.com/api-gateway/`

Amazon AWS A cloud-based computing platform provided by Amazon Inc.. 1, 3–5

`https://aws.amazon.com`

Amazon Cognito A web-based identity provider that is part of Amazon's AWS offering. 4

`https://aws.amazon.com/cognito/`

Amazon Lambda A serverless function execution environment provided as part of Amazon's AWS offering. 4

`https://aws.amazon.com/lambda/`

AngularJS A web application technology provided by Google. 4

`https://angularjs.io/`

Cyberchef A Javascript-based visual text processing tool. 2

`https://gchq.github.io/CyberChef/`

ElasticSearch A popular open source log centralization technology product. 2

`https://www.elastic.co`

Github Web-based version control system based on Git. 3–5

`https://www.github.com`

IBM QRadar Security Incident and Event Manager product provided by IBM. 2

`https://www.ibm.com/security/security-intelligence/qradar`

Java Object oriented programming language and virtual machine platform supplied by Oracle Inc.. 3

`https://www.java.com`

Lizard Labs Software tool company with a Microsoft systems offering. 2

`https://www.lizard-labs.com`

Python Interpreted programming language. 3

`https://www.python.org`

ReactJS A web application technology provided by Facebook. 4

`https://reactjs.org/`

SANS Security training and cooperative research organisation. 1

`https://www.sans.org`

YouTube A video focused social networking site provided by Google. 5

`https://www.youtube.com/nuttybrewer/`

Acronyms

API Application Programming Interface. 4

CI/CD Continuous Integration/Continuous Delivery. 2, 4

ETL Extraction, Transformation and Loading. 1–3

SPA Single Page Application. 3, 4

SQL Structured Query Language. 3

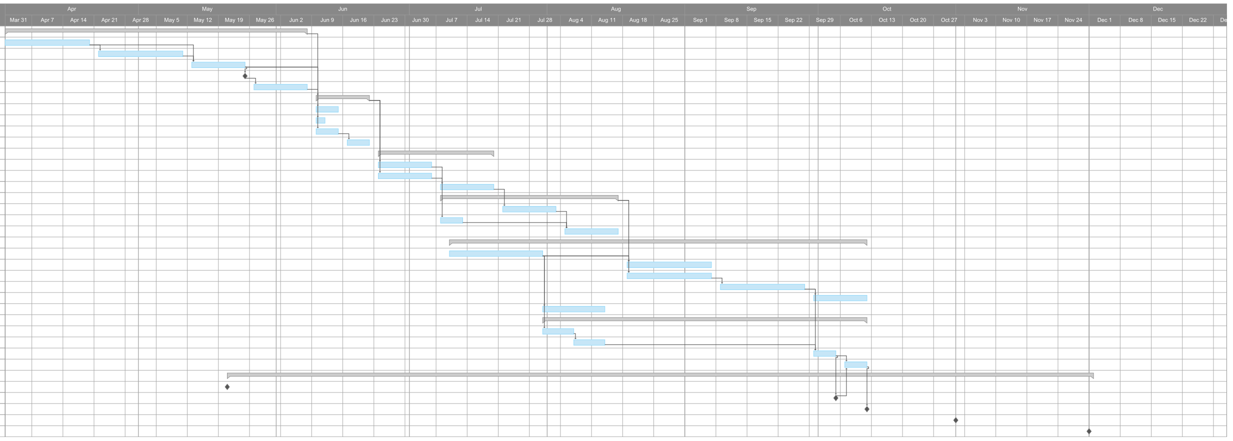
WBS Work Breakdown Structure. 5

A.1 Project Timeline

Description	Duration	Start	End	Predecessors	
1	**Requirements Phase**	50d	01/04/19	07/06/19	
2	Gather customer requirements	15d	01/04/19	19/04/19	
3	Use Case Definitions	15d	22/04/19	10/05/19	2
4	Customer Acceptance Test definition	10d	13/05/19	24/05/19	2, 3
5	Customer Acceptance of Specification Document	0	24/05/19	24/05/19	4
6	User Story Definiton (UI)	10d	27/05/19	07/06/19	5
7	**Research Phase**	10d	10/06/19	21/06/19	1
8	UI Framework technology review	5d	10/06/19	14/06/19	6
9	Github Authentication and API access & gap analysis	2d	10/06/19	11/06/19	4
10	AWS Component review & gap analysis	5d	10/06/19	14/06/19	4
11	Identify missing components that need to be developed or integrated	5d	17/06/19	21/06/19	10
12	**Design Phase**	20d	24/06/19	19/07/19	
13	Design log submission and storage approach	10d	24/06/19	05/07/19	7
14	Design Pattern Storage and version control approach	10d	24/06/19	05/07/19	7
15	User Interface Design	10d	08/07/19	19/07/19	14
16	Development Phase	30d	08/07/19	16/08/19	
17	Develop UI prototype	10d	22/07/19	02/08/19	15
18	Develop log storage API	5d	08/07/19	12/07/19	13
19	Develop Pattern editing interface	10d	05/08/19	16/08/19	17, 18
20	**Implementation Phase**	68d	10/07/19	11/10/19	
21	Implement Jenkins/TravisCI/CircleCI for deployment & Testing tasks using prototypes	15d	10/07/19	30/07/19	
22	Implement log storage API	15d	19/08/19	06/09/19	21, 16
23	Implement Pattern Sharing API	15d	19/08/19	06/09/19	21, 16
24	Implement Pattern Development & Sharing interface	15d	09/09/19	27/09/19	23
25	User Documentation	10d	30/09/19	11/10/19	24
26	Administration & Deployment documentation	10d	31/07/19	13/08/19	21
27	**Testing & Evaluation Phase**	53d	31/07/19	11/10/19	
28	Implement Unit Testing environment	5d	31/07/19	06/08/19	21
29	Implement Integration Testing environment	5d	07/08/19	13/08/19	28
30	Finalize Use-Case and User Story issues	5d	30/09/19	04/10/19	24, 29
31	Correct Customer Acceptance Test refusals	5d	07/10/19	11/10/19	30, 34
32	**Imposed Milestones**	139d	21/05/19	01/12/19	
33	Progress Review	0	21/05/19	21/05/19	
34	Initial Customer Acceptance Tests	0	04/10/19	04/10/19	30
35	Final Customer Acceptance Tests	0	11/10/19	11/10/19	31
36	Final Delivery to customer (project close)	0	01/11/19	01/11/19	
37	Final Delivery to TeeSide	0	01/12/19	01/12/19	

Table 8: Proposed Project Timeline

	Task Name	Duration	Start	Finish
1	**Requirements Phase**	50d	01/04/19	07/06/19
2	Gather customer requirements	15d	01/04/19	19/04/19
3	Use Case Definitions	15d	22/04/19	10/05/19
4	Customer Acceptance Test definition	10d	13/05/19	24/05/19
5	*Customer Acceptance of Specification Document*	0	24/05/19	24/05/19
6	User Story Definiton (UI)	10d	27/05/19	07/06/19
7	**Research Phase**	10d	10/06/19	21/06/19
8	UI Framework technology review	5d	10/06/19	14/06/19
9	Github Authentication and API access & gap analysis	2d	10/06/19	11/06/19
10	AWS Component review & gap analysis	5d	10/06/19	14/06/19
11	Identify missing components that need to be developed or integrated	5d	17/06/19	21/06/19
12	**Design Phase**	20d	24/06/19	19/07/19
13	Design log submission and storage approach	10d	24/06/19	05/07/19
14	Design Pattern Storage and version control approach	10d	24/06/19	05/07/19
15	User Interface Design	10d	08/07/19	19/07/19
16	**Development Phase**	30d	08/07/19	16/08/19
17	Develop UI prototype	10d	22/07/19	02/08/19
18	Develop log storage API	5d	08/07/19	12/07/19
19	Develop Pattern editing interface	10d	05/08/19	16/08/19
20	**Implementation Phase**	68d	10/07/19	11/10/19
21	Implement Jenkins/TravisCI/CircleCI for deployment & Testing tasks using prototypes	15d	10/07/19	30/07/19
22	Implement log storage API	15d	19/08/19	06/09/19
23	Implement Pattern Sharing API	15d	19/08/19	06/09/19
24	Implement Pattern Development & Sharing interface	15d	09/09/19	27/09/19
25	User Documentation	10d	30/09/19	11/10/19
26	Administration & Deployment documentation	10d	31/07/19	13/08/19
27	**Testing & Evaluation Phase**	53d	31/07/19	11/10/19
28	Implement Unit Testing environment	5d	31/07/19	06/08/19
29	Implement Integration Testing environment	5d	07/08/19	13/08/19
30	Finalize Use-Case and User Story issues	5d	30/09/19	04/10/19
31	Correct Customer Acceptance Test refusals	5d	07/10/19	11/10/19
32	**Imposed Milestones**	139d	21/05/19	01/12/19
33	Progress Review	0	21/05/19	21/05/19
34	Initial Customer Acceptance Tests	0	04/10/19	04/10/19
35	Final Customer Acceptance Tests	0	11/10/19	11/10/19
36	Final Delivery to customer (project close)	0	01/11/19	01/11/19
37	Final Delivery to TeeSide	0	01/12/19	01/12/19



Exported on April 1, 2019 1:36:09 o'clock PM MEST

B Project Code Repositories, Commit Logs and Issue Tracking

The repositories listed below contain all the source code that drives the components demonstrated in this project. Note that there are a few missing components that allow to deploy this within the customer's portal. These components can be made available upon request but live inside the customer's internal code repositories. Namely, all the functionality that allows for the registration of the domain name within DNS, the deployment of the initial Amazon Cloudfront site and the implementation of SSL certificates was all bootstrapped using the customer provided environment.

Customer Issues were also submitted to these repositories. All the commit logs and customers issues can be accessed in the links below. Should any of the repositories not be publicly available, please provide your GitHub nickname via email to *patrick.ethier@praguecollege.cz* and access will be granted.

B.1 Repositories

OAuth2 Function for Amazon Cloudfront
https://github.com/nuttybrewer/CloudFrontOAuthCognito

ReactJS Web Application to be deployed to Amazon S3 and Amazon Cloudfront
https://github.com/nuttybrewer/CloudFormationReactJSWebsite

Main Amazon CloudFormation deployment templates that creates the Amazon Cloudfront distribution components.
https://github.com/nuttybrewer/CloudFrontWebsiteDeployment

Amazon API Gateway and Amazon Lambda function to implement HCON testing API.
https://github.com/nuttybrewer/CloudFormationAWSAPIGW

B.2 Commit/Blog Entries

B.2.1 Overall project commit listing

Listing 9: GitHub commit logs for the overall project

```
commit 8a36528b0cdd541e62dfdab638e31339029baee0
Author: nuttybrewer <nuttybrewer@gmail.com>
Date:   Tue Nov 19 18:13:06 2019 +0100

    Next version of thesis

commit b795c6f94602b043ef9b03a192437c31186acadd
Author: nuttybrewer <nuttybrewer@gmail.com>
Date:   Thu Nov 14 14:00:24 2019 +0100

    Updated thesis doc with research elements

commit 1c7a9ac0246a4064bfb2794b43d2a4e42fdcf264
Author: nuttybrewer <nuttybrewer@gmail.com>
Date:   Mon Nov 4 11:23:38 2019 +0100

    Report introduction

commit 33f0372d493c2974c31997e241aa61550d23888b
Author: nuttybrewer <nuttybrewer@gmail.com>
Date:   Tue Oct 29 11:43:11 2019 +0100

    fix wordcount

commit 4ec403b24e79b656b08f170786ce953287d0eb99
Author: nuttybrewer <nuttybrewer@gmail.com>
Date:   Tue Oct 29 11:42:09 2019 +0100

    Adding test plan

commit 67c2f592805dbbd8e15ebb9500e85289eaad6c41
Author: nuttybrewer <nuttybrewer@gmail.com>
Date:   Mon Oct 7 12:36:29 2019 +0200

    Add use cases and package diagrams

commit cc901f68896ede4a102126689130c50e401f8ded
Author: nuttybrewer <nuttybrewer@gmail.com>
Date:   Mon Oct 7 12:35:42 2019 +0200

    Fix intro page and outline

commit c023320bd7fc6a21ed00c4cc0f95f72f476fad15
Author: nuttybrewer <nuttybrewer@gmail.com>
Date:   Mon Oct 7 12:35:31 2019 +0200

    Fix intro page and outline

```

```
commit 1dae5dd5a26898d0ff57325b94e6e48069114166
Author: nuttybrewer <nuttybrewer@gmail.com>
Date:   Thu Aug 29 11:26:02 2019 +0200

    Added link for third party HOCON parser

commit fe07e31ebb3be39beaf4d9437ccd71d068516e0a
Author: nuttybrewer <nuttybrewer@gmail.com>
Date:   Sat Aug 3 18:12:30 2019 +0200

    Add new component that will contain the Github OAUTH2 endpoint

commit 88451d307a1dacebd24013cbf0eb582deb41355e
Author: nuttybrewer <nuttybrewer@gmail.com>
Date:   Thu Aug 1 21:34:53 2019 +0200

    Add packages to show what's produced by each Repository

commit c2e012f2f8b7c037b5750e82bd6297816fddbbd2
Author: nuttybrewer <nuttybrewer@gmail.com>
Date:   Thu Aug 1 19:21:24 2019 +0200

    Update the repository components

commit e731e5bbdf60da4912f457c3cb96f7524253a8cf
Author: nuttybrewer <nuttybrewer@gmail.com>
Date:   Thu Aug 1 18:44:10 2019 +0200

    Moved this to it's own repository CloudFrontJWTValidator

commit 8dce9ddef0465d7128deef3155852f6693064bf5
Author: nuttybrewer <nuttybrewer@gmail.com>
Date:   Thu Aug 1 18:43:34 2019 +0200

    Links added as working on other repos

commit d1811e2a902920582e0de24f9409019b454d57b6
Author: nuttybrewer <nuttybrewer@gmail.com>
Date:   Mon Jul 22 12:54:07 2019 +0200

    Add JWTValidator project

commit 67966c20aa9943c54ea65c9bf10eb732d0ed370e
Author: nuttybrewer <nuttybrewer@gmail.com>
Date:   Mon Jul 22 12:50:31 2019 +0200

    Try the circleci multirepo approach proposed at https://github.com/Tufin/circleci-monorepo/blob/mas

commit 0ad2c59ddd6cd017ac067b6dcede7a903662d70d
Author: nuttybrewer <nuttybrewer@gmail.com>
Date:   Sun Jul 21 16:27:30 2019 +0200

    check all outputs

commit a5bd99a8d0d07e303ae937dd7ae57d94c4b56202
Author: nuttybrewer <nuttybrewer@gmail.com>
```

```
Date:   Sun Jul 21 16:25:32 2019 +0200

    See if this fails if I've got inverted logic

commit 281230a17fee7a3a2804d74b343bb45d5eeb4cd5
Author: nuttybrewer <nuttybrewer@gmail.com>
Date:   Sun Jul 21 16:24:02 2019 +0200

    put back when and add an ls command so it's a substantial test

commit ef173a1babe084ac0e668fae4944d0fecaf52190
Author: nuttybrewer <nuttybrewer@gmail.com>
Date:   Sun Jul 21 16:21:22 2019 +0200

    put back when and add an ls command so it's a substantial test

commit ae0ab672fed16fdd400c8718aa28eeda118014e8
Author: nuttybrewer <nuttybrewer@gmail.com>
Date:   Sun Jul 21 16:11:47 2019 +0200

    Force an error condition on the when

commit 42cafc5aa64fb2a751f5c87c23c6cc4c0bd12cf9
Author: nuttybrewer <nuttybrewer@gmail.com>
Date:   Sun Jul 21 16:07:31 2019 +0200

    introduce when condition

commit c1fef9d894bd64814488bc215b69b6c034d43f52
Author: nuttybrewer <nuttybrewer@gmail.com>
Date:   Sun Jul 21 16:03:11 2019 +0200

    Typo

commit 323ca71c785ea5efe4c0ed255787a7104046e131
Author: nuttybrewer <nuttybrewer@gmail.com>
Date:   Sun Jul 21 16:01:57 2019 +0200

    Try if the current path returns files, then run, if not, then skip

commit 3f79fb39fe74f3d39c3e53e62c5e3e04f1040d58
Author: nuttybrewer <nuttybrewer@gmail.com>
Date:   Sun Jul 21 15:52:59 2019 +0200

    Let's see what the output of the git diff-tree is when we run inside a sub-directory

commit 6fab07b849df77fa599e2c13390c05b39e9fdc3c
Author: nuttybrewer <nuttybrewer@gmail.com>
Date:   Sun Jul 21 15:52:16 2019 +0200

    Let's see what the output of the git diff-tree is when we run inside a sub-directory

commit dd694321393ddc3435a8feb6da3214527dc92bc2
Author: nuttybrewer <nuttybrewer@gmail.com>
Date:   Sun Jul 21 15:49:31 2019 +0200

```

```
    Let's see what the output of the git diff-tree is when we run inside a sub-directory

commit 385eddb8495ef6635ef579e5b0ed3acbc0d023f2
Author: nuttybrewer <nuttybrewer@gmail.com>
Date:   Sun Jul 21 15:44:19 2019 +0200

    Let's see what the output of the git diff-tree is when we run inside a sub-directory

commit c259272f453a18e0eff22a36a15955a9c91988b0
Author: nuttybrewer <nuttybrewer@gmail.com>
Date:   Sun Jul 21 15:43:34 2019 +0200

    Let's see what the output of the git diff-tree is when we run inside a sub-directory

commit 13b1b251c9d530c4876388fb3d8b385692e6075b
Author: nuttybrewer <nuttybrewer@gmail.com>
Date:   Sun Jul 21 15:36:12 2019 +0200

    Let's see what the output of the git diff-tree is when we run inside a sub-directory

commit 06fca068808261e99869528cb6a63f5b752e9ebf
Author: nuttybrewer <nuttybrewer@gmail.com>
Date:   Sun Jul 21 14:56:11 2019 +0200

    add the lambda job

commit ecf2a29a859482a62653565e4b50efb63f69f418
Author: nuttybrewer <nuttybrewer@gmail.com>
Date:   Sun Jul 21 14:54:02 2019 +0200

    First try at directory masking

commit 6dd2f9295a54d9d98eb09c40c069c2c550e0adf0
Author: nuttybrewer <nuttybrewer@gmail.com>
Date:   Sun Jul 21 14:28:51 2019 +0200

    Remove customizations

commit b59b3d32c2512c52f454386d2fdd3fe394d42d54
Author: nuttybrewer <nuttybrewer@gmail.com>
Date:   Sun Jul 21 14:26:43 2019 +0200

    Try forcing the python3 version into the container

commit e695cd145416f5d7c460afb7a3e2c1d79dd85643
Author: nuttybrewer <nuttybrewer@gmail.com>
Date:   Sun Jul 21 14:26:03 2019 +0200

    Try forcing the python3 version into the container

commit d4df7196bb8554714ffe453f5741593e9a038ff9
Author: nuttybrewer <nuttybrewer@gmail.com>
Date:   Sun Jul 21 14:17:43 2019 +0200

    add python3 into the aws-cli job

```

```
commit 776cf3304d3f56f7512455bd867d05cbf29556c8
Author: nuttybrewer <nuttybrewer@gmail.com>
Date:   Sun Jul 21 14:15:54 2019 +0200

    typo

commit 9928255deffc16105f140b32cd09fef495d6d2ec
Author: nuttybrewer <nuttybrewer@gmail.com>
Date:   Sun Jul 21 14:15:17 2019 +0200

    Oops, saved with example exec, removed it

commit 4961b3683af5cdf0431931ef906396978a3ad11d
Author: nuttybrewer <nuttybrewer@gmail.com>
Date:   Sun Jul 21 14:12:55 2019 +0200

    Try installing python3 into circleci before running orb

commit 6263567c1885a2876b5397ee649e2beb995dae6f
Author: nuttybrewer <nuttybrewer@gmail.com>
Date:   Sun Jul 21 13:45:52 2019 +0200

    Trying to get aws-cli working for this project, changed the orb version

commit 1ab876c68ed36f268c1a276c337124781949c4e5
Author: nuttybrewer <nuttybrewer@gmail.com>
Date:   Sun Jul 21 13:34:56 2019 +0200

    Trying to get aws-cli working for this project

commit 5e0ec4b577ddeef8bb55a20eb72fa91774cc38fd
Author: nuttybrewer <nuttybrewer@gmail.com>
Date:   Sun Jul 21 13:06:24 2019 +0200

    fix filenamegit add .circleci/config.yml !git add .circleci/config.yml !

commit c02c7f48521ef3e45dd7ea1695db1361b05407ef
Author: nuttybrewer <nuttybrewer@gmail.com>
Date:   Sun Jul 21 13:02:34 2019 +0200

    Give up, do the tutorial

commit 40ec883bbb783c2265ba064b64758d0f2eaf6c20
Author: nuttybrewer <nuttybrewer@gmail.com>
Date:   Sun Jul 21 12:57:26 2019 +0200

    remove the workflow to see if that is what is crashing

commit bf47b91b6af17a0f77dd6c9e6850d186d525e717
Author: nuttybrewer <nuttybrewer@gmail.com>
Date:   Sun Jul 21 12:53:31 2019 +0200

    Add a build job the proper circle way and then point a workflow at it.

commit fbf7767e92d3f0bd370734083ddd922d4fa42d9b
Author: nuttybrewer <nuttybrewer@gmail.com>
```

```
Date:   Sun Jul 21 12:50:39 2019 +0200

    Fix the docker image

commit 3a77ca237320eeba7eb57539440d0d4f689e7f1e
Author: nuttybrewer <nuttybrewer@gmail.com>
Date:   Sun Jul 21 12:45:33 2019 +0200

    Adding .DS_Store to be ignored

commit 950faac921dd367612b8efd8689cfe42b136e4c3
Author: nuttybrewer <nuttybrewer@gmail.com>
Date:   Sun Jul 21 12:43:51 2019 +0200

    Adding automated build via CircleCI

commit 3cd0cad71f794bd0fe8c6fd95dec7fbab2167e71
Author: nuttybrewer <nuttybrewer@gmail.com>
Date:   Sun Jul 21 12:40:36 2019 +0200

    Certification creation is either blocking by using the amazon resource or the inline Lambda functio

commit 435a35c149ca75ff1f242177e229120965a66b5b
Author: nuttybrewer <nuttybrewer@gmail.com>
Date:   Sun Jul 21 12:37:41 2019 +0200

    add links and notes from research

commit 9f90aaa0bf6d786d038cd3e235fb4e6175f01e45
Author: nuttybrewer <nuttybrewer@gmail.com>
Date:   Sun Jul 21 12:36:48 2019 +0200

    Simple Test Lambda to try a different approach based on CI deployment for LatEdgeJWTValidator

commit 0268e363580cc857813288bad0d2df15d7ec3124
Author: nuttybrewer <nuttybrewer@gmail.com>
Date:   Sun Jul 21 12:30:23 2019 +0200

    Adding a Readme, mainly to inventory and provide a link to documentation for each template

commit b3d400a2dfe6ec68576f2f782eaa5b40bd953e71
Author: nuttybrewer <nuttybrewer@gmail.com>
Date:   Sun Jul 21 12:29:40 2019 +0200

    Adding a template to create the S3 bucket that will contain all the lambda function zip files

commit 695bab331145176674448864a4039a8219cd22ff
Author: nuttybrewer <nuttybrewer@gmail.com>
Date:   Thu Jul 18 18:18:18 2019 +0200

    More research and architecture on the CD/CI front. Trying to use AWS Cloudformation to deploy the b

commit ff4df8ac88f328b4c540c9ea30ac8fb6328a8881
Author: nuttybrewer <nuttybrewer@gmail.com>
Date:   Mon Jul 15 21:02:18 2019 +0200

```

```
    Add modelling use cases from original requirements

commit ce2eda554cc9b75e6a3ccb476b8a9a63ea9946da
Author: nuttybrewer <nuttybrewer@gmail.com>
Date:   Mon Jun 3 17:15:19 2019 +0200

    Change labels

commit 91d90e2caf14225b2e60a27c10c7786b077227df
Author: Patrick Ethier <patrick@secureops.com>
Date:   Mon Jun 3 17:08:21 2019 +0200

    Add the basic authentication use case to retrieve the index.html page

commit 4d2a530b7ca74c3a1d235664f45bbf293f080bd6
Author: Patrick Ethier <nuttybrewer@gmail.com>
Date:   Fri Apr 12 10:19:19 2019 +0200

    Initial commit
```

B.2.2 Cloudfront commit listing

Listing 10: GitHub commit logs for Amazon Cloudfront component

```
commit b0f1f62f2ab795fda563ac3564240569784c7e45
Author: Patrick Ethier <patrick@secureops.com>
Date:   Fri Aug 30 13:55:25 2019 +0200

    Use new index.htmld etector function

commit 891f33b8ba5ef46896e74d282f47d9324287cc1d
Author: Patrick Ethier <patrick@secureops.com>
Date:   Tue Aug 20 17:20:17 2019 +0200

    Fix bucket policy to return 404 instead of 403

commit 911a6b62e35df48bfb80d78fefbccd1cbf1f8cc7
Author: Patrick Ethier <patrick@secureops.com>
Date:   Sat Aug 3 18:06:31 2019 +0200

    Split the deployment in two stacks, add a dependency between the two stacks, add the JTWValidator f

commit c4321b3c16af3e936321b2673f58a72bbb3108be
Author: Patrick Ethier <patrick@secureops.com>
Date:   Wed Jul 31 20:16:20 2019 +0200

    Unset the AWS environment variables to avoid credential leakage

commit f67f9c83c2baeef5371499dc1f48a6b9d585ec30
Author: Patrick Ethier <patrick@secureops.com>
Date:   Wed Jul 31 20:16:03 2019 +0200

    Remove the old cache pattern and add two new ones. ISSUE: Still need to add the lambda at edge func

commit b43f6df05ae944966e389cd9dd73f9ee0b1ea6f7
Author: Patrick Ethier <patrick@secureops.com>
Date:   Mon Jul 29 14:20:58 2019 +0200

    Fix last line that didn't remove temporary file

commit b196f58d38ca2c92db054934ef8894dcd67595ac
Author: Patrick Ethier <patrick@secureops.com>
Date:   Mon Jul 29 14:11:46 2019 +0200

    Initial instructions

commit 26ca443e860a1a158fe8e148ba94cd7f78daf533
Author: Patrick Ethier <patrick@secureops.com>
Date:   Mon Jul 29 13:59:12 2019 +0200

    NodeJS Apache 2.0 license

commit 65977aa733ed059a03925eb5513947c289f97a62
Author: Patrick Ethier <patrick@secureops.com>
Date:   Mon Jul 29 13:58:03 2019 +0200
```

```

    Unit tests for the Promise version of the Lambda function

commit 515294a3759da5d7910177f25d832203d0d62db9
Author: Patrick Ethier <patrick@secureops.com>
Date:   Mon Jul 29 13:57:17 2019 +0200

    Wrapper functions around cfn-response and timers to implement a 'stall' function

commit 70ac455f19e465b6be6931ac0f2335684a5ae2a7
Author: Patrick Ethier <patrick@secureops.com>
Date:   Mon Jul 29 13:56:45 2019 +0200

    Mocks required for the Promise version of the unit tests

commit 4d181c05f1862f537438697c63947b0eb1f0ce80
Author: Patrick Ethier <patrick@secureops.com>
Date:   Mon Jul 29 13:56:18 2019 +0200

    Promise version of the lambda function. Unit tests shows everything works, but Lambda refuses to ex

commit fb8238a1f2e8d3f76f6e36a87de6c38aad58f9f9
Author: Patrick Ethier <patrick@secureops.com>
Date:   Mon Jul 29 13:55:08 2019 +0200

    Shortcut shell script that will run CircleCI build locally instead of on their cloud services

commit 83a01dd69aeeec2e442144af79a14fb9a6634605
Author: Patrick Ethier <patrick@secureops.com>
Date:   Mon Jul 29 13:51:13 2019 +0200

    Remove localci.sh from .gitignore, add .deploy.env

commit a8d813f558e4e317fb93a856696c9a6f42b4173f
Author: Patrick Ethier <patrick@secureops.com>
Date:   Mon Jul 29 13:50:01 2019 +0200

    If CloudFormation deployment gets the lambda function is able to output it's first line, this will

commit b0c77465160d503aea8194ef9738b2c89a297108
Author: Patrick Ethier <patrick@secureops.com>
Date:   Mon Jul 29 13:48:45 2019 +0200

    File required by circleci to export coverage and test results properly

commit cf4672573157e65f1775d8850d55850fde348ef8
Author: Patrick Ethier <patrick@secureops.com>
Date:   Mon Jul 29 13:47:42 2019 +0200

    NPM package dependencies

commit 7b544bb7f75aa9f92841114c3a7a499e51f80be6
Author: Patrick Ethier <patrick@secureops.com>
Date:   Mon Jul 29 13:47:08 2019 +0200

    Lambda function that uses callbacks, this is the only working version in this code base at the mome
```

```

commit 52e1c07b96e915e510db24e17b78822b658eccd2
Author: Patrick Ethier <patrick@secureops.com>
Date:   Mon Jul 29 13:45:54 2019 +0200

    Unit tests for the main lambda function that uses callbacks

commit bf327ae43b77eaa3333c7a3ddda4ff3caa1fa041
Author: Patrick Ethier <patrick@secureops.com>
Date:   Mon Jul 29 13:45:10 2019 +0200

    Mock function of the cfn-response module. This is so we don't make actual calls to the Cloudformati

commit 3447cd0de6dd9d1c5b92319c40a80d15399bedb2
Author: Patrick Ethier <patrick@secureops.com>
Date:   Mon Jul 29 13:43:55 2019 +0200

    File that masks out what gets placed inside the Lambda function's .zip file

commit 19685a7ce38a23bce1af33f29e0028a176193ab8
Author: Patrick Ethier <patrick@secureops.com>
Date:   Mon Jul 29 13:42:57 2019 +0200

    Portion of the Cloudfront deployment that creates the distribution, the S3 bucket that holds the si

commit 5a18436defd922d33bbb806c492cb9f987ea5ea7
Author: Patrick Ethier <patrick@secureops.com>
Date:   Mon Jul 29 13:42:24 2019 +0200

    Portion of the Cloudfront deployment that creates the AWS ACM certificate

commit 1048faa36aa4e530a453e38e678d2c261f9676c0
Author: Patrick Ethier <patrick@secureops.com>
Date:   Mon Jul 29 13:41:33 2019 +0200

    Mask out unwanted files like .deploy.env and other temporary files

commit 1e6b42aa7f2b5f3b67a9a332ad417e0d3778bf13
Author: Patrick Ethier <patrick@secureops.com>
Date:   Mon Jul 29 13:40:47 2019 +0200

    CircleCI build

commit 29df65f610fbfdccab6dd57423403456b6aab066
Author: Patrick Ethier <nuttybrewer@gmail.com>
Date:   Tue Jul 23 18:10:41 2019 +0200

    Initial commit
```

B.2.3 API Gateway commit listing

Listing 11: GitHub commit logs for Amazon API Gateway component

```
commit 4f8d2bbfc0e65b9fc5a3f642d34f436381faaf3b
Author: Patrick Ethier <patrick@secureops.com>
Date:   Fri Aug 30 14:01:03 2019 +0200

    Switch around the apidomain and remove the deployment step

commit ccc8653c32f51372003b00981f9253fe51ca9f7a
Author: Patrick Ethier <patrick@secureops.com>
Date:   Fri Aug 30 13:59:32 2019 +0200

    Adding a distribution domain name so we don't have to use an ACM with APIGW but rather just mask it

commit 141ac5e1874c2bfcb11739ff1737464a8dcf1e5e
Author: Patrick Ethier <patrick@secureops.com>
Date:   Fri Aug 30 13:58:34 2019 +0200

    Move the deployment into the API.json file since updating it doesn't create a new deployment but ju

commit 7fdbf843d84db901847846f41822bedc9f21439f
Author: Patrick Ethier <patrick@secureops.com>
Date:   Sat Aug 3 17:59:18 2019 +0200

    Change environment variables to reflect .circleci/config.yaml

commit 6ac5ec184199a0bab3502dc55bb53793e487ea2b
Author: Patrick Ethier <patrick@secureops.com>
Date:   Sat Aug 3 17:58:56 2019 +0200

    Add .deploy.env to ignore

commit 810ee99be89eca04858f8361991cd9a7aa07e3b4
Author: Patrick Ethier <patrick@secureops.com>
Date:   Sat Aug 3 17:58:37 2019 +0200

    Creates the actual API

commit 3879047e7acc028e7c624f333845e45d84f1de8f
Author: Patrick Ethier <patrick@secureops.com>
Date:   Sat Aug 3 17:58:18 2019 +0200

    Defines the custom domain for the API

commit c92b1e89fa187adac7294dc569006fa75e330052
Author: Patrick Ethier <patrick@secureops.com>
Date:   Sat Aug 3 17:57:52 2019 +0200

    Define the S3 bucket for all the contents of the API

commit 69bb72359d6e6a892c8c1da4956ee91625bf4d6e
Author: Patrick Ethier <patrick@secureops.com>
Date:   Sat Aug 3 17:57:26 2019 +0200
```

```

    Certificate script for the API

commit 56dc541e55ff21337076566712efa969059a9c53
Author: Patrick Ethier <patrick@secureops.com>
Date:   Sat Aug 3 17:57:07 2019 +0200

    CircleCI build configuration

commit c722eb5e7528c73d540ce02eaa9e1a783cd95e23
Author: Patrick Ethier <nuttybrewer@gmail.com>
Date:   Thu Aug 1 17:54:31 2019 +0200

    Initial commit
```

B.2.4 OAuth commit listing

Listing 12: GitHub commit logs for OAuth2 component

```
commit 411bd7cf2917a94b931c91e0ce25dc2e7bc040b1
Author: Patrick Ethier <nuttybrewer@gmail.com>
Date:   Fri Oct 18 14:00:21 2019 +0200

    Fixes #1 use referer instead of URI when it's available.

commit 4d8572d0276b00bc700fd62e1002d44076609f27
Merge: 5c47b18 01c98ca
Author: Patrick Ethier <nuttybrewer@gmail.com>
Date:   Wed Oct 9 10:32:14 2019 +0200

    Merge branch 'master' of github.com:nuttybrewer/CloudFrontOAuthCognito

commit 5c47b18705aa6792a15aa8d2ce7ddf9d12a362c1
Author: Patrick Ethier <nuttybrewer@gmail.com>
Date:   Wed Oct 9 10:32:01 2019 +0200

    Updating package version to avoid NPM vulnerabilities

commit 01c98ca02fad9a0abd2a148e04b10e2b307404f7
Author: Patrick Ethier <nuttybrewer@gmail.com>
Date:   Tue Oct 8 10:53:02 2019 +0200

    update license with name and year

commit d4059e414dc48f74f8d26a3b32a31ec2efce539f
Author: Patrick Ethier <patrick@secureops.com>
Date:   Sun Aug 11 00:49:30 2019 +0200

    Added unit tests and refactored to handle different OAuth providers.

commit 76d1592fe13114029d86c9d8f106116ebe7d0e1a
Author: Patrick Ethier <nuttybrewer@gmail.com>
Date:   Tue Aug 6 22:36:54 2019 +0200

    Initial commit
```

B.2.5 ReactJS commit listing

Listing 13: GitHub commit logs for ReactJS component

```
commit e966778c5ffdcaf5d7db2bf1f8b56d12dad8e4a2
Author: Patrick Ethier <nuttybrewer@gmail.com>
Date:   Thu Oct 17 23:15:12 2019 +0200

    Adding the proper modals to add sections and vendors

commit 88454fc8781358c527eea62a7a59f58ef240bc4f
Author: Patrick Ethier <nuttybrewer@gmail.com>
Date:   Thu Oct 17 11:12:26 2019 +0200

    Add new vendor dialog

commit 0e9540a91dbd3d925522b6054a92a3251ce3f90c
Author: Patrick Ethier <nuttybrewer@gmail.com>
Date:   Sat Sep 21 18:29:42 2019 +0200

    Add buttongroup for tree operations (add/delete/activate/deactive) extractors and vendor files

commit b9a84bc777bdc8288d3b412af8595b8060804e73
Author: Patrick Ethier <nuttybrewer@gmail.com>
Date:   Sat Sep 21 18:28:33 2019 +0200

    Create an onBlur event on the editor with the anticipation that we will need this to update the tre

commit e8e4fb03f1130c03da36c5feec7d6f10a11eebcc
Author: Patrick Ethier <nuttybrewer@gmail.com>
Date:   Sat Sep 21 18:27:28 2019 +0200

    Remove old commented code no longer used

commit de8c626f615037e4325694d9e770935590212c71
Author: Patrick Ethier <nuttybrewer@gmail.com>
Date:   Sat Sep 21 18:26:01 2019 +0200

    Fix root source to reflect actual path

commit 07f37994589d6e214da22aadf507b3352b5bc7d5
Author: Patrick Ethier <nuttybrewer@gmail.com>
Date:   Sat Sep 21 18:23:13 2019 +0200

    Fix 2 pixel offset when a vendor file has no extractors in the tree

commit f5f0f560974f5462866abb9d4e03e46041d85283
Author: Patrick Ethier <nuttybrewer@gmail.com>
Date:   Sat Sep 21 18:22:44 2019 +0200

    Update version of react-beardtree

commit 51b08756b46fbd6e1344a1f8bd3539a377caf261
Author: Patrick Ethier <nuttybrewer@gmail.com>
Date:   Sat Sep 21 18:22:00 2019 +0200
```

```

    Add functions to enable and disable extractors to the ini file

commit 98247f3b259f37b22ea00731b680f184514e4577
Author: Patrick Ethier <nuttybrewer@gmail.com>
Date:   Tue Sep 17 13:29:42 2019 +0200

    mock out the monaco editor because jest doesn't like 'ECM' compiled javascript

commit 5958928b02a6e98e91187bba26ee6797e0ea50ba
Author: Patrick Ethier <nuttybrewer@gmail.com>
Date:   Tue Sep 17 13:28:48 2019 +0200

    Add/Remove sections, includes, commit multiple files, delete multiple files, fix root properties fi

commit f536ec1fa7100f0c22482b62a28590ab7244e4f4
Author: Patrick Ethier <nuttybrewer@gmail.com>
Date:   Tue Sep 17 13:28:09 2019 +0200

    add mock proxy and install typescript for jst

commit c71419ce6894b1bbff665c3ea7555a609e0189ee
Author: Patrick Ethier <nuttybrewer@gmail.com>
Date:   Thu Sep 12 14:33:39 2019 +0200

    Fix add vendor, add functionality to ini objects (add vendor, add include, addvalue, etc...)

commit abee2899b8234ae5193ca6538c0cd5914320b45d
Author: Patrick Ethier <nuttybrewer@gmail.com>
Date:   Wed Sep 11 17:12:45 2019 +0200

    Implement the add section function from the website, fix managing the linked list as there were err

commit ddb4475ef457329465de339c4cb450f2118fb97d
Author: Patrick Ethier <nuttybrewer@gmail.com>
Date:   Tue Sep 10 13:07:09 2019 +0200

    Fix inconsistent config display and selection of non-morphline sections

commit 9e534d0f555e07748539153dfeed746b075e3bf3
Author: Patrick Ethier <nuttybrewer@gmail.com>
Date:   Mon Sep 9 10:37:28 2019 +0200

    Swap old DeniTree with react-treebeard, install the remove section logic, change how the navigation

commit 280b603a93aabe51216474c6baf3b80f36fcafbc
Author: Patrick Ethier <nuttybrewer@gmail.com>
Date:   Wed Sep 4 10:01:41 2019 +0200

    Fixed the display of valid JSON

commit 83f130f620b8a69445d1faae214db9b20accf560
Author: Patrick Ethier <nuttybrewer@gmail.com>
Date:   Tue Sep 3 19:02:46 2019 +0200

    Adding the Testing panel and correcting a whole slew of UI glitches
```

```

commit 41ae78f416128da8bbb99a7003f6f63d099c3308
Author: Patrick Ethier <nuttybrewer@gmail.com>
Date:   Fri Aug 23 11:52:35 2019 +0200

    Fix Fork dialog, remove the landing page so it can go to customer's default website

commit 0b8b02c3b2767a34069f4665313e95bb9d97b9ed
Author: Patrick Ethier <nuttybrewer@gmail.com>
Date:   Tue Aug 20 13:52:45 2019 +0200

    Adding Commit functionality and fix some layout issues

commit 007103a35ef31fd2ca9262620c8f0e80ee769c28
Author: Patrick Ethier <nuttybrewer@gmail.com>
Date:   Sat Aug 17 19:19:42 2019 +0200

    Fix various UI glitches and refactor loading screens

commit 69802f16236b1794bcdede8d4c2b033d380a8a9c
Author: Patrick Ethier <nuttybrewer@gmail.com>
Date:   Fri Aug 16 00:17:44 2019 +0200

    Fix dialog to show errors

commit e8a19d050e41ac2f635fdb6dc57a545cef058d10
Author: Patrick Ethier <nuttybrewer@gmail.com>
Date:   Fri Aug 16 00:12:10 2019 +0200

    Introduce modal dialogs to make navigation easier and prepare for first customer progress report an

commit 597a82f9055e512de8e980d04a1b25f01c9d3723
Author: Patrick Ethier <nuttybrewer@gmail.com>
Date:   Fri Aug 16 00:11:17 2019 +0200

    Adding FieldExtraction navigation, tree and editors in read-only mode

commit 9320f5bd009b12fcb8fdd86a6c44f8b5515aa07d
Author: Patrick Ethier <nuttybrewer@gmail.com>
Date:   Fri Aug 16 00:10:09 2019 +0200

    Adding loading of includes via callback and associated variable substitution

commit 6f63cab21fc75000057b40995be2c395beb26db9
Author: Patrick Ethier <nuttybrewer@gmail.com>
Date:   Tue Aug 13 00:25:38 2019 +0200

    Finish the parser, add include hooks, handle variable interpolation, fix array mappings

commit cb91c9ecf30a633633f680a0ab8e255e8836d57a
Author: Patrick Ethier <nuttybrewer@gmail.com>
Date:   Mon Aug 12 01:20:33 2019 +0200

    Fix so that multiple keys are considered arrays and add an option to encode using multiple keys ins

commit 406b07e4a23412286bab60b8f19a8b76d583db19
```

```
Author: Patrick Ethier <nuttybrewer@gmail.com>
Date:   Mon Aug 12 00:27:06 2019 +0200

    Adding Monaco editor, and first attempt at bootstrapping the interface because navigation was becom

commit a5457ad36155dc5d48505b44fe3279921ce66f91
Author: Patrick Ethier <nuttybrewer@gmail.com>
Date:   Sun Aug 11 00:37:47 2019 +0200

    Adding first iteration of Github components as part of research into how to use the API

commit 9e45ab066a53100a9d0d14c46a61c47d16ee9d28
Author: Patrick Ethier <nuttybrewer@gmail.com>
Date:   Sun Aug 11 00:37:19 2019 +0200

    Adding high order component for cookie management

commit 8cf632dc5006e2a6376edfc4b8dacc7595e621d3
Author: Patrick Ethier <nuttybrewer@gmail.com>
Date:   Sun Aug 11 00:36:47 2019 +0200

    Adding high order cookie management

commit 7c4e31e23858927eaaa87840f7e262b77bb59c1e
Author: Patrick Ethier <nuttybrewer@gmail.com>
Date:   Sun Aug 11 00:35:49 2019 +0200

    Adding packages for mock proxy, nodemon, util, jsonwebtoken, etc.

commit d14cfd5c330770f58e708f943c7c6544ff1f4968
Author: Patrick Ethier <nuttybrewer@gmail.com>
Date:   Sun Aug 11 00:34:08 2019 +0200

    Adding a mock server to generate simulated OAuth tokens for the development website. Still need to

commit 5a2abe5294cee5f7978f9a5c232268fa1ff00e8b
Author: Patrick Ethier <nuttybrewer@gmail.com>
Date:   Wed Jul 31 20:17:40 2019 +0200

    Try and specify the proper MIME type for favicon, still doesn't work

commit 746a51ccb626a18765c792aac9b0bb02e3ac9f80
Author: Patrick Ethier <nuttybrewer@gmail.com>
Date:   Mon Jul 29 15:39:10 2019 +0200

    Remove .deploy.env from repo

commit 64c4696d8fb81318b3f1d83427d6f13d5a9ea0f5
Author: Patrick Ethier <nuttybrewer@gmail.com>
Date:   Mon Jul 29 15:36:23 2019 +0200

    add .deploy.env to gitignore

commit 12e6a89cddbf4fc111df43405014028b2ab23af1
Author: Patrick Ethier <nuttybrewer@gmail.com>
Date:   Mon Jul 29 15:35:06 2019 +0200
```

```

    add S3 sync

commit edbe69b9a4cf34f904fbe8ceb6a61565d8f08627
Author: Patrick Ethier <nuttybrewer@gmail.com>
Date:   Mon Jul 29 15:34:46 2019 +0200

    add variables to support S3 push

commit b1bdc3ec023fea1a3a0464986b419c6403210330
Author: Patrick Ethier <nuttybrewer@gmail.com>
Date:   Mon Jul 29 15:34:27 2019 +0200

    add cleanup function

commit 400ea2ff543c89b7e1d1a395d8835a8c57ab84ba
Author: Patrick Ethier <patrick@secureops.com>
Date:   Mon Jul 29 14:58:15 2019 +0200

    First version of CircleCI deployment to see if npm build works

commit 54f53928499484751f7108159968f10a384d9ffe
Author: Patrick Ethier <patrick@secureops.com>
Date:   Mon Jul 29 14:57:37 2019 +0200

    Initial package files provided by create-react-app tool

commit 4c08b90c584cc08fcdbee4265056b85cdf9d9127
Author: Patrick Ethier <patrick@secureops.com>
Date:   Mon Jul 29 14:57:16 2019 +0200

    Initial React icons and css provided by create-react-app too

commit 1d38d42c4524382b7a3a5c5e0567e149ad9e1dae
Author: Patrick Ethier <patrick@secureops.com>
Date:   Mon Jul 29 14:56:57 2019 +0200

    Initial React components provided by create-react-app tool

commit 2a2d80ed8154ad1d8410e4bd218a39a110165652
Author: Patrick Ethier <patrick@secureops.com>
Date:   Mon Jul 29 14:56:11 2019 +0200

    Default README.md provided by create-react-app tool

commit 9c51767f14f97025e11fb1ec76201c4ef76a432f
Author: Patrick Ethier <patrick@secureops.com>
Date:   Mon Jul 29 14:55:52 2019 +0200

    update .gitignore with defaults provided by create-react-app tool

commit e97dd79f2af9a389d2e991b1973d3c400b80e9f1
Author: Patrick Ethier <nuttybrewer@gmail.com>
Date:   Mon Jul 29 14:13:48 2019 +0200

    Initial commit
```

Listing 14: GitHub commit logs for path / index.hmtl default object component

```
commit 2276210e69928f934809bf5d483b062cec9a6fab
Author: Patrick Ethier <patrick@secureops.com>
Date:   Tue Oct 8 13:54:43 2019 +0200

    Don't need environment in this function

commit 6aba90ecadee5deadd1b37a5b8e8addbf5a68de1
Author: Patrick Ethier <patrick@secureops.com>
Date:   Tue Oct 8 13:54:15 2019 +0200

    Don't need environment in this function

commit f3635514373bb4c50fba97470cd01557c64b92a2
Author: Patrick Ethier <patrick@secureops.com>
Date:   Tue Oct 8 13:48:18 2019 +0200

    Fix unit tests to remove unnecessary code

commit 3d2bd93b0d3233b528d1ad4e07abbf0ec65eb7ce
Author: Patrick Ethier <patrick@secureops.com>
Date:   Tue Oct 8 13:47:36 2019 +0200

    Fix Licenses and unused test portions left over from copy/paste. Also correct package.json and remo

commit 6a729aee38c07f694e7aa8c7e03bf8908cacc7d1
Author: Patrick Ethier <patrick@secureops.com>
Date:   Tue Oct 8 12:36:04 2019 +0200

    Adding Apache License

commit 00ee52b125a95db51b5384bcfa0f6e0e07e76370
Author: Patrick Ethier <patrick@secureops.com>
Date:   Fri Aug 30 13:54:15 2019 +0200

    switch the function from a JWT validator to a path/directory detector for CF since it doesn't know

commit 71e7b56ce7dafa80d0947374c4b8f66a55ad02cb
Author: Patrick Ethier <patrick@secureops.com>
Date:   Thu Aug 22 17:29:08 2019 +0200

    Rename the function to SubSPAIndexMask

commit bcaaa8ebabfeb46ac5c208e28c127a1768ae8327
Author: Patrick Ethier <patrick@secureops.com>
Date:   Sat Aug 3 18:05:19 2019 +0200

    Make the stack output a value so other stacks can depend on it

commit bd0b6e3744fce11b22cf4a2bac7cc29f7aa7df07
Author: Patrick Ethier <patrick@secureops.com>
Date:   Wed Jul 31 20:13:53 2019 +0200

    Add new functionality that recognizes JWT tokens in param/headers/cookies. Also fix so that the def
```

```

commit ea075185e6346ed4833f0e2ca8bc1f38fe6fec16
Author: Patrick Ethier <patrick@secureops.com>
Date:   Wed Jul 31 20:12:12 2019 +0200

    Switch the node execution to 10.X and add edgelambda service to the S3 bucket permissions

commit d9af435e0b3b85315fd486c38ed5b68e40d5750b
Author: Patrick Ethier <patrick@secureops.com>
Date:   Wed Jul 31 20:11:12 2019 +0200

    Remove unnecessary mockup code

commit d41ca85d548f857127c45245e775dc2c2d460e51
Author: Patrick Ethier <patrick@secureops.com>
Date:   Wed Jul 31 20:10:41 2019 +0200

    Remove jwt-to-pem and move jest-junit to devel. Also fix test and cleanup functions to reflect inje

commit 84f066ade0867216b8f2db9f4d171211bee15e5b
Author: Patrick Ethier <patrick@secureops.com>
Date:   Wed Jul 31 20:09:11 2019 +0200

    Fix jest configuration to work properly

commit d0b17a970289d11f1fb6a8de9a014c7f47d519d0
Author: Patrick Ethier <patrick@secureops.com>
Date:   Wed Jul 31 20:08:16 2019 +0200

    MAask out mocks and templates from the distribution

commit baff2a8a261400c930e7f967521a244260a1319b
Author: Patrick Ethier <patrick@secureops.com>
Date:   Wed Jul 31 20:07:43 2019 +0200

    Mask out environment files and newly fixed junit files

commit 78cd537564a3555733e09e98547bfe22dc72db80
Author: Patrick Ethier <patrick@secureops.com>
Date:   Wed Jul 31 20:06:45 2019 +0200

    Fix CI deployment build to use new generatejwt instead of CURL tool

commit f2e5c2ee70a7f965bc3cd2f0ae62bbb70900afff
Author: Patrick Ethier <patrick@secureops.com>
Date:   Wed Jul 31 20:06:13 2019 +0200

    Utility code to download JWT keys from the web before deployment and convert them to PEM format

commit 3389868c2341a0ba3999851af67d1c40c4671e5a
Author: Patrick Ethier <patrick@secureops.com>
Date:   Wed Jul 31 20:05:36 2019 +0200

    JWT validation function

commit d3b1ae56fc68d22cec0c2b2120a4931b897a709f
```

```
Author: Patrick Ethier <patrick@secureops.com>
Date:   Wed Jul 31 20:05:18 2019 +0200

    Unit test for the JWT validation functions

commit bb3d3a3e7f3a98d279af16533b0574a18007b91b
Author: Patrick Ethier <patrick@secureops.com>
Date:   Wed Jul 31 20:04:46 2019 +0200

    We want to inject the configuration at test/build time because LambdaAtEdge doesn't support setting

commit 596d9450d2c0b63731caeca119724beba2830bfa
Author: Patrick Ethier <patrick@secureops.com>
Date:   Wed Jul 31 20:03:37 2019 +0200

    Mock files to stop unit tests from reaching out to Cognito

commit cbe30ee4085060bb5bc0266b62d812cd6f0fc00e
Author: Patrick Ethier <patrick@secureops.com>
Date:   Wed Jul 31 20:02:57 2019 +0200

    localci.sh file to automatically build, test and deploy to Amazon

commit 66dc6712c6fe6d5ce025ad57d6c9706807728a6b
Author: Patrick Ethier <patrick@secureops.com>
Date:   Tue Jul 23 17:08:03 2019 +0200

    Try command line expansion instead of assignment

commit 985986a11a4c36d9f35eb9113ccaa89444eea026
Author: Patrick Ethier <patrick@secureops.com>
Date:   Tue Jul 23 17:02:46 2019 +0200

    Extra quote removed

commit 447cf5f78a6eba16607e1dda70c355fdfba406bd
Author: Patrick Ethier <patrick@secureops.com>
Date:   Tue Jul 23 16:59:34 2019 +0200

    missing a dot

commit 99c2b54d7dbe057b7bcbe9fb5dd84df31ea35c0e
Author: Patrick Ethier <patrick@secureops.com>
Date:   Tue Jul 23 16:42:37 2019 +0200

    typo

commit 3ef9a848cb6ba22b50f349ab19a2ea68c0556eb8
Author: Patrick Ethier <patrick@secureops.com>
Date:   Tue Jul 23 16:37:56 2019 +0200

    Specify the zipfile version

commit 383ec6c26485e2e65b18132e26fc2e5799be1484
Author: Patrick Ethier <patrick@secureops.com>
Date:   Tue Jul 23 14:56:50 2019 +0200
```

```

    Include JWT code to make sure the packaging works

commit ec95e7bee58e7b9ece9e7b96063759f90b1f2f9c
Author: Patrick Ethier <patrick@secureops.com>
Date:   Tue Jul 23 14:38:29 2019 +0200

    Add comment to tests so we can see how the versioning work upon deployment

commit d3a052c38d42a92f5d463c0ac641c542e4a05871
Author: Patrick Ethier <patrick@secureops.com>
Date:   Tue Jul 23 14:33:18 2019 +0200

    typo

commit 45b183b94977a66f0b1cddde1a0d7cc0dfafbb52
Author: Patrick Ethier <patrick@secureops.com>
Date:   Tue Jul 23 14:23:18 2019 +0200

    Add a version and an alias to the deployment

commit f8dab88acab75573f6f062014c2683d29faa4bed
Author: Patrick Ethier <patrick@secureops.com>
Date:   Tue Jul 23 12:17:39 2019 +0200

    Slash rocks

commit 504bb5e300f4a36cb86d80f60a85664278ddeca1
Author: Patrick Ethier <patrick@secureops.com>
Date:   Tue Jul 23 12:13:20 2019 +0200

    Parameters aren't the same as 'create-stack' directive. see if the docs are correct

commit ae071cc6d38dd20e82e73a60f559c48271e50a1a
Author: Patrick Ethier <patrick@secureops.com>
Date:   Tue Jul 23 12:02:52 2019 +0200

    adding comments to README for new file layout

commit a81626618cd5ffc6c5f52477a98f6b0a70592480
Author: Patrick Ethier <patrick@secureops.com>
Date:   Tue Jul 23 11:05:34 2019 +0200

    Add a checkout to the aws container

commit 8f16e420103ac2416ccbadd92bfd467925f1e02a
Author: Patrick Ethier <patrick@secureops.com>
Date:   Tue Jul 23 11:02:32 2019 +0200

    Typos

commit 56dcb553f893e35bea868a21ad4a3767d4775622
Author: Patrick Ethier <patrick@secureops.com>
Date:   Tue Jul 23 10:56:12 2019 +0200

    Fix path to cf template
```

```

commit 8b218c9f8f1c4ba371fdc5ad28e1ee60297eeacb
Author: Patrick Ethier <patrick@secureops.com>
Date:   Tue Jul 23 10:53:32 2019 +0200

    Try creating the Lambda Function

commit 6723331b385764d4055ffb41ab8da9d1170d3489
Author: Patrick Ethier <patrick@secureops.com>
Date:   Tue Jul 23 10:38:18 2019 +0200

    fix command completion path for cat | jq

commit 803810861a75013c759a17252e144e31f0f205f8
Author: Patrick Ethier <patrick@secureops.com>
Date:   Tue Jul 23 10:29:57 2019 +0200

    Moving tests directory

commit d0216ab4660da148074e7eab962f1c7e3a5a9be4
Author: Patrick Ethier <patrick@secureops.com>
Date:   Tue Jul 23 10:28:02 2019 +0200

    Fix cache package.json location

commit f58cac13c3acf8d6107bcfb5eb2b2ea9c3854a10
Author: Patrick Ethier <patrick@secureops.com>
Date:   Tue Jul 23 10:25:21 2019 +0200

    typo

commit fc82401bf288d8106d9bc6d4e4ca0c43aaf2d147
Author: Patrick Ethier <patrick@secureops.com>
Date:   Tue Jul 23 10:23:46 2019 +0200

    cd didn't work, add it to all the commands instead and correct the paths

commit 3216494a5dc339227506d35ef6eda744608a0e35
Author: Patrick Ethier <patrick@secureops.com>
Date:   Tue Jul 23 10:15:25 2019 +0200

    Move node files to their own directory and add CloudFormation template for the resource

commit 86ac1336383924d0d5158e585af07844afde5f78
Author: Patrick Ethier <patrick@secureops.com>
Date:   Mon Jul 22 22:11:24 2019 +0200

    Revert to using a known filename and modify environment variables not to clash with CIRCLE_ namespa

commit 783eda878a8f80f714392631d609540456fb4b49
Author: Patrick Ethier <patrick@secureops.com>
Date:   Mon Jul 22 21:47:41 2019 +0200

    add aws context

commit 147b2eb9c694b621c4f34c142554644a769caea4
```

```
Author: Patrick Ethier <patrick@secureops.com>
Date:   Mon Jul 22 21:43:46 2019 +0200

    Create a build directory and attempt to use that

commit 1097df7ca7dc1cbbd2b62ac70902c8c6c01321b4
Author: Patrick Ethier <patrick@secureops.com>
Date:   Mon Jul 22 21:37:42 2019 +0200

    Emulate workspace example

commit 317b4f4cbf9e2ddff51393f3e786eeb6149d25a8
Author: Patrick Ethier <patrick@secureops.com>
Date:   Mon Jul 22 21:26:41 2019 +0200

    it doesn't like the wildcard, try escaping it with quotes

commit 0e3facd317b9647840d6a3b31dcf8fca88eb4a2c
Author: Patrick Ethier <patrick@secureops.com>
Date:   Mon Jul 22 21:18:19 2019 +0200

    Add portion that uploads to S3

commit 7bf3e5b107bd2fec19d109453f93864b1f8b2f51
Author: Patrick Ethier <patrick@secureops.com>
Date:   Mon Jul 22 19:58:45 2019 +0200

    Check if the bucket exists

commit c48c54bacef9a27cb248390908e7abd102d0e8ed
Author: Patrick Ethier <patrick@secureops.com>
Date:   Mon Jul 22 19:48:47 2019 +0200

    typo and add *.zip to files to clean up

commit b8189e473ce7539b8b8483c22c9d1085ada24c1c
Author: Patrick Ethier <patrick@secureops.com>
Date:   Mon Jul 22 19:46:21 2019 +0200

    Add a step to zip the contents and then copy it to a file containing the branchname

commit 9b9a3962bd3f49949097b2450dbc93ec2378a2a9
Author: Patrick Ethier <patrick@secureops.com>
Date:   Mon Jul 22 15:28:02 2019 +0200

    Upload initial unit testing to see if it results in coverage

commit 35d70ad815aa1e05fd47a7b15ab69c4383a8452f
Author: Patrick Ethier <patrick@secureops.com>
Date:   Mon Jul 22 13:29:33 2019 +0200

    fix typo

commit 818c24ab25b834e9df31f7a2b167dff78b31e22e
Author: Patrick Ethier <patrick@secureops.com>
Date:   Mon Jul 22 13:26:49 2019 +0200
```

```

    Add node container

commit 9823c48ffda3f757c7da81515ec7a8ea679d7677
Author: Patrick Ethier <patrick@secureops.com>
Date:   Mon Jul 22 13:13:26 2019 +0200

    initial commit

commit 4fb03b9aa827ec709a687b2aead167a6de92ce11
Author: Patrick Ethier <nuttybrewer@gmail.com>
Date:   Mon Jul 22 12:59:49 2019 +0200

    Initial commit
```

C Test Plan

page left blank

User Guide & Test Plan

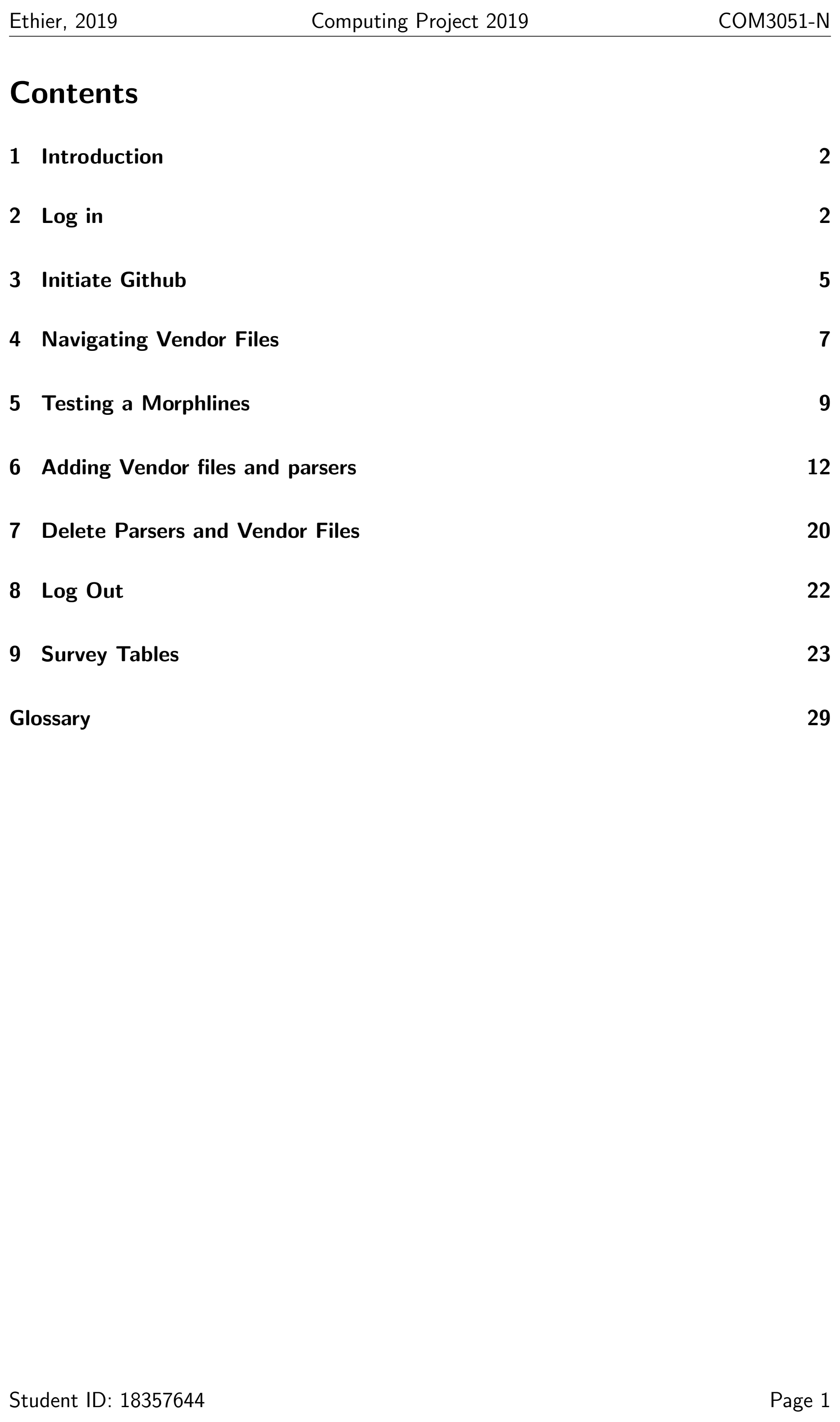
Patrick Ethier

October 29, 2019

Abstract

This document presents the user walkthrough and end-user test cases for the *Field Extraction Parser Workbench* web application.

Place: Prague College
Student ID: 18357644
Module Code: COM3051-N
Project Supervisor:Julian Warren
Semester Code: 1904
Word Count: 2076



Contents

1 Introduction

This document presents a walkthrough of the *Field Extraction Parser Workbench* web application. Each section below is cumulative. Should end-users experience bugs, they can report them at the following link using the Github account created for the purpose of using the web application.
https://github.com/nuttybrewer/CloudFormationReactJSWebsite/issues
Users are also encouraged to copy/paste the table in section 9 and return it via email to the author as feedback along with overall comments. This application will only be available in for testing at https://bob2.redlabnet.com until the 5th of December, 2019 at which point it will be taken down. Users are responsible for managing their Github account independently of this testing process past this date.

2 Log in

2.1 Open a browser to https://bob2.redlabnet.com
Click on the *login* button.

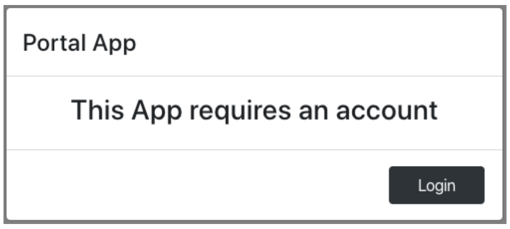
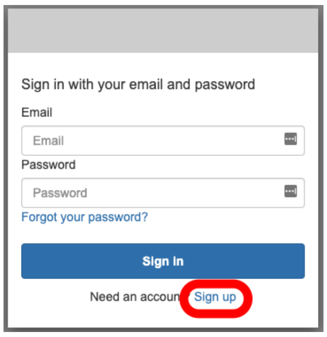


Figure 1: Landing page

2.2 If you do not have an account, click on *Sign up* link



Sign in with your email and password

Email

Email

Password

Password

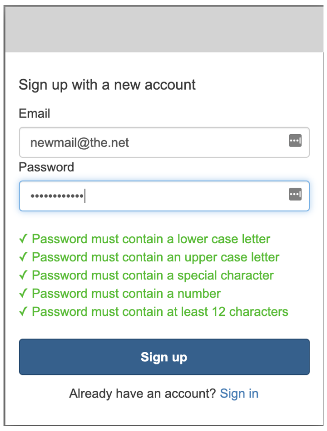
Forgot your password?

Sign in

Need an accoun Sign up

Figure 2: Login Screen

2.3 Enter the information. Note this must be a valid email address as you will be emailed a verification code in the next step



Sign up with a new account

Email

newmail@the.net

Password

••••••••••••

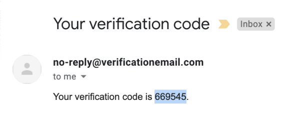
✓ Password must contain a lower case letter
✓ Password must contain an upper case letter
✓ Password must contain a special character
✓ Password must contain a number
✓ Password must contain at least 12 characters

Sign up

Already have an account? Sign in

Figure 3: Cognito sign-up page

2.4 Look in your email, you should receive and email with a verification code within a few minutes.



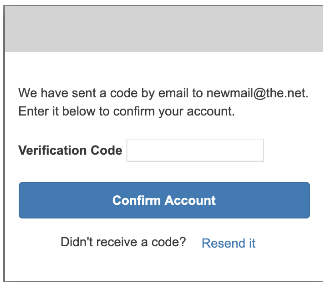
Your verification code Inbox ×

no-reply@verificationemail.com
to me

Your verification code is 669545.

Figure 4: Confirmation email

2.5 Enter this code in the *Verification code* text input

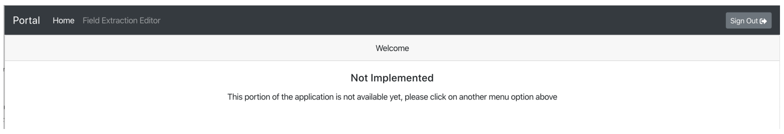


We have sent a code by email to newmail@the.net.
Enter it below to confirm your account.

Verification Code

Figure 5: Verification prompt

2.6 This will bring you to the initial web portal site



Not Implemented

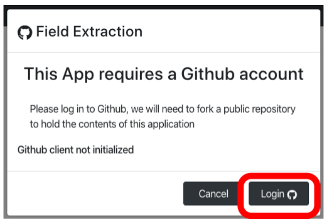
This portion of the application is not available yet, please click on another menu option above

Figure 6: Portal landing page

3 Initiate Github

In order to execute this function, you must be at the last step of 2 and have an existing Github account. It is suggested to create and log into Github via https://github.com prior to performing this step.

3.1 Click on *Field Extraction Editor* navigation item at the top of the screen
Click on the *login* button in the Github modal.



Field Extraction

This App requires a Github account

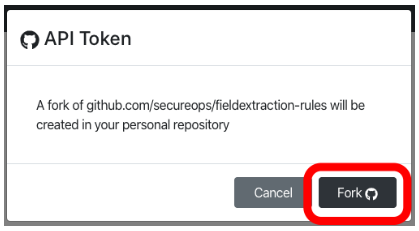
Please log in to Github, we will need to fork a public repository to hold the contents of this application

Github client not initialized

Figure 7: Github login dialog

Log into your github account. If you do not have an account, go through the sign-up process

3.2 A personal copy of the fieldextraction-rules repository will be created. Click on the *Fork* Button to do this.



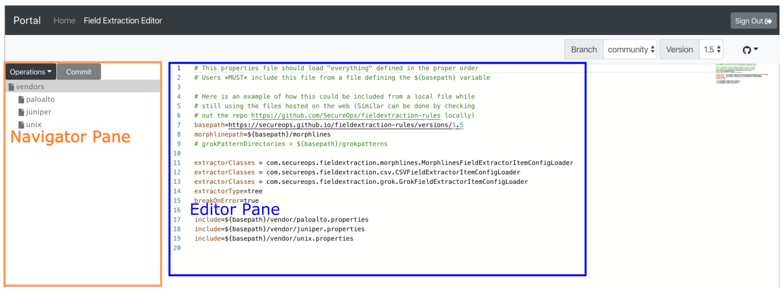
API Token

A fork of github.com/secureops/fieldextraction-rules will be created in your personal repository

Figure 8: Github fork dialog

3.3 Validate, using another web browser window or tab, that the fork succeeded by going to https://github.com/<**your-github-user**>/*fieldextraction-rules*.

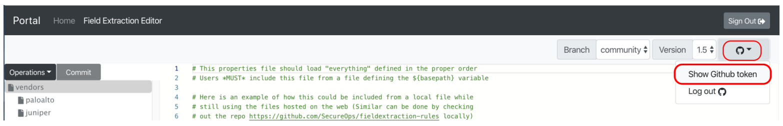
3.4 The default *Field Extraction Editor* environment should also be available on your browser and be similar as represented in figure 9



```
# This properties file should load "everything" defined in the proper order
# Users *MUST* include this file from a file defining the ${basepath} variable

# Here is an example of how this could be included from a local file while
# still using the files hosted on the web (Similar can be done by checking
# out the repo https://github.com/SecureOps/fieldextraction-rules locally)
basepath=https://secureops.github.io/fieldextraction-rules/versions/1.5
morphlinepath=${basepath}/morphlines
# grokPatternDirectories = ${basepath}/grokpatterns

extractorClasses = com.secureops.fieldextraction.morphlines.MorphlinesFieldExtractorItemConfigLoader
extractorClasses = com.secureops.fieldextraction.csv.CSVFieldExtractorItemConfigLoader
extractorClasses = com.secureops.fieldextraction.grok.GrokFieldExtractorItemConfigLoader
extractorType=tree
breakOnError=true

include=${basepath}/vendor/paloalto.properties
include=${basepath}/vendor/juniper.properties
include=${basepath}/vendor/unix.properties

```

Figure 9: Initial Field Extraction Editor

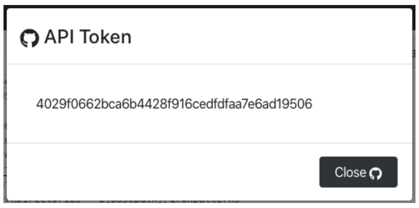
3.5 Click on the *Github* icon at the top left corner and select **Show Github token** menu item.



```
# This properties file should load "everything" defined in the proper order
# Users *MUST* include this file from a file defining the ${basepath} variable

# Here is an example of how this could be included from a local file while
# still using the files hosted on the web (Similar can be done by checking
# out the repo https://github.com/SecureOps/fieldextraction-rules locally)
```

Figure 10: Show Github token menu

3.6 Verify that the following dialog (with a different token value) appears and then press the *Close* button.

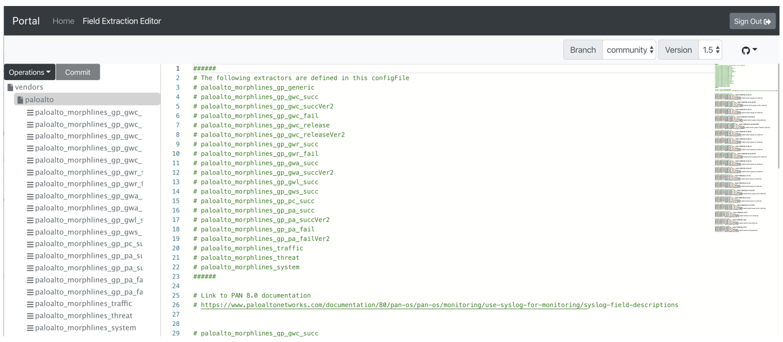


4029f0662bca6b4428f916cedfdfaa7e6ad19506

Figure 11: Show Github token dialog

4 Navigating Vendor Files

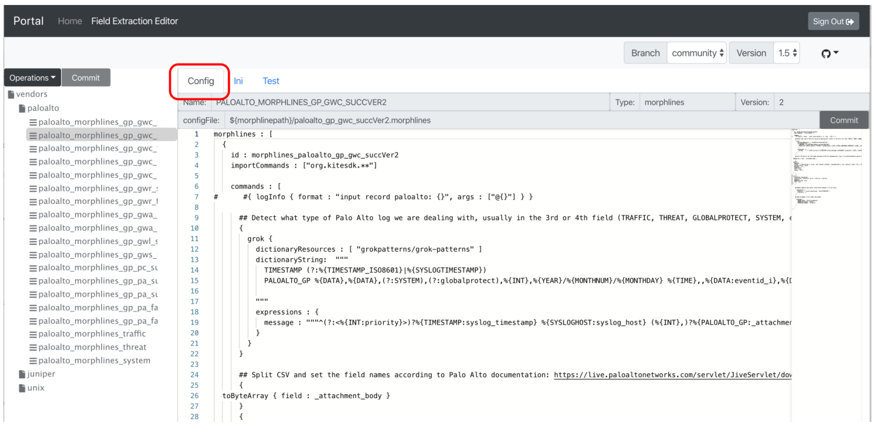
4.1 Select the *paloalto* vendor in the tree. Each *vendor* has parsers represented by a *Java* **.properties** file that will change to show all the parsers that are defined for this vendor.


Portal
Home
Field Extraction Editor
Sign Out
Branch
community
Version
1.5
Operations
Commit
vendors
paloalto
paloalto_morphlines_gp_gwc_
paloalto_morphlines_gp_gwc_
paloalto_morphlines_gp_gwc_
paloalto_morphlines_gp_gwc_
paloalto_morphlines_gp_gwc_
paloalto_morphlines_gp_gwr_
paloalto_morphlines_gp_gwr_
paloalto_morphlines_gp_gwa_
paloalto_morphlines_gp_gwa_
paloalto_morphlines_gp_gwl_
paloalto_morphlines_gp_gws_
paloalto_morphlines_gp_pc_su
paloalto_morphlines_gp_pa_su
paloalto_morphlines_gp_pa_su
paloalto_morphlines_gp_pa_fa
paloalto_morphlines_gp_pa_fa
paloalto_morphlines_traffic
paloalto_morphlines_threat
paloalto_morphlines_system

```
######
# The following extractors are defined in this configFile
# paloalto_morphlines_gp_generic
# paloalto_morphlines_gp_gwc_succ
# paloalto_morphlines_gp_gwc_succVer2
# paloalto_morphlines_gp_gwc_fail
# paloalto_morphlines_gp_gwc_release
# paloalto_morphlines_gp_gwc_releaseVer2
# paloalto_morphlines_gp_gwr_succ
# paloalto_morphlines_gp_gwr_fail
# paloalto_morphlines_gp_gwa_succ
# paloalto_morphlines_gp_gwa_succVer2
# paloalto_morphlines_gp_gwl_succ
# paloalto_morphlines_gp_gws_succ
# paloalto_morphlines_gp_pc_succ
# paloalto_morphlines_gp_pa_succ
# paloalto_morphlines_gp_pa_succVer2
# paloalto_morphlines_gp_pa_fail
# paloalto_morphlines_gp_pa_failVer2
# paloalto_morphlines_traffic
# paloalto_morphlines_threat
# paloalto_morphlines_system
######

# Link to PAN 8.0 documentation
# https://www.paloaltonetworks.com/documentation/80/pan-os/pan-os/monitoring/use-syslog-for-monitoring/syslog-field-descriptions


# paloalto_morphlines_gp_gwc_succ
```

Figure 12: Viewing a vendor file

4.2 Select one of the parsers. The editor pane should switch to the parser definition. In the case of the *paloalto* definitions, these are defined using a **.morphlines** file which should be displayed in the *Config* tab.


Portal
Home
Field Extraction Editor
Sign Out
Branch
community
Version
1.5
Operations
Commit
Config
Ini
Test
Commit
vendors
paloalto
paloalto_morphlines_gp_gwc_
paloalto_morphlines_gp_gwc_
paloalto_morphlines_gp_gwc_
paloalto_morphlines_gp_gwc_
paloalto_morphlines_gp_gwc_
paloalto_morphlines_gp_gwr_
paloalto_morphlines_gp_gwr_
paloalto_morphlines_gp_gwa_
paloalto_morphlines_gp_gwa_
paloalto_morphlines_gp_gwl_
paloalto_morphlines_gp_gws_
paloalto_morphlines_gp_pc_su
paloalto_morphlines_gp_pa_su
paloalto_morphlines_gp_pa_su
paloalto_morphlines_gp_pa_fa
paloalto_morphlines_gp_pa_fa
paloalto_morphlines_traffic
paloalto_morphlines_threat
paloalto_morphlines_system
juniper
unix

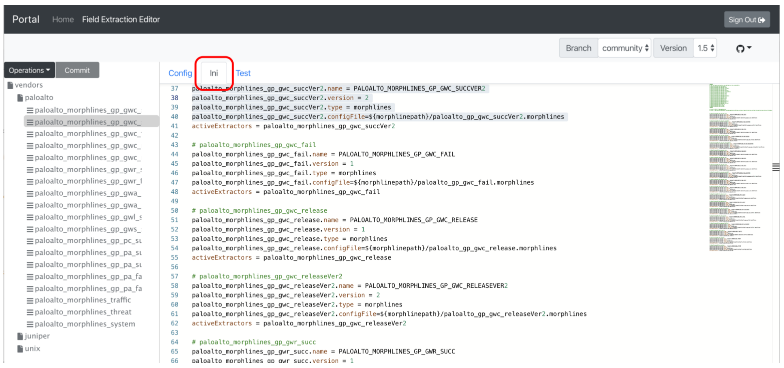
Name: PALOALTO_MORPHLINES_GP_GWC_SUCCVER2 Type: morphlines Version: 2

configFile: ${morphlinepath}/paloalto_gp_gwc_succVer2.morphlines

```
morphlines : [
  {
    id : morphlines_paloalto_gp_gwc_succVer2
    importCommands : ["org.kitesdk.**"]

    commands : [
#      #{ logInfo { format : "input record paloalto: {}", args : ["@{}"] } }

      ## Detect what type of Palo Alto log we are dealing with, usually in the 3rd or 4th field (TRAFFIC, THREAT, GLOBALPROTECT, SYSTEM, e
      {
        grok {
          dictionaryResources : [ "grokpatterns/grok-patterns" ]
          dictionaryString:  """
            TIMESTAMP (?:%{TIMESTAMP_ISO8601}|%{SYSLOGTIMESTAMP})
            PALOALTO_GP %{DATA},%{DATA},(?:SYSTEM),(?:globalprotect),%{INT},%{YEAR}/%{MONTHNUM}/%{MONTHDAY} %{TIME},,%{DATA:eventid_i},%{D

          """
          expressions : {
            message : """^(?:<%{INT:priority}>)?%{TIMESTAMP:syslog_timestamp} %{SYSLOGHOST:syslog_host} (%{INT},)?%{PALOALTO_GP:_attachmen
          }
        }
      }

      ## Split CSV and set the field names according to Palo Alto documentation: https://live.paloaltonetworks.com/servlet/JiveServlet/dow
      {
  toByteArray { field : _attachment_body }
      }
      {
```

Figure 13: A typical Morphlines file

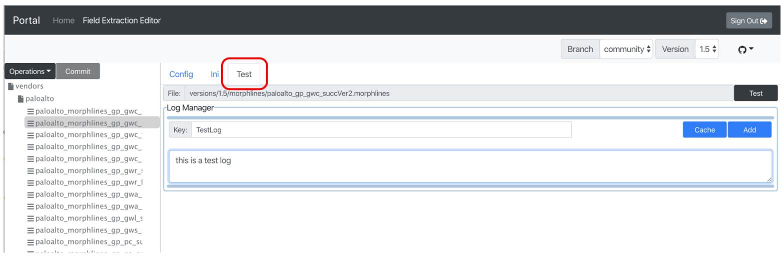
4.3 Select the *Ini* tab. The previous *Java* **.properties** should appear with the relevant section highlighted.


Portal
Home
Field Extraction Editor
Sign Out
Branch
community
Version
1.5
Operations
Commit
Config
Ini
Test
vendors
paloalto
paloalto_morphlines_gp_gwc_
paloalto_morphlines_gp_gwr_
paloalto_morphlines_gp_gwa_
paloalto_morphlines_gp_gwl_
paloalto_morphlines_gp_gws_
paloalto_morphlines_gp_pc_su
paloalto_morphlines_gp_pa_su
paloalto_morphlines_gp_pa_fa
paloalto_morphlines_traffic
paloalto_morphlines_threat
paloalto_morphlines_system
juniper
unix

```
paloalto_morphlines_gp_gwc_succVer2.name = PALOALTO_MORPHLINES_GP_GWC_SUCCVER2
paloalto_morphlines_gp_gwc_succVer2.version = 2
paloalto_morphlines_gp_gwc_succVer2.type = morphlines
paloalto_morphlines_gp_gwc_succVer2.configFile=${morphlinepath}/paloalto_gp_gwc_succVer2.morphlines
activeExtractors = paloalto_morphlines_gp_gwc_succVer2

# paloalto_morphlines_gp_gwc_fail
paloalto_morphlines_gp_gwc_fail.name = PALOALTO_MORPHLINES_GP_GWC_FAIL
paloalto_morphlines_gp_gwc_fail.version = 1
paloalto_morphlines_gp_gwc_fail.type = morphlines
paloalto_morphlines_gp_gwc_fail.configFile=${morphlinepath}/paloalto_gp_gwc_fail.morphlines
activeExtractors = paloalto_morphlines_gp_gwc_fail

# paloalto_morphlines_gp_gwc_release
paloalto_morphlines_gp_gwc_release.name = PALOALTO_MORPHLINES_GP_GWC_RELEASE
paloalto_morphlines_gp_gwc_release.version = 1
paloalto_morphlines_gp_gwc_release.type = morphlines
paloalto_morphlines_gp_gwc_release.configFile=${morphlinepath}/paloalto_gp_gwc_release.morphlines
activeExtractors = paloalto_morphlines_gp_gwc_release

# paloalto_morphlines_gp_gwc_releaseVer2
paloalto_morphlines_gp_gwc_releaseVer2.name = PALOALTO_MORPHLINES_GP_GWC_RELEASEVER2
paloalto_morphlines_gp_gwc_releaseVer2.version = 2
paloalto_morphlines_gp_gwc_releaseVer2.type = morphlines
paloalto_morphlines_gp_gwc_releaseVer2.configFile=${morphlinepath}/paloalto_gp_gwc_releaseVer2.morphlines
activeExtractors = paloalto_morphlines_gp_gwc_releaseVer2

# paloalto_morphlines_gp_gwr_succ
paloalto_morphlines_gp_gwr_succ.name = PALOALTO_MORPHLINES_GP_GWR_SUCC
paloalto_morphlines_gp_gwr_succ.version = 1
```

Figure 14: Highlighted .Ini file section

4.4 Select the *Test* tab. In the case of a **.morphlines** parser, this will display a testing panel that will allow to submit sample logs using the Morphlines defined in the *Config* tab.



File: versions/1.5/morphlines/paloalto_gp_gwc_succVer2.morphlines

Key: TestLog

this is a test log

Figure 15: Morphlines test panel

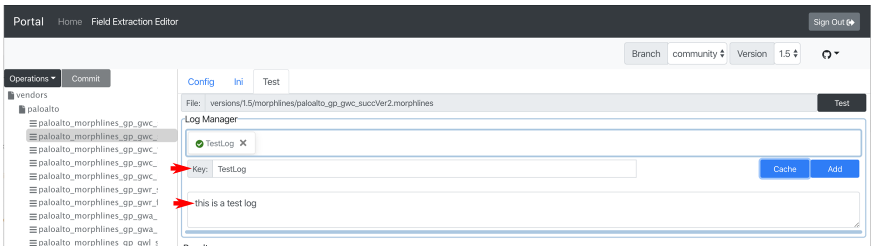
5 Testing a Morphlines

The web service is used to test each parser individually. Multiple logs can be sent to the web service which will process each log and echo back an array of objects described in listing 1. The *matches* attribute will not be populated in cases where the log item submitted doesn't match. If the log item matches, the resulting *keyvalue pairs* will be provided as a dictionary object in the *matches* attribute.

Listing 1: Test Object Definition

```
{
        "<logname>": {
                "log": "original_log_text",
                "matches": {
                        "key1": "value1",
                        "key2": "value2"
                }
        }
}
```

5.1 Select any of the *paloalto* vendor parsers in the tree. The *Config* tab should be presented. Select the *Test* tab. Type a log name in the *Key* input field and in the textarea type in a simulated log (any text will do).


Portal
Home
Field Extraction Editor
Sign Out
Branch
community
Version
1.5
Operations
Commit
vendors
paloalto
paloalto_morphlines_gp_gwc_
paloalto_morphlines_gp_gwc_
paloalto_morphlines_gp_gwc_
paloalto_morphlines_gp_gwc_
paloalto_morphlines_gp_gwc_
paloalto_morphlines_gp_gwr_
paloalto_morphlines_gp_gwr_
paloalto_morphlines_gp_gwa_
paloalto_morphlines_gp_gwa_
paloalto_morphlines_gp_gwl_
Config
Ini
Test
Log Manager
TestLog
Cache
Add

File: versions/1.5/morphlines/paloalto_gp_gwc_succVer2.morphlines

Key: TestLog

this is a test log

Figure 16: Defining a log entry

5.2 Click on the *Add* button and a small log icon should appear with a *green* check on it to delineate that this log is currently being displayed in the form.

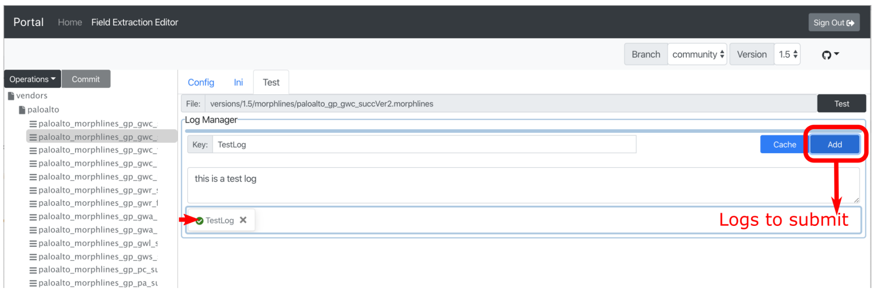


Figure 17: Adding a log entry to be tested

5.3 Press on the *Test* button. This will submit the log to the testing web service. This may take up to 30 seconds to return a result.

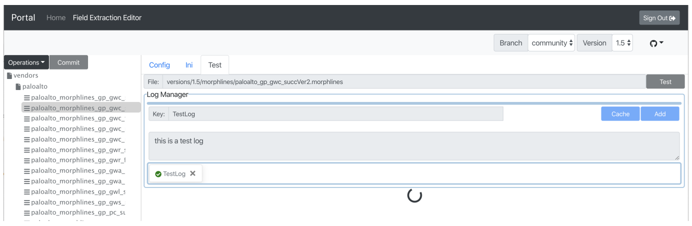
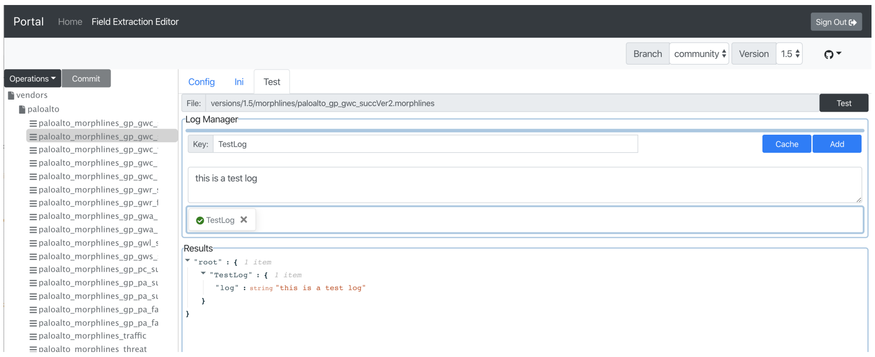


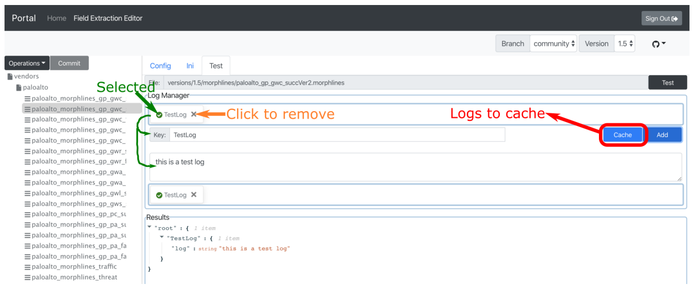
Figure 18: Waiting for the Morphlines test to return



```
▾ "root" : { 1 item
   ▾ "TestLog" : { 1 item
      "log" : string "this is a test log"
   }
}
```

Figure 19: Selected Morphlines doesn't match the submitted log

5.4 Click on the *Cache* button. This will add the current log into the cache and persist it across browser sessions and across the different parsers. The icon should appear at the top of the *Log Manager* section.


Portal
Home
Field Extraction Editor
Sign Out
Branch
community
Version
1.5
Operations
Commit
vendors
paloalto
paloalto_morphlines_gp_gwc_
paloalto_morphlines_gp_gwc_
paloalto_morphlines_gp_gwc_
paloalto_morphlines_gp_gwc_
paloalto_morphlines_gp_gwc_
paloalto_morphlines_gp_gwr_
paloalto_morphlines_gp_gwr_
paloalto_morphlines_gp_gwa_
paloalto_morphlines_gp_gwa_
paloalto_morphlines_gp_gwl_
paloalto_morphlines_gp_gws_
paloalto_morphlines_gp_pc_su
paloalto_morphlines_gp_pa_su
paloalto_morphlines_gp_pa_su
paloalto_morphlines_gp_pa_fa
paloalto_morphlines_gp_pa_fa
paloalto_morphlines_traffic
paloalto_morphlines_threat
Config
Ini
Test
Test
Selected
Log Manager
TestLog
Click to remove
Logs to cache
Cache
Add
TestLog
Results

File: versions/1.5/morphlines/paloalto_gp_gwc_succVer2.morphlines

Key: TestLog

this is a test log

```
▾ "root" : { 1 item
  ▾ "TestLog" : { 1 item
     "log" : string "this is a test log"
  }
}
```

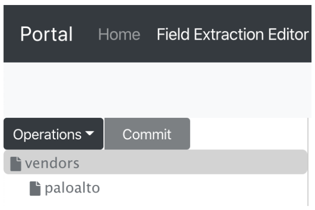
Figure 20: Waiting for the morphline test to return

The bottom listing of the log manager represent logs that are submitted to the testing web service. Logs can be removed from either the cache or current logs to send for testing by pressing the *X* on the log's icon.
Logs must be selected and manually added to either the list of logs to be submitted for testing or the cache using the *Add* and *Cache* buttons.

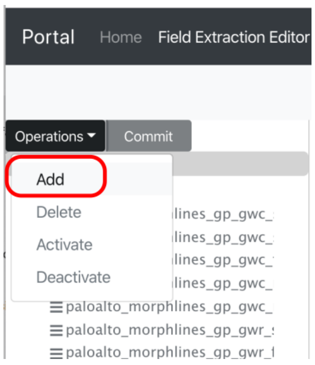
6 Adding Vendor files and parsers

6.1 Select the *Vendors* top-level item in the navigation tree.

6.2 Select *Add* to create a new vendor file.



(a) Select Vendors



(b) Select Add

6.3 Type in a new vendor name and press *Create*

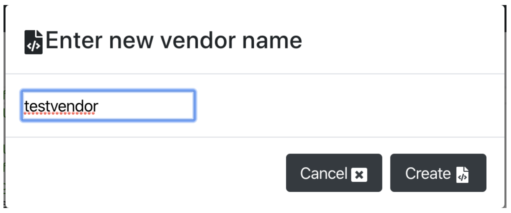


Figure 22: Selected Morphlines doesn't match the submitted log

6.4 An empty file should appear in the editor panel and the new vendor file should be selected in the navigator.

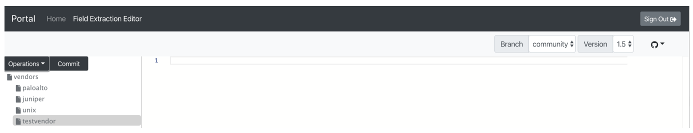


Figure 23: New empty vendor file

6.5 With the new vendor selected, click the *Operations* button and click *Add*. This will add a new parser to the vendor.

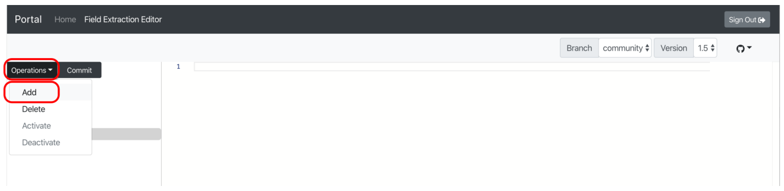
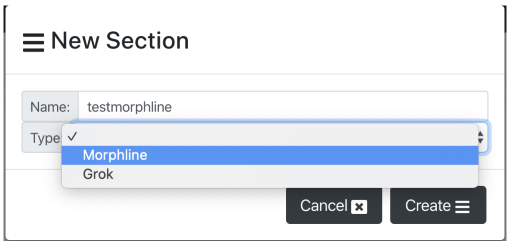


Figure 24: Adding a parser

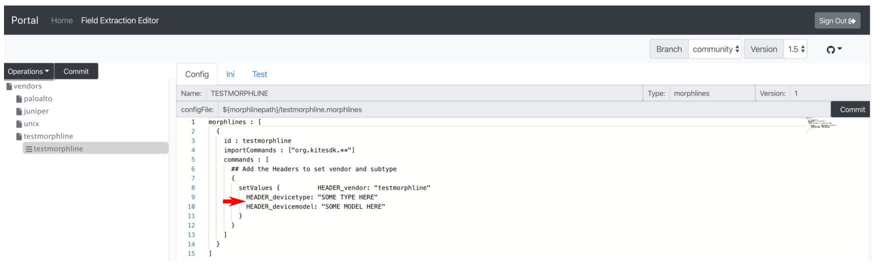
6.6 Enter a name for the new parser and select *Morphlines* in the type dialog



Name: testmorphline

Figure 25: New parser dialog

6.7 The contents of the new *morphlines* should appear in the **Config** tab. A template is offered with the only command being *setValues* which set the parser details prefixed by **HEADER**_. These prefixed values will be set as metadata inside the *matched* logs. More commands are available from http://kitesdk.org/docs/current/morphlines/

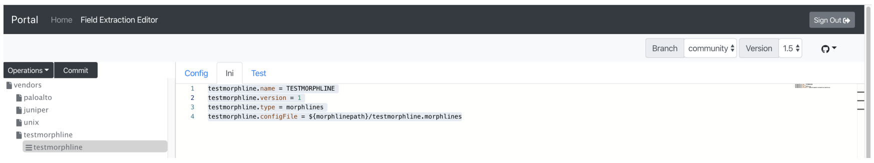


Name: TESTMORPHLINE
Type: morphlines
Version: 1
configFile: ${morphlinepath}/testmorphline.morphlines

```
morphlines : [
  {
    id : testmorphline
    importCommands : ["org.kitesdk.**"]
    commands : [
      ## Add the Headers to set vendor and subtype
      {
        setValues {          HEADER_vendor: "testmorphline"
          HEADER_devicetype: "SOME TYPE HERE"
          HEADER_devicemodel: "SOME MODEL HERE"
        }
      }
    ]
  }
]
```

Figure 26: New parser ini pane

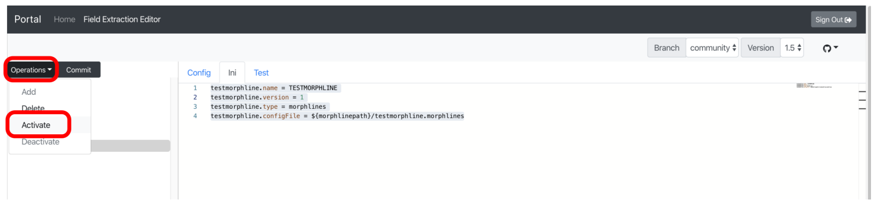
6.8 Click on the **Ini** tab. The selected parser in the navigation tree will be *highlighted* inside the properties file.



```
testmorphline.name = TESTMORPHLINE
testmorphline.version = 1
testmorphline.type = morphlines
testmorphline.configFile = ${morphlinepath}/testmorphline.morphlines
```

Figure 27: Activate section menu item

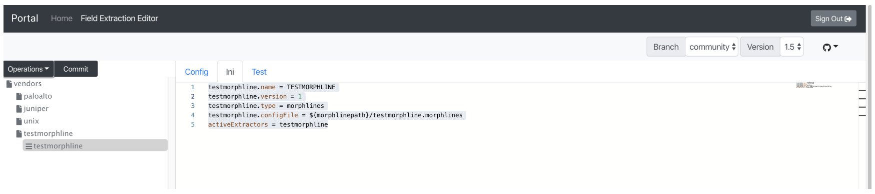
6.9 In order to eventually be used by remote logger nodes, the extractor needs to be activated. Click on the parser in the navigation tree, select *Operations* and then the *Activate* option



```
testmorphline.name = TESTMORPHLINE
testmorphline.version = 1
testmorphline.type = morphlines
testmorphline.configFile = ${morphlinepath}/testmorphline.morphlines
```

Figure 28: New parser config pane

6.10 The parser should now have a line saying **activeExtractors=** which indicates that the extractor will be considered to parse logs.



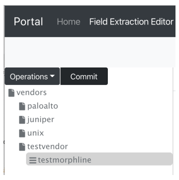
```
testmorphline.name = TESTMORPHLINE
testmorphline.version = 1
testmorphline.type = morphlines
testmorphline.configFile = ${morphlinepath}/testmorphline.morphlines
activeExtractors = testmorphline
```

Figure 29: Activated parser

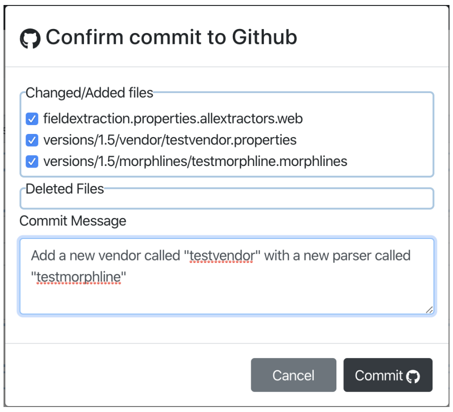
6.11 In order to save the new document to Github, click on *Commit* above the navigation tree

6.12 A dialog will appear asking for a comment to submit along with the new document versions

Three files will appear in the dialog. This is due to the fact that an **include** statement is needed for the new vendor definition inside the top-level config, then a **reference** to the actual morphlines file from within the vendor file is also needed along with the resulting **.morphlines** file.



(a) New parser test results, notice the **matches** section for the log



Confirm commit to Github

Changed/Added files

- ☑ fieldextraction.properties.allextractors.web
- ☑ versions/1.5/vendor/testvendor.properties
- ☑ versions/1.5/morphlines/testmorphline.morphlines

Deleted Files

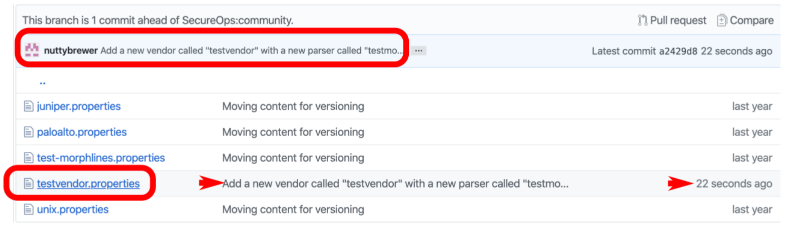
Commit Message

Add a new vendor called "testvendor" with a new parser called "testmorphline"

(b) Commit dialog

This step demonstrates one of the key features of this application, which is to automatically manage adding and removing parsers and vendors without having to understand the relationship between all the different text files.

6.13 Navigate to your Github repository and verify that your commit message has been received and the new file appears in *versions/1.5/vendors*.



This branch is 1 commit ahead of SecureOps:community.

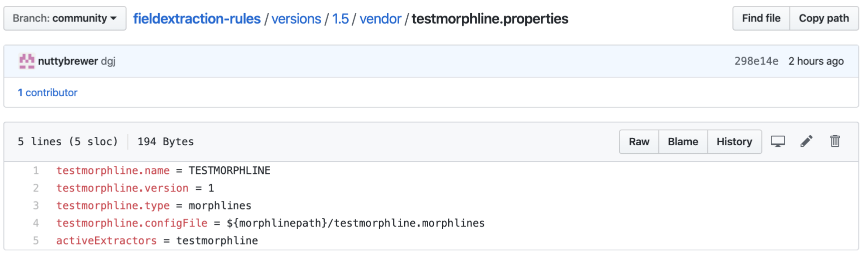
nuttybrewer Add a new vendor called "testvendor" with a new parser called "testmo... Latest commit a2429d8 22 seconds ago

..

juniper.properties	Moving content for versioning	last year
paloalto.properties	Moving content for versioning	last year
test-morphlines.properties	Moving content for versioning	last year
testvendor.properties	Add a new vendor called "testvendor" with a new parser called "testmo...	22 seconds ago
unix.properties	Moving content for versioning	last year

Figure 31: New parser test results, notice the **matches** section for the log

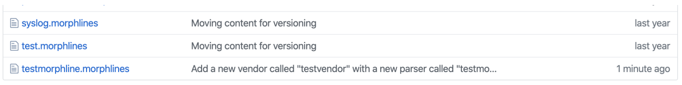
6.14 The contents of the file should match the *INI* tab inside the web app.


Branch: community
fieldextraction-rules / versions / 1.5 / vendor / testmorphline.properties
Find file
Copy path
nuttybrewer dgj
298e14e 2 hours ago
1 contributor
5 lines (5 sloc) 194 Bytes
Raw
Blame
History

```
testmorphline.name = TESTMORPHLINE
testmorphline.version = 1
testmorphline.type = morphlines
testmorphline.configFile = ${morphlinepath}/testmorphline.morphlines
activeExtractors = testmorphline
```

Figure 32: Contents of new **.properties** file are the same as the **INI** panel in the app

6.15 Also note that /versions/1.5/morphlines contains your new Morphlines definition file.



syslog.morphlines	Moving content for versioning	last year
test.morphlines	Moving content for versioning	last year
testmorphline.morphlines	Add a new vendor called "testvendor" with a new parser called "testmo...	1 minute ago

Figure 33: A new **.morphlines** file was committed to Github

6.16 Click on Test tab. Note the test log should be saved in the cache based on the testing in section 5

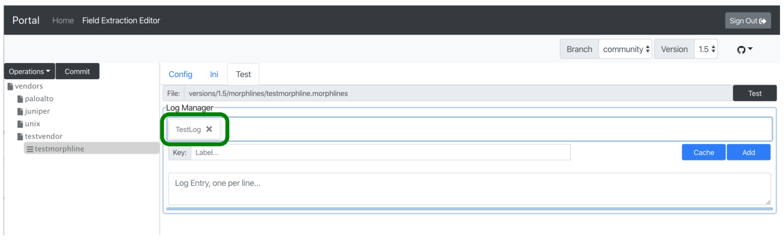
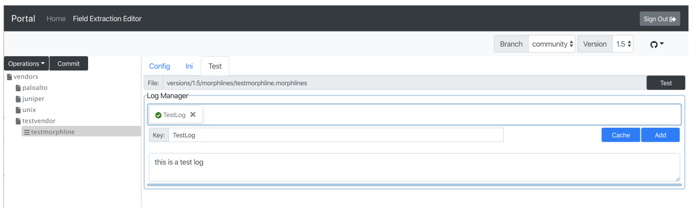


Figure 34: Test new parser pane

6.17 Select this test log and the click *Add* to add it to the test submissions

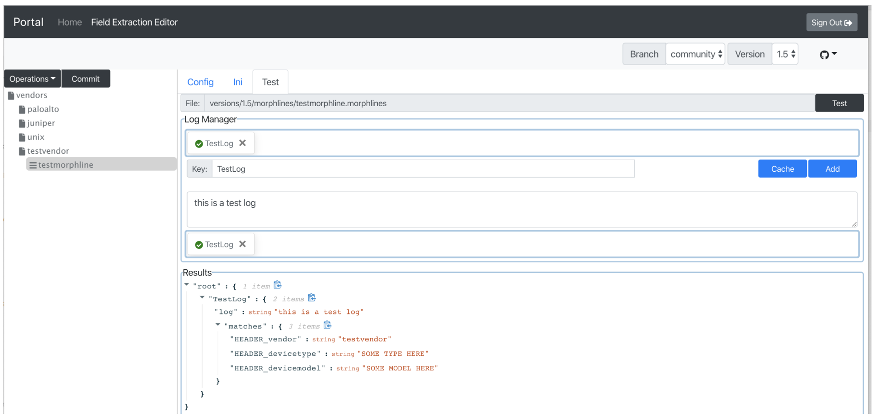


Key: TestLog

this is a test log

Figure 35: Submit the previous testlog using the new parser

6.18 Click on *Test* at the top right corner. This may take a few seconds



File: versions/1.5/morphlines/testmorphline.morphlines

Key: TestLog

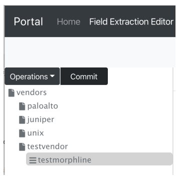
this is a test log

```
"root" : { 1 item
  "TestLog" : { 2 items
    "log" : string "this is a test log"
    "matches" : { 3 items
      "HEADER_vendor" : string "testvendor"
      "HEADER_devicetype" : string "SOME TYPE HERE"
      "HEADER_devicemodel" : string "SOME MODEL HERE"
    }
  }
}
```

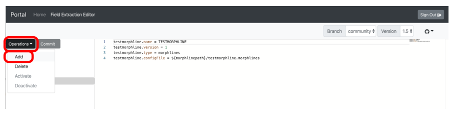
Figure 36: New parser test results, notice the **matches** section for the log

6.19 Repeat the steps above using the *Grok* parser type. Note that *Grok* parsers do not currently have a test tab since this is not implemented on the web service API as part of this project yet.

6.20 Click on the desired vendor on the navigation tab, click on *Operations* and *Add*



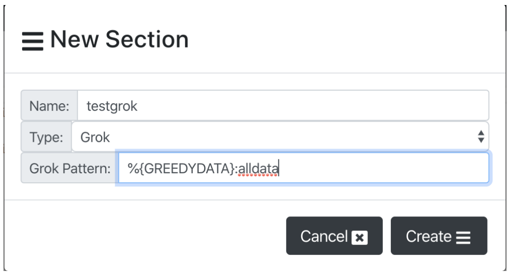
(a) Operations menu



```
testmorphline.name = TESTMORPHLINE
testmorphline.version = 1
testmorphline.type = morphlines
testmorphline.configFile = ${morphlinepath}/testmorphline.morphlines
```

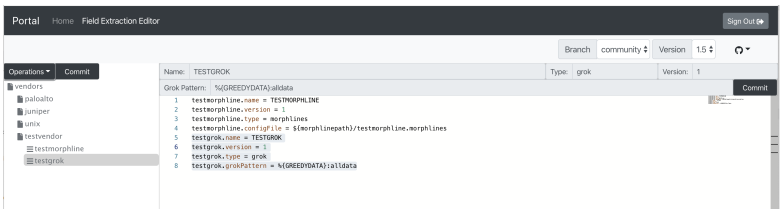
(b) Add parser

6.21 Select *Grok* type. For more information on Grok, please see https://www.elastic.co/guide/en/logstash/cu filters-grok.html. For now add any string in the *Grok Pattern* field



Name: testgrok
Type: Grok
Grok Pattern: %{GREEDYDATA}:alldata

Figure 38: Create Grok

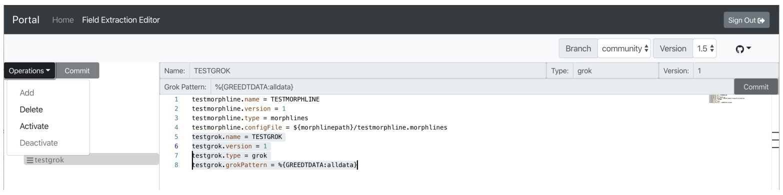


Name: TESTGROK
Type: grok
Version: 1
Grok Pattern: %{GREEDYDATA}:alldata

```
testmorphline.name = TESTMORPHLINE
testmorphline.version = 1
testmorphline.type = morphlines
testmorphline.configFile = ${morphlinepath}/testmorphline.morphlines
testgrok.name = TESTGROK
testgrok.version = 1
testgrok.type = grok
testgrok.grokPattern = %{GREEDYDATA}:alldata
```

Figure 39: Grok parser definition

6.22 Click on *Operations* and activate/deactivate the parser

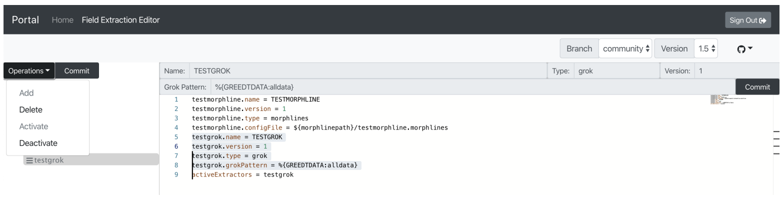


Name: TESTGROK Type: grok Version: 1

Grok Pattern: %{GREEDTDATA:alldata}

```
testmorphline.name = TESTMORPHLINE
testmorphline.version = 1
testmorphline.type = morphlines
testmorphline.configFile = ${morphlinepath}/testmorphline.morphlines
testgrok.name = TESTGROK
testgrok.version = 1
testgrok.type = grok
testgrok.grokPattern = %{GREEDTDATA:alldata}
```

Figure 40: Activate Grok



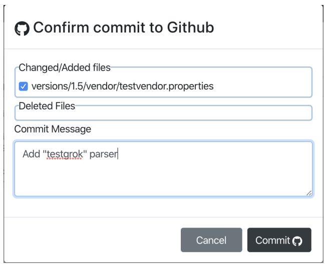
Name: TESTGROK Type: grok Version: 1

Grok Pattern: %{GREEDTDATA:alldata}

```
testmorphline.name = TESTMORPHLINE
testmorphline.version = 1
testmorphline.type = morphlines
testmorphline.configFile = ${morphlinepath}/testmorphline.morphlines
testgrok.name = TESTGROK
testgrok.version = 1
testgrok.type = grok
testgrok.grokPattern = %{GREEDTDATA:alldata}
activeExtractors = testgrok
```

Figure 41: Deactivate Grok

6.23 Click on *Commit* button, only one file should appear as we're only making changes to the **.properties** file.



Confirm commit to Github

Changed/Added files

☑ versions/1.5/vendor/testvendor.properties

Deleted Files

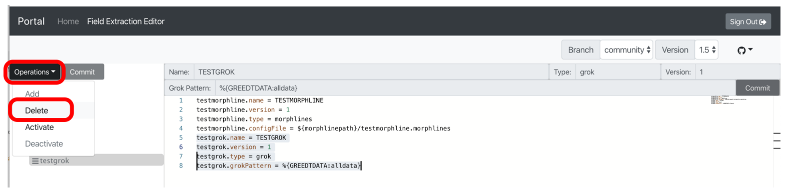
Commit Message

Add "testgrok" parser

Figure 42: Commit Grok parser

7 Delete Parsers and Vendor Files

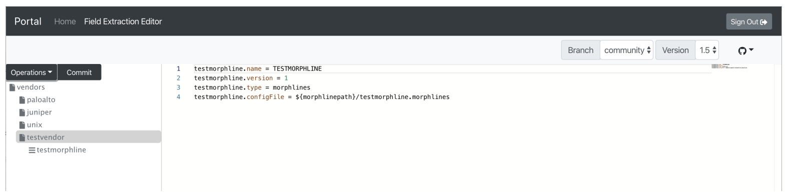
7.1 Click on the parser, select *Operations* and then *Delete*.



Name: TESTGROK Type: grok Version: 1
Grok Pattern: %{GREEDTDATA:alldata}

```
testmorphline.name = TESTMORPHLINE
testmorphline.version = 1
testmorphline.type = morphlines
testmorphline.configFile = ${morphlinepath}/testmorphline.morphlines
testgrok.name = TESTGROK
testgrok.version = 1
testgrok.type = grok
testgrok.grokPattern = %{GREEDTDATA:alldata}
```

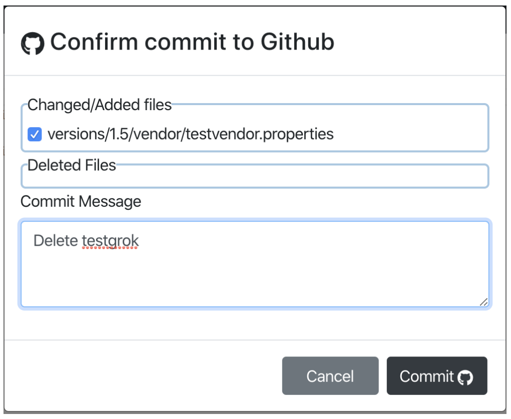
Figure 43: Delete the Grok parser



```
testmorphline.name = TESTMORPHLINE
testmorphline.version = 1
testmorphline.type = morphlines
testmorphline.configFile = ${morphlinepath}/testmorphline.morphlines
```

Figure 44: Deleted Grok parser

7.2 Click on *Commit*.


Confirm commit to Github

Changed/Added files

☑ versions/1.5/vendor/testvendor.properties

Deleted Files

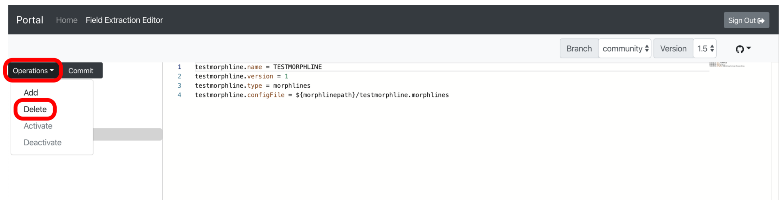
Commit Message

Delete testgrok

Figure 45: Commit deleted Grok parser

Since the vendor file is the only file modified it is the only one in the list.

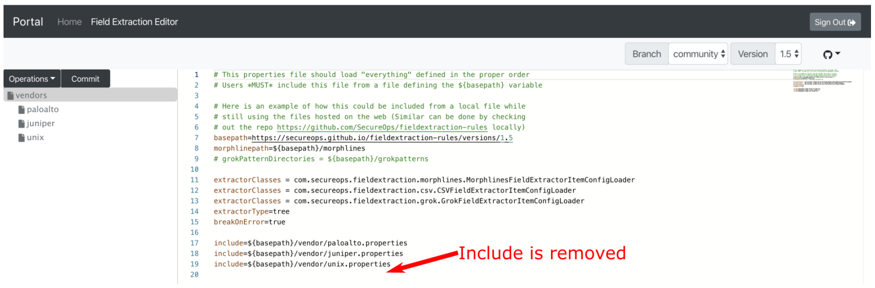
7.3 Click on the vendor file, select *Operations* and then *Delete*.



```
testmorphline.name = TESTMORPHLINE
testmorphline.version = 1
testmorphline.type = morphlines
testmorphline.configFile = ${morphlinepath}/testmorphline.morphlines
```

Figure 46: Delete the vendor file

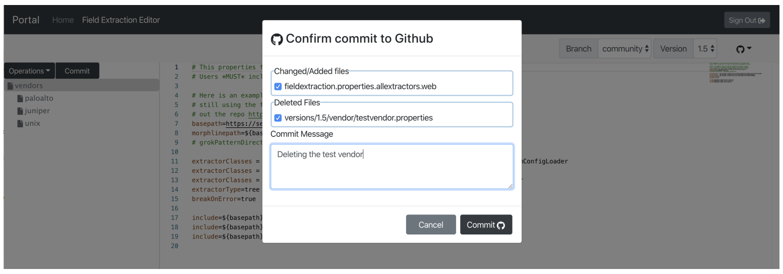
7.4 Click on *Vendors* in the navigation tree, notice the include for the vendor file is now gone.



```
# This properties file should load "everything" defined in the proper order
# Users *MUST* include this file from a file defining the ${basepath} variable

# Here is an example of how this could be included from a local file while
# still using the files hosted on the web (Similar can be done by checking
# out the repo https://github.com/SecureOps/fieldextraction-rules locally)
basepath=https://secureops.github.io/fieldextraction-rules/versions/1.5
morphlinepath=${basepath}/morphlines
# grokPatternDirectories = ${basepath}/grokpatterns

extractorClasses = com.secureops.fieldextraction.morphlines.MorphlinesFieldExtractorItemConfigLoader
extractorClasses = com.secureops.fieldextraction.csv.CSVFieldExtractorItemConfigLoader
extractorClasses = com.secureops.fieldextraction.grok.GrokFieldExtractorItemConfigLoader
extractorType=tree
breakOnError=true

include=${basepath}/vendor/paloalto.properties
include=${basepath}/vendor/juniper.properties
include=${basepath}/vendor/unix.properties
```

Figure 47: Confirm the vendor file is no longer included

7.5 *Commit* the results. Notice there are files in the changed/add and delete sections.



Confirm commit to Github

Changed/Added files

✓ fieldextraction.properties.allextractors.web

Deleted Files

✓ versions/1.5/vendor/testvendor.properties

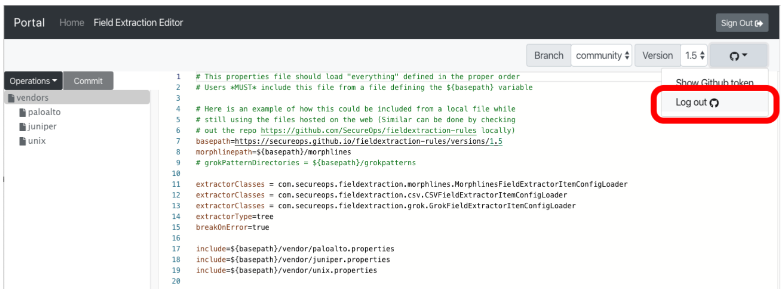
Commit Message

Deleting the test vendor

Figure 48: Commit deleted file

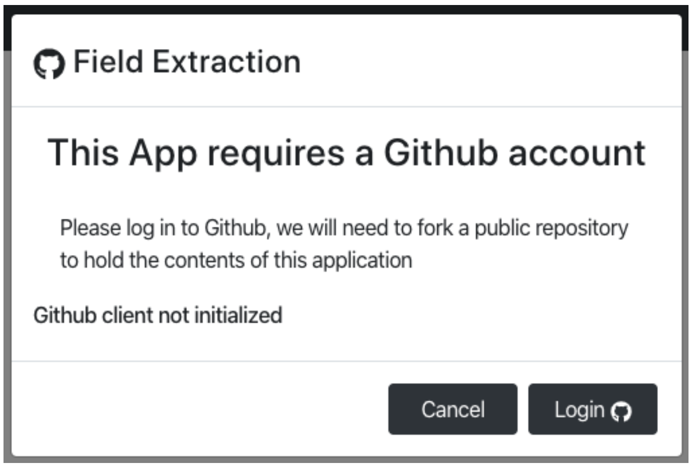
8 Log Out

8.1 Click on *Github* icon and click on *Log out*. The Github modal should appear.



```
# This properties file should load "everything" defined in the proper order
# Users *MUST* include this file from a file defining the ${basepath} variable

# Here is an example of how this could be included from a local file while
# still using the files hosted on the web (Similar can be done by checking
# out the repo https://github.com/SecureOps/fieldextraction-rules locally)
basepath=https://secureops.github.io/fieldextraction-rules/versions/1.5
morphlinepath=${basepath}/morphlines
# grokPatternDirectories = ${basepath}/grokpatterns

extractorClasses = com.secureops.fieldextraction.morphlines.MorphlinesFieldExtractorItemConfigLoader
extractorClasses = com.secureops.fieldextraction.csv.CSVFieldExtractorItemConfigLoader
extractorClasses = com.secureops.fieldextraction.grok.GrokFieldExtractorItemConfigLoader
extractorType=tree
breakOnError=true

include=${basepath}/vendor/paloalto.properties
include=${basepath}/vendor/juniper.properties
include=${basepath}/vendor/unix.properties
```

Figure 49: Show Github menu



Field Extraction

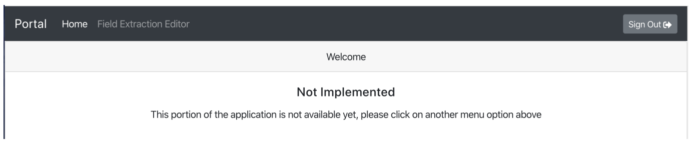
This App requires a Github account

Please log in to Github, we will need to fork a public repository to hold the contents of this application

Github client not initialized

Figure 50: Github login dialog is presented

8.2 You should be redirected to the landing page. If your token is more than an hour old, you will have been automatically signed out and provided with the login prompt.



Welcome

Not Implemented

This portion of the application is not available yet, please click on another menu option above

Figure 51: Landing page

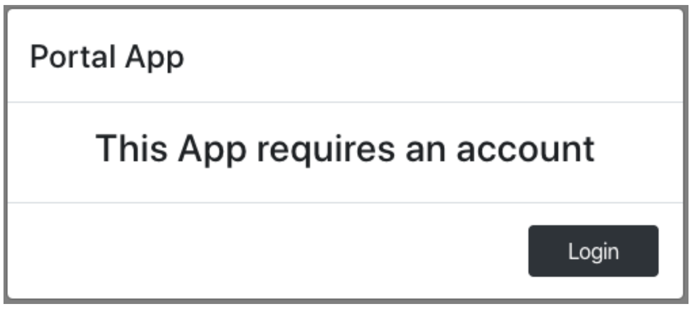


Figure 52: Login prompt

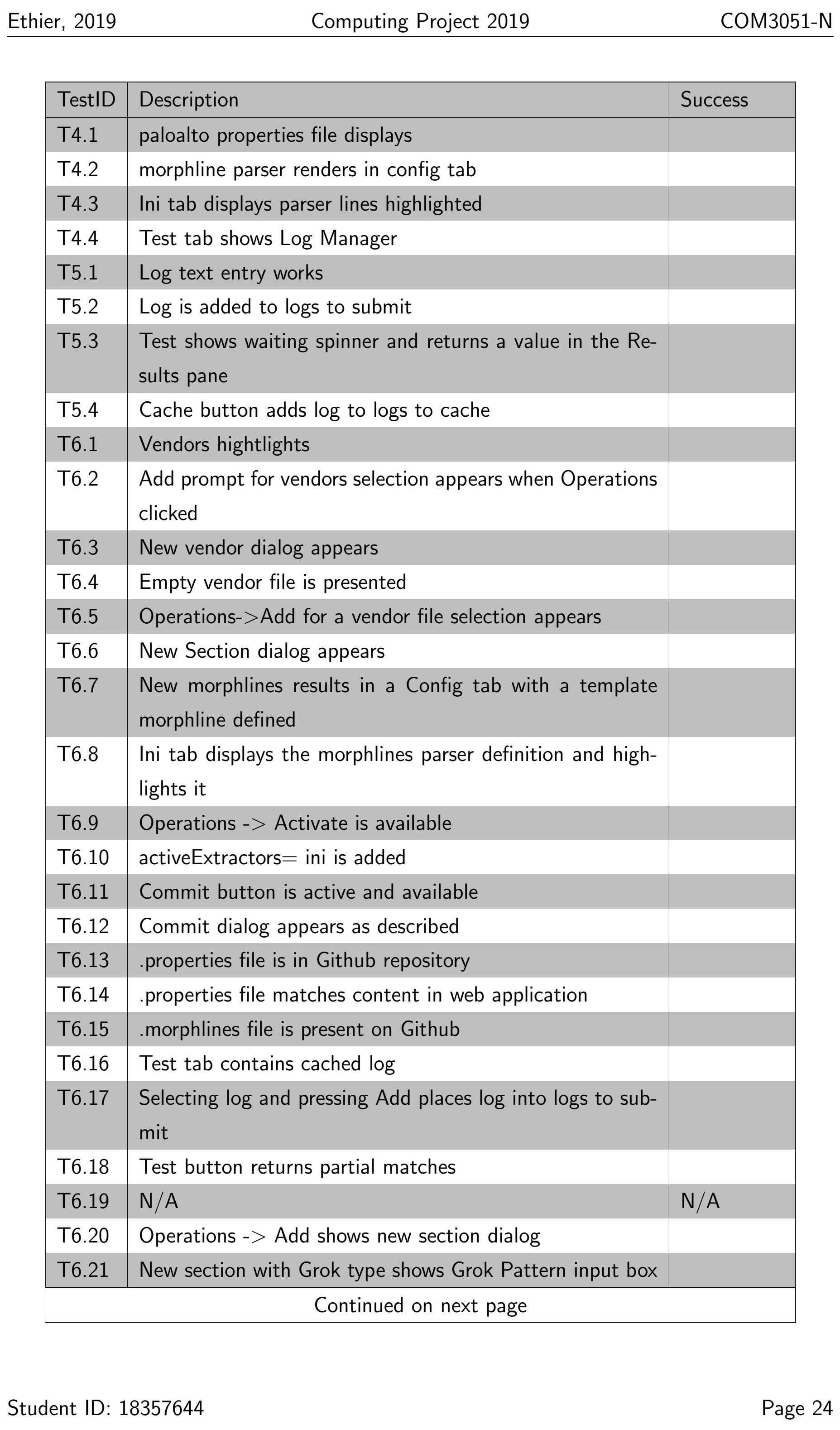
9 Survey Tables

The tables below are made available to provide feedback. Copy/paste each table into an email and respond to the author where applicable.
For the test table, please provide a short yes/no or X to represent a failure.
Please refer to https://github.com/nuttybrewer/CloudFormationReactJSWebsite/issues in order to report issues.
Include the **TextID** in the summary of the issue for easy reference by the author and other users. If an issue for the *TestID* already exists, please add a comment to the existing issue.

Table 1: Test Cases

TestID	Description	Success
T2.1	Login button works	
T2.2	Signup works	
T2.3	New password works	
T2.4	Validation email received	
T2.5	Verification works	
T2.6	Landing page works	
T3.1	Github log successful	
T3.2	Github fork successful	
T3.3	Github repository verified	
T3.4	FieldExtraction page appears	
T3.5	Github menu appears	
T3.6	Github token displayed	

Continued on next page

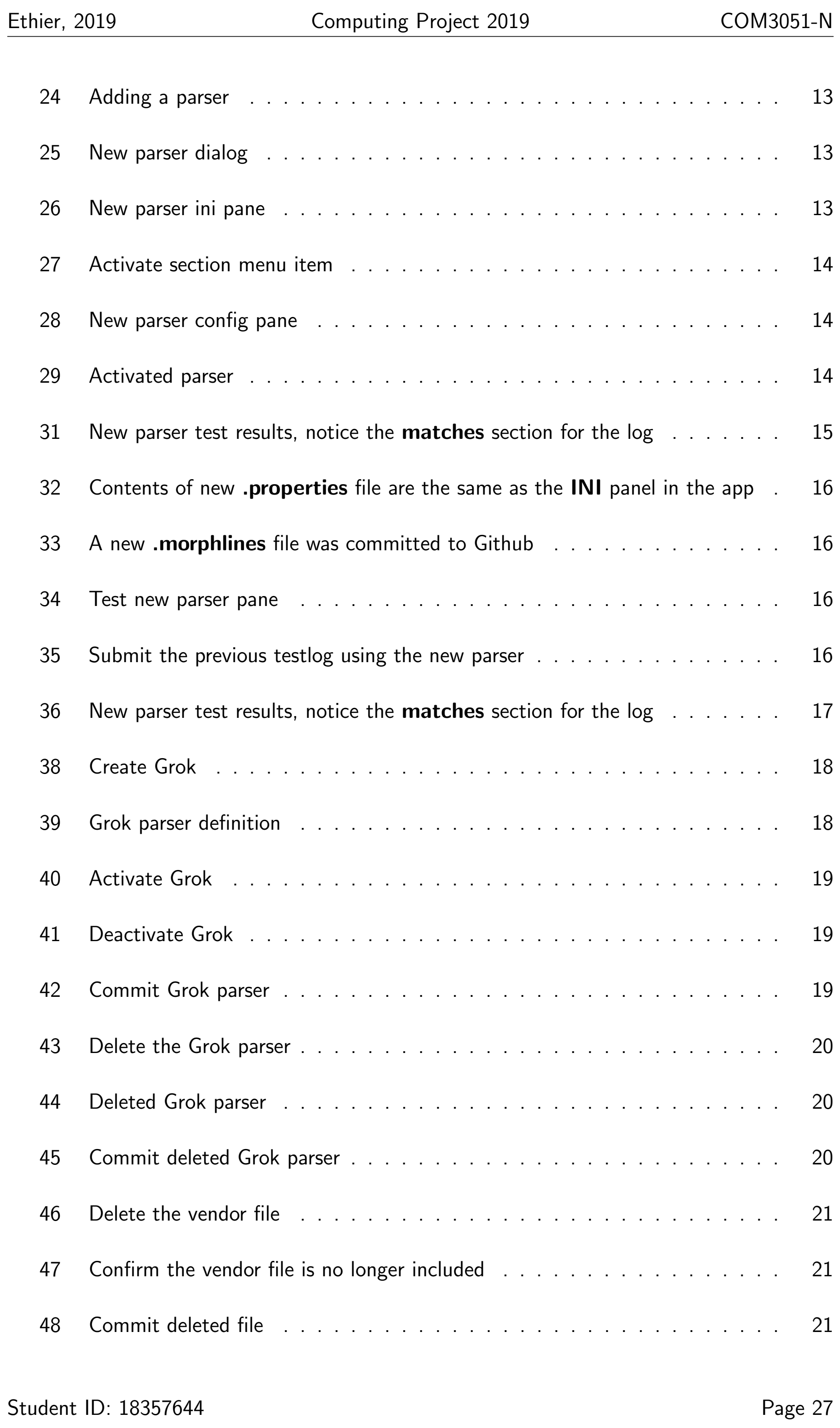


TestID	Description	Success
T4.1	paloalto properties file displays	
T4.2	morphline parser renders in config tab	
T4.3	Ini tab displays parser lines highlighted	
T4.4	Test tab shows Log Manager	
T5.1	Log text entry works	
T5.2	Log is added to logs to submit	
T5.3	Test shows waiting spinner and returns a value in the Results pane	
T5.4	Cache button adds log to logs to cache	
T6.1	Vendors hightlights	
T6.2	Add prompt for vendors selection appears when Operations clicked	
T6.3	New vendor dialog appears	
T6.4	Empty vendor file is presented	
T6.5	Operations->Add for a vendor file selection appears	
T6.6	New Section dialog appears	
T6.7	New morphlines results in a Config tab with a template morphline defined	
T6.8	Ini tab displays the morphlines parser definition and highlights it	
T6.9	Operations -> Activate is available	
T6.10	activeExtractors= ini is added	
T6.11	Commit button is active and available	
T6.12	Commit dialog appears as described	
T6.13	.properties file is in Github repository	
T6.14	.properties file matches content in web application	
T6.15	.morphlines file is present on Github	
T6.16	Test tab contains cached log	
T6.17	Selecting log and pressing Add places log into logs to submit	
T6.18	Test button returns partial matches	
T6.19	N/A	N/A
T6.20	Operations -> Add shows new section dialog	
T6.21	New section with Grok type shows Grok Pattern input box	

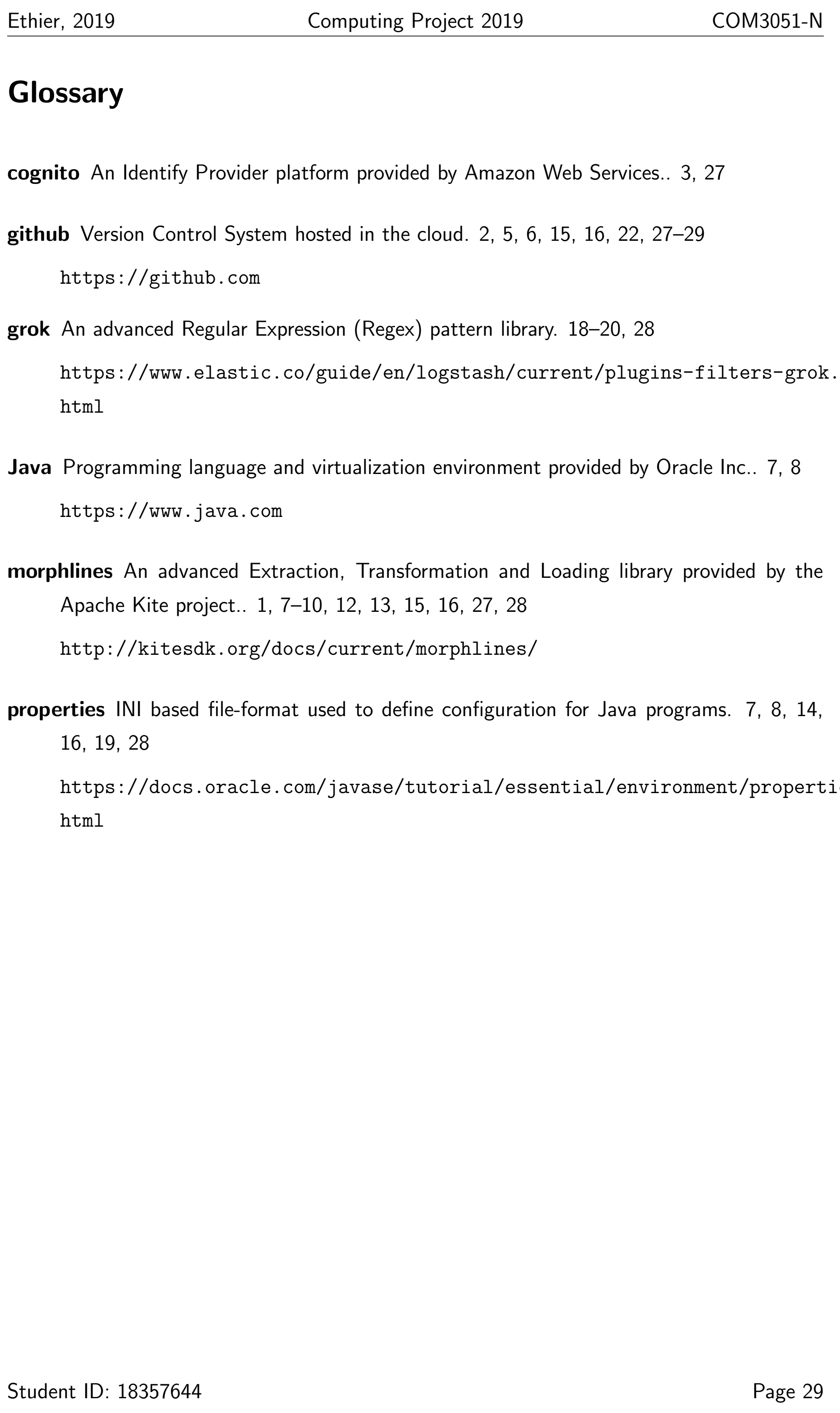
Continued on next page

TestID	Description	Success
T6.22	Activate/Deactivate adds and removes the activeExtractors= line	
T6.23	Commit modifies only the .properties file	
T7.1	Operations->Delete removes the parser definition from the .properties file	
T7.2	Commit shows only 1 .properties file	
T7.3	Operations -> Delete is available for the new vendor	
T7.4	Include line is removed from the top-level .properties file	
T7.5	Commit operation shows top-level file in changed files and vendor file in delete	
T8.1	Github logout is available and results in Github login screen	
T8.2	Cancel results in the landing page or login prompt	

List of Figures



List of Tables



Glossary

cognito An Identify Provider platform provided by Amazon Web Services.. 3, 27

github Version Control System hosted in the cloud. 2, 5, 6, 15, 16, 22, 27–29

`https://github.com`

grok An advanced Regular Expression (Regex) pattern library. 18–20, 28

`https://www.elastic.co/guide/en/logstash/current/plugins-filters-grok.html`

Java Programming language and virtualization environment provided by Oracle Inc.. 7, 8

`https://www.java.com`

morphlines An advanced Extraction, Transformation and Loading library provided by the Apache Kite project.. 1, 7–10, 12, 13, 15, 16, 27, 28

`http://kitesdk.org/docs/current/morphlines/`

properties INI based file-format used to define configuration for Java programs. 7, 8, 14, 16, 19, 28

`https://docs.oracle.com/javase/tutorial/essential/environment/properties.html`

C.1 User Tests

C.1.1 First customer test revision

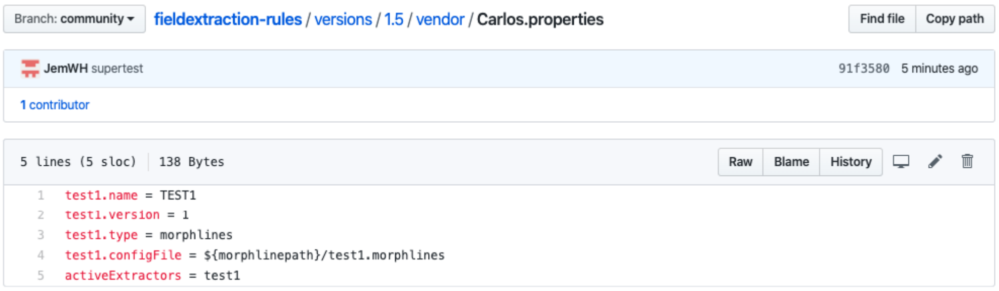
page left blank

Gmail **Patrick Ethier <nuttybrewer@gmail.com>**

Field Extraction Test Plan and Validation

Carlos Retamero <carlosretamero@googlemail.com> Tue, Oct 29, 2019 at 3:18
To: Patrick Ethier <nuttybrewer@gmail.com>

Table 1: Test Cases
T2.1 Login button works - X
T2.2 Signup works - X
T2.3 New password works - X
T2.4 Validation email received - X
T2.5 Verification works - **Returns to login after introducing verification code.**
T2.6 Landing page works - X
T3.1 Github log successful - X
T3.2 Github fork successful - X
T3.3 Github repository verified - X
T3.4 FieldExtraction page appears - X
T3.5 Github menu appears - X
T3.6 Github token displayed - X
T4.1 paloalto properties file displays - X
T4.2 morphline parser renders in config tab - X
T4.3 Ini tab displays parser lines highlighted - X
T4.4 Test tab shows Log Manager - X
T5.1 Log text entry works - X
T5.2 Log is added to logs to submit - X
T5.3 Test shows waiting spinner and returns a value in the Results pane - X
T5.4 Cache button adds log to logs to cache - X
T6.1 Vendors hightlights - X
T6.2 Add prompt for vendors selection appears when Operations clicked - X
T6.3 New vendor dialog appears - X
T6.4 Empty vendor file is presented - X
T6.5 Operations->Add for a vendor file selection appears - X
T6.6 New Section dialog appears - X
T6.7 New morphlines results in a Config tab with a template morphline defined - X
T6.8 Ini tab displays the morphlines parser definition and highlights it - X
T6.9 Operations -> Activate is available - X
T6.10 activeExtractors= ini is added - X
T6.11 Commit button is active and available - X
T6.12 Commit dialog appears as described - X
T6.13 .properties file is in Github repository - X
T6.14 .properties file matches content in web application - X
T6.15 .morphlines file is present on Github - X
T6.16 Test tab contains cached log - X
T6.17 Selecting log and pressing Add places log into logs to submit - X
T6.18 Test button returns partial matches - X
T6.20 Operations -> Add shows new section dialog - X
T6.21 New section with Grok type shows Grok Pattern input box - X
T6.22 Activate/Deactivate adds and removes the activeExtractors= line - X
T6.23 Commit modifies only the .properties file - **Commit function on webpage but did not show up in github**



```
test1.name = TEST1
test1.version = 1
test1.type = morphlines
test1.configFile = ${morphlinepath}/test1.morphlines
activeExtractors = test1
```

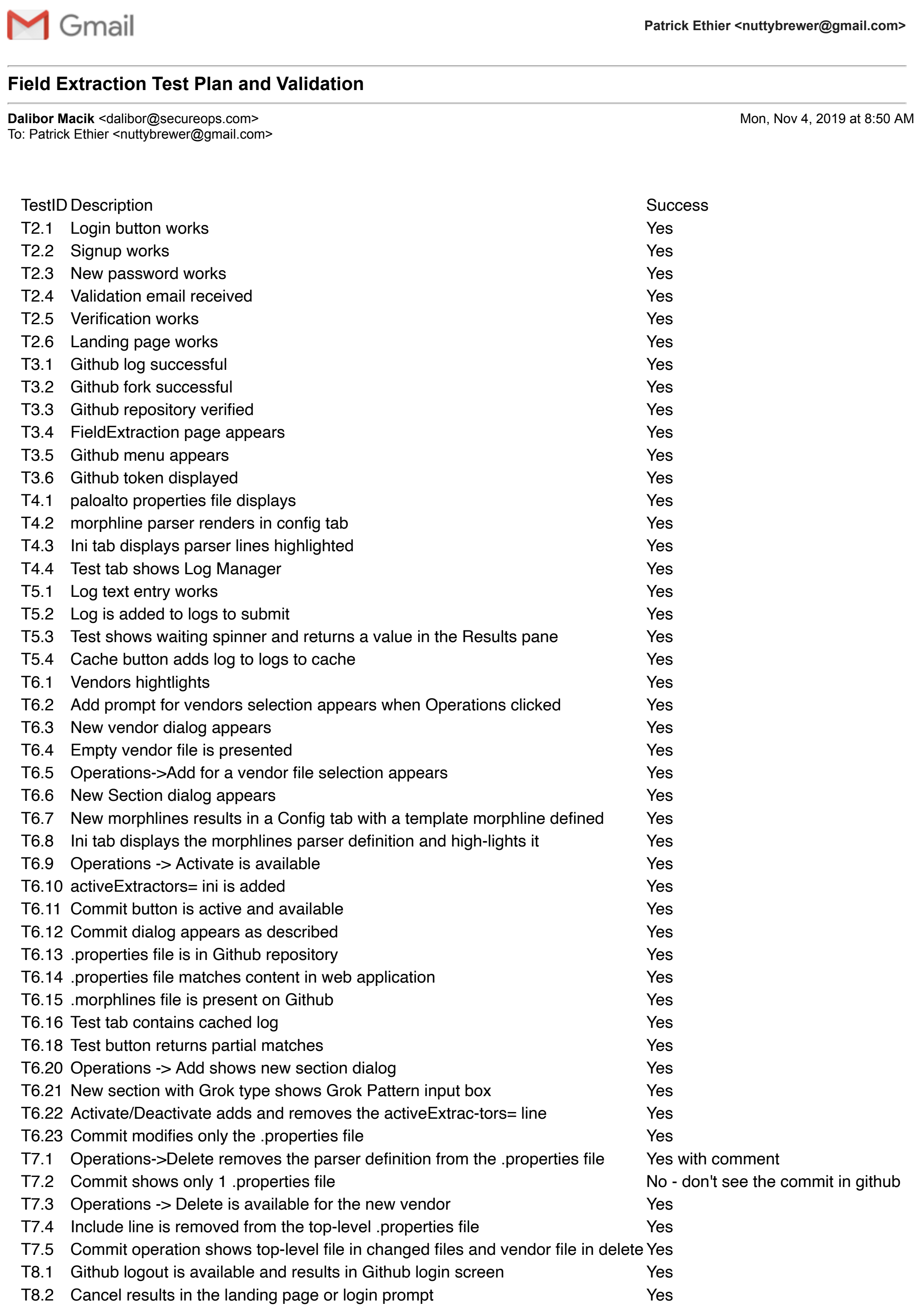
T7.1 Operations->Delete removes the parser definition from the .properties file- X
T7.2 Commit shows only 1 .properties file - X
T7.3 Operations -> Delete is available for the new vendor - X
T7.4 Include line is removed from the top-level .properties file - X
T7.5 Commit operation shows top-level file in changed files and vendor file in delete - **Morphile file did not delete in github**
T8.1 Github logout is available and results in Github login screen - X
T8.2 Cancel results in the landing page or login prompt - X

[Quoted text hidden]
--
Thank you,

Carlos Retamero Díaz

C.1.2 Final customer test acceptance

page left blank



Gmail

Patrick Ethier <nuttybrewer@gmail.com>

Field Extraction Test Plan and Validation

Dalibor Macik <dalibor@secureops.com>
To: Patrick Ethier <nuttybrewer@gmail.com>

Mon, Nov 4, 2019 at 8:50 AM

TestID	Description	Success
T2.1	Login button works	Yes
T2.2	Signup works	Yes
T2.3	New password works	Yes
T2.4	Validation email received	Yes
T2.5	Verification works	Yes
T2.6	Landing page works	Yes
T3.1	Github log successful	Yes
T3.2	Github fork successful	Yes
T3.3	Github repository verified	Yes
T3.4	FieldExtraction page appears	Yes
T3.5	Github menu appears	Yes
T3.6	Github token displayed	Yes
T4.1	paloalto properties file displays	Yes
T4.2	morphline parser renders in config tab	Yes
T4.3	Ini tab displays parser lines highlighted	Yes
T4.4	Test tab shows Log Manager	Yes
T5.1	Log text entry works	Yes
T5.2	Log is added to logs to submit	Yes
T5.3	Test shows waiting spinner and returns a value in the Results pane	Yes
T5.4	Cache button adds log to logs to cache	Yes
T6.1	Vendors hightlights	Yes
T6.2	Add prompt for vendors selection appears when Operations clicked	Yes
T6.3	New vendor dialog appears	Yes
T6.4	Empty vendor file is presented	Yes
T6.5	Operations->Add for a vendor file selection appears	Yes
T6.6	New Section dialog appears	Yes
T6.7	New morphlines results in a Config tab with a template morphline defined	Yes
T6.8	Ini tab displays the morphlines parser definition and high-lights it	Yes
T6.9	Operations -> Activate is available	Yes
T6.10	activeExtractors= ini is added	Yes
T6.11	Commit button is active and available	Yes
T6.12	Commit dialog appears as described	Yes
T6.13	.properties file is in Github repository	Yes
T6.14	.properties file matches content in web application	Yes
T6.15	.morphlines file is present on Github	Yes
T6.16	Test tab contains cached log	Yes
T6.18	Test button returns partial matches	Yes
T6.20	Operations -> Add shows new section dialog	Yes
T6.21	New section with Grok type shows Grok Pattern input box	Yes
T6.22	Activate/Deactivate adds and removes the activeExtrac-tors= line	Yes
T6.23	Commit modifies only the .properties file	Yes
T7.1	Operations->Delete removes the parser definition from the .properties file	Yes with comment
T7.2	Commit shows only 1 .properties file	No - don't see the commit in github
T7.3	Operations -> Delete is available for the new vendor	Yes
T7.4	Include line is removed from the top-level .properties file	Yes
T7.5	Commit operation shows top-level file in changed files and vendor file in delete	Yes
T8.1	Github logout is available and results in Github login screen	Yes
T8.2	Cancel results in the landing page or login prompt	Yes

[Quoted text hidden]
--
Dalibor Macik, GCIA
Level 3 Network Security Analyst
SecureOps Inc.

dalibor@secureops.com

D Code Listings and Examples

Listing 15: CircleCI deployment of OAuth functionality

```yaml
version: 2.1
orbs:
  aws-cli: circleci/aws-cli@0.1.13
jobs:
  build:
    docker:
      - image: circleci/node:latest
    executor:
      name: aws-cli/default
      python-version: "3.5"
    steps:
      - checkout
      - run:
          name: Create FQDN and stack name environment variables
          command: |
            RAW_STACK_NAME="${CLOUDFRONT_SITE_NAME}-${ROUTE53_DOMAIN_NAME}-${FUNCTION_NAME}-${CIRCLE_BR
            STACK_NAME=$(echo $RAW_STACK_NAME | tr -cs "[:alnum:][:cntrl:]" "-"); \
            DISTRIBUTION_DOMAIN="https://${CLOUDFRONT_SITE_NAME}.${ROUTE53_DOMAIN_NAME}"; \
            echo "${STACK_NAME}"; \
            echo "${DISTRIBUTION_DOMAIN}"; \
            echo "export STACK_NAME=${STACK_NAME}" >> $BASH_ENV; \
            echo "export DISTRIBUTION_DOMAIN=$DISTRIBUTION_DOMAIN" >> $BASH_ENV; \
            echo "export LAMBDA_S3_PREFIX=${STACK_NAME}/" >> $BASH_ENV; \
            echo "export LAMBDA_FUNCTION_NAME=${STACK_NAME}" >> $BASH_ENV; \
      - run:
          name: update-npm
          command: 'sudo npm install -g npm@latest'
      - restore_cache: # special step to restore the dependency cache
          # Read about caching dependencies: https://circleci.com/docs/2.0/caching/
          key: dependency-cache-{{ checksum "nodejs/package.json" }}
      - run:
          name: install-node-dependencies
          command: cd nodejs && npm install
      - save_cache: # special step to save the dependency cache
          key: dependency-cache-{{ checksum "nodejs/package.json" }}
          paths:
            - ./nodejs/node_modules
      - run: # run tests
          name: run npm test
          command: cd nodejs && npm test
          environment:
            JEST_JUNIT_OUTPUT: "reports/junit/junit.xml"
      - store_artifacts: # special step to save test results as as artifact
          # Upload test summary for display in Artifacts: https://circleci.com/docs/2.0/artifacts/
          path: nodejs/reports/junit
          prefix: tests
      - store_artifacts: # for display in Artifacts: https://circleci.com/docs/2.0/artifacts/
          path: nodejs/coverage
          prefix: coverage
      - store_test_results: # for display in Test Summary: https://circleci.com/docs/2.0/collect-test-da
          path: nodejs/reports/junit
      - run:
          name: Clean up env.js
          command: rm nodejs/env.js
      - run: # Package env.sh for production deployment since Lambda@Edge doesn't support environment va
          name: Get JWT Keys from Cognito
          command: |
            echo 'export COGNITO_JWKS=$(cd nodejs && node generatejwt.js)' >> $BASH_ENV;
      - run:
          name: Generate environment variables file
          command: cd nodejs && node_modules/.bin/envsub __templates__/env.js.template env.js
      - run:
          name: ENV sanity check
          command: cat nodejs/env.js
      - run:
          name: package the output
          command: cd nodejs && npm run pack
      - run:
          name: create build directory
          command: mkdir -p nodejs/build
      - run:
          name: copy output to a file with the branchname
          command: mv nodejs/$(cat nodejs/package.json | jq -r '.name').zip nodejs/build/$LAMBDA_FUNCTIO
      - run: ls -alh nodejs/
      - aws-cli/install
      - aws-cli/configure:
          profile-name: example
      - run:
          name: Deploy Zip File to lambda function S3 Bucket
          command: |
            if [[ ! -z $LAMBDA_S3_BUCKET_NAME ]]; then
              $(aws s3api head-bucket --bucket $LAMBDA_S3_BUCKET_NAME);
              if [[ $? == 0 ]]; then
                aws s3 cp nodejs/build/$STACK_NAME.$CIRCLE_BRANCH.zip \
                s3://$LAMBDA_S3_BUCKET_NAME/$LAMBDA_S3_PREFIX$STACK_NAME.zip;
              else
                echo "bucket does not exist or permission is not there to view it.";
                exit 1;
              fi
            fi
      - run:
          name: Deploy Lambda Function to AWS
          command: |
            aws cloudformation deploy --stack-name $STACK_NAME \
            --template-file ./CloudFormation/lambda.json \
            --parameter-overrides \
            "lambdaBucketName=$LAMBDA_S3_BUCKET_NAME" \
            "lambdaZipPath=$LAMBDA_S3_PREFIX$LAMBDA_FUNCTION_NAME.zip" \
            "lambdaFunctionName=$LAMBDA_FUNCTION_NAME" \
            "lambdaZipObjectVersion=$(aws s3api list-object-versions \
            --bucket $LAMBDA_S3_BUCKET_NAME \
            --prefix $LAMBDA_S3_PREFIX$LAMBDA_FUNCTION_NAME.zip \
            --max-items 1 | jq -r '.Versions[0].VersionId')" \
            "buildVersion=$CIRCLE_BUILD_NUM" \
            --capabilities CAPABILITY_IAM
      - run:
          name: Create a new version of the lambda code
          command: |
            echo 'export LAMBDA_VER=$(aws lambda publish-version \
            --function-name $LAMBDA_FUNCTION_NAME | jq -r '.Version')' >> $BASH_ENV
      - run:
          name: Create a tag reflecting the build number
          command: |
            source $BASH_ENV;
            aws lambda create-alias --function-name $LAMBDA_FUNCTION_NAME \
            --name "cideployed-${CIRCLE_BUILD_NUM:-$LAMBDA_VER}" \
            --function-version $LAMBDA_VER
      - run:
          name: Clean up
          command: cd nodejs && npm run cleanup
```

Listing 16: Amazon CloudFormation script to deploy OAuth capabilities

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Parameters": {
    "lambdaBucketName": {
      "Type": "String",
      "Description": "The S3 bucket name where the zipfile for the function is stored"
    },
    "lambdaZipPath": {
      "Type": "String",
      "Description": "The path to the zipfile inside the bucket."
    },
    "lambdaFunctionName": {
      "Type": "String",
      "Description": "The name or label to give the Lambda function"
    },
    "lambdaZipObjectVersion": {
      "Type": "String",
      "Description": "VersionId of the zipFile inside S3"
    }
  },
  "Resources": {
    "CreateJWTValidatorLambdaRole": {
      "Type": "AWS::IAM::Role",
      "Properties": {
        "AssumeRolePolicyDocument": {
          "Version": "2012-10-17",
          "Statement": [{
            "Effect": "Allow",
            "Principal": {
              "Service": ["lambda.amazonaws.com", "edgelambda.amazonaws.com"]
            },
            "Action": ["sts:AssumeRole"]
          }]
        },
        "ManagedPolicyArns": [
          "arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole"
        ]
      }
    },
    "CreateAndReturnJWTValidatorLambdaFunction": {
      "Type": "AWS::Lambda::Function",
      "Properties": {
        "FunctionName": { "Ref": "lambdaFunctionName"},
        "Code": {
          "S3Bucket": { "Ref": "lambdaBucketName" },
          "S3Key": { "Ref": "lambdaZipPath" },
          "S3ObjectVersion": { "Ref": "lambdaZipObjectVersion" },
        },
        "Handler": "index.handler",
        "Runtime": "nodejs10.x",
        "Timeout": "5",
        "Role": { "Fn::GetAtt": [ "CreateJWTValidatorLambdaRole", "Arn" ] }
      }
    }
  },
  "Outputs": {
    "JWTValidatorOutput": {
      "Description": "ARN of the Java Web Token verification function",
      "Value": { "Fn::GetAtt": [ "CreateAndReturnJWTValidatorLambdaFunction", "Arn" ] },
      "Export": { "Name" : {"Fn::Sub": "${AWS::StackName}-JWTValidatorArn" }}
    }
  }
}
```

Listing 17: Amazon Lambda function that implements OAuth capabilities

```
'use strict';
const util = require('util');
const fetch = require('node-fetch');
const { Headers } = require('node-fetch');
const { URLSearchParams } = require('url');
const url = require('url');
const querystring = require('querystring');
const env = require('./env');
const jwtCheck = require('./jwt');
const myutil = require('./util');


const getReferer = (cfrequest, domain) => {
  if (cfrequest.headers && cfrequest.headers.referer) {
    const referer_item = cfrequest.headers.referer.find((item) => item.key === 'Referer');
    if(referer_item && referer_item.value) {
      console.log(`referer_item.value is ${referer_item.value}`);
      const parsed_referer = url.parse(referer_item.value);
      console.log(`Evaluating referer as ${parsed_referer.hostname} and ${domain.hostname}`);
      if(domain.hostname === parsed_referer.hostname) {
        console.log(`Matched domain, setting the referer as path ${parsed_referer.path}`)
        return parsed_referer.path;
      }
    }
  }
  return null;
};

exports.handler = (event, context, lambda_return_cb) => {
  console.log(util.inspect(event, {
    showHidden: false,
    depth: null
  }));

  const cfrequest = event.Records[0].cf.request;
  const uri = cfrequest.uri;
  const queryString = cfrequest.querystring;
  const domain = url.parse(env.DISTRIBUTION_DOMAIN);
  // Load the default provider from the path, default to Cognito
  var provider=env.OAUTH_PROVIDERS.cognito; // Sane default


  // OAuth or is this just a JWT check?
  if (!uri.startsWith('/oauth')) {
    console.log("Matching simple JWT check");
    if (myutil.checkAuth(cfrequest)) {
      lambda_return_cb(null, cfrequest);
      return true;
    }
    lambda_return_cb(null, myutil.oauth_redirect(uri, provider, getReferer(cfrequest, domain)));
    return false;
  }

  const provider_match = uri.match(/\/oauth\/([a-zA-Z0-9-]+)/);
  if (provider_match) {
    const potential_provider = env.OAUTH_PROVIDERS[provider_match[1]];
    if (potential_provider) {
      console.log("Setting OAuth provider to " + potential_provider.local_oauth_name);
      provider = potential_provider;
    }
  }

  // Check auth if the provider is not Cognito
  if (!provider.primary) {
    if (!myutil.checkAuth(cfrequest)) {
      console.log("Primary authentication failed trying to get to " + uri);
      // TODO Should look through and find the primary provider manually,
      // for now just use Cognito
      lambda_return_cb(null, myutil.oauth_redirect(uri, env.OAUTH_PROVIDERS.cognito, getReferer(cfrequest, domain)));
      return false;
    }
  }

  // Check if this is an authorization request
  if(uri == "/oauth/" + provider.local_oauth_name + provider.local_authorize_path) {
    lambda_return_cb(null, myutil.oauth_redirect(uri, provider, getReferer(cfrequest, domain)) );
    return true;
  }

  // Check if this is a token response from the provider

  if(uri.startsWith("/oauth/" + provider.local_oauth_name + provider.local_token_path)) {
    if (queryString) {
      console.log("QueryString detected: " + util.inspect(queryString, {
        showHidden: false,
        depth: null
      }));
      const qsItems = querystring.parse(queryString);
      var next_uri;
      console.log(`Returned from provider, qsItems is ${JSON.stringify(qsItems)}`);
      if (qsItems[provider.provider_state_key]) {
        // Check if we made the request, if we didn't return a 404
        next_uri = myutil.oauth_state_check(qsItems[provider.provider_state_key]);
        console.log(`Final redirect has a next uri of ${next_uri}`);
        if (next_uri === null) {
          // Invalid requests don't deserve an explanation!
          lambda_return_cb(null, { "status": 404} );
          return false;
        }
      }
      if (qsItems.code) {
        console.log("Code detected: " + qsItems.code);
        const params = new URLSearchParams();
        params.append('grant_type', 'authorization_code');
        params.append('client_id', provider.client_id);
        // Use uri instead of the provider.local_token_path because they should be the same
        params.append('redirect_uri', env.DISTRIBUTION_DOMAIN + uri);
        params.append('code', qsItems.code);
        var headers = {};
        headers['Accept'] = 'application/json';
        if (provider.post_auth_type === 'Authorization') {
          headers['Authorization'] = 'Basic ' +
            Buffer.from(provider.client_id + ":" + provider.client_secret).toString('base64');
        }
        else if (provider.post_auth_type === 'Parameter') {
          params.append('client_secret', provider.client_secret);
        }
        console.log("Fetching POST" + provider.provider_url + provider.provider_token_path);
        console.log("Params:" + util.inspect(params, {
          showHidden: false,
          depth: null
        }));
        console.log("Headers: " + util.inspect(headers, {
          showHidden: false,
          depth: null
        }));
        return fetch(provider.provider_url + provider.provider_token_path, {
            method: 'POST',
            body: params,
            headers: headers
          })
          .then(res => {
            console.log("Fetch result: " + util.inspect(res, {
              showHidden: false,
              depth: null
            }));
            return res.json(); })
          .then(json => {
            console.log("authorization_response received json: " +
              util.inspect(json, {
                showHidden: false,
                depth: null
              })
            );
            console.log(provider);
            lambda_return_cb(null, myutil.final_redirect(next_uri, json.access_token, provider));
            return true;
          })
          .catch(error => {
            console.log("Error fetching auth code: " + util.inspect(error, {
              showHidden: false,
              depth: null
            }));
            lambda_return_cb(null, {status: 503, statusDescription: error.message});
            return false;
          });
      }
    }
  }

  console.log("Authorization fell through");
  lambda_return_cb(null, {status: 404, statusDescription: "Authorization fell through"});
  return false;
};
```

Listing 18: .deploy.env file

```
export "LAMBDA_S3_BUCKET_NAME=lambdabucket";
export "FUNCTION_NAME=CFOauthEndpoints";
export "COGNITO_USERPOOL_ID=us-east-1_";
export "COGNITO_CLIENT_ID=abcdefg";
export "COGNITO_DOMAIN=https://customer.auth.us-east-1.amazoncognito.com";
export "COGNITO_CLIENT_SECRET=abcdefg";
export "GITHUB_CLIENT_ID=1234567890";
export "GITHUB_CLIENT_SECRET=abcdefg";
export "CLOUDFRONT_SITE_NAME=portal";
export "ROUTE53_DOMAIN_NAME=customer.tld";\
export "JWT_SIGNATURE_SECRET=secret";
```

Listing 19: Using GitHub raw API to commit files

```
  commitContents(files, message) {
    this.setState({showCommitModal: false});
    const { client, reponame, owner, branch } = this.props;
    const { data } = this.state;

    const filesToCommit = [];
    if (! files || files.length === 0) {
      return new Promise((resolve) => resolve());
    }
    return new Promise((resolve, reject) => {
      client.git.getRef({
        owner:owner,
        repo: reponame,
        ref: `heads/${branch}`}
      ).then((ref) => {
          return Promise.all(
            files.map((path) => {
              return client.git.createBlob({
                owner: owner,
                repo: reponame,
                content: data[path].decoded
              })
              .then((blob) => {
                filesToCommit.push({
                  sha: blob.data.sha,
                  path: path,
                  mode: '100644',
                  type: 'blob'
                });
              })
            })
          ).then( () =>
            client.git.createTree({
              owner:owner,
              repo:reponame,
              tree: filesToCommit,
              base_tree: ref.data.object.sha
            }).then((tree) =>
              client.git.createCommit({
                owner: owner,
                repo:reponame,
                message: message,
                tree: tree.data.sha,
                parents: [ref.data.object.sha]
              }).then((commit) =>
                client.git.updateRef({
                  owner: owner,
                  repo: reponame,
                  ref: `heads/${branch}`,
                  sha: commit.data.sha
                }).then(() => {
                  files.forEach((item) => {
                    data[item].changed = false
                  })
                  this.setState({data: data});
                  resolve();
                })
              )
            )
          )
      })
      .catch((error) => {
        console.log(`Commit failed ${error.message}`);
        reject(error)
      })
    })
  }
```

E Deployment Guide

page left blank

Computing Project 2019 Deployment Guide

Patrick Ethier

December 1, 2019

1

Contents

1 Prequisites

The customer's portal shell must be deployed with the following *.deploy.env* defined:

Listing 1: Default portal settings

```
export DOMAINNAME=redlabnet.com;
export SITENAME=bob2;
export ACM_STACKNAME="ACMCertificateApprover-master";
export AWS_DEFAULT_REGION=us-east-1
export "COGNITO_USERPOOL_ID=us-east-1_abcdefg";

export "COGNITO_DOMAIN=https://poolid.auth.us-east-1.amazoncognito.com";
export "COGNITO_CLIENT_ID=abcdefg";
export "COGNITO_CLIENT_SECRET=secret";
export "JWT_SIGNATURE_SECRET=secret";
```

Note that **SITENAME** and **DOMAINNAME** must be consistent throughout all the modules.

2 Modules

Deploy/Redeploy the modules below.

2.1 CloudFrontDirectoryIndexFunction

Implements the function that intercepts paths in the URL and adds index.html when necessary.

2.1.1 README.md

CloudFrontDirectoryIndexFunction

Small LamdaAtEdge function that checks for a directory path and injects a *index.html* if detected.

nodejs/ contains the actual code.
CloudFormation/ contains the CF template to deploy the function into AWS.
.circleci/config.yaml contains the deployment steps.

Deployment steps

To deploy this function, define a *.deploy.env* file in the root directory
and export the following environment variables:

```
export "LAMBDA_S3_BUCKET_NAME=<S3 bucket name>";
export "FUNCTION_NAME=<Lambda Function Name>";
export "CLOUDFRONT_SITE_NAME=<HOST portion of FQDN>";
export "ROUTE53_DOMAIN_NAME=<Domain name portion of FQDN>";
```

In order to deploy, install docker from https://www.docker.com/products/docker-desktop.

Then install CircleCI CLI from https://circleci.com/docs/2.0/local-cli-getting-started/#section=getting-started

Once the *.deploy.env* is defined, simply run the localci.sh file using
bash to deploy and redeploy.

A stack named [site_name]-[domain_name]-[function_name]-[branch_name] will become available inside your AWS Console CloudFormation dashboard.

2.2 CloudFrontOAuthCognito

Implements the Lambda@Edge function to provide OAuth2 functionality to the website.

2.2.1 README.md

CloudFrontOAuthCognito

LamdaAtEdge function performs OAuth2 authorization code flow.

nodejs/ contains the actual code.
CloudFormation/ contains the CF template to deploy the function into AWS.
.circleci/config.yaml contains the deployment steps.

Deployment steps

To deploy this function, define a *.deploy.env* file in the root directory
and export the following environment variables:

```
export "LAMBDA_S3_BUCKET_NAME=<S3 bucket name to contain function>";
export "FUNCTION_NAME=<Lambda Function Name>";
export "CLOUDFRONT_SITE_NAME=<HOST portion of FQDN>";
export "ROUTE53_DOMAIN_NAME=<Domain name portion of FQDN>";
export "COGNITO_USERPOOL_ID=<cognito user pool ID";
export "COGNITO_CLIENT_ID=<cognito provided client ID>";
export "COGNITO_DOMAIN=https://<Cognito pool url>";
export "COGNITO_CLIENT_SECRET=<cognito provided client secret>";
export "GITHUB_CLIENT_ID=<github provided client id>";
export "GITHUB_CLIENT_SECRET=<github provided client secret>";
export "JWT_SIGNATURE_SECRET=<somesecretyoudecide>";
```

In order to deploy, install docker from https://www.docker.com/products/docker-desktop.

Then install CircleCI CLI from https://circleci.com/docs/2.0/local-cli-getting-started/#section=getting-started

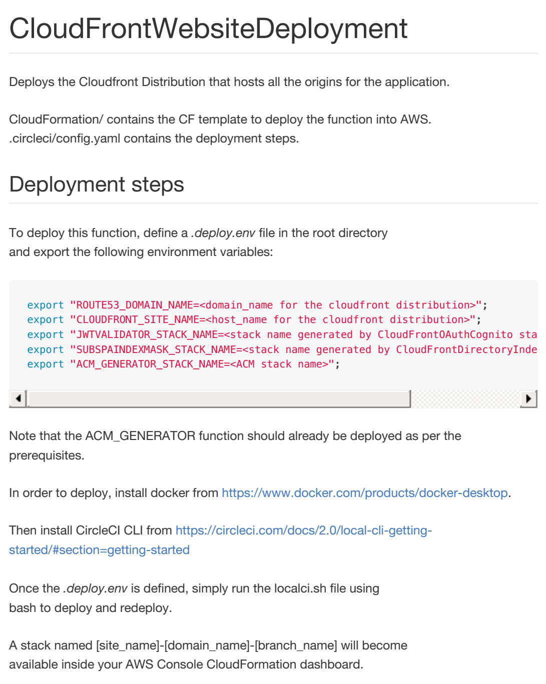
Once the *.deploy.env* is defined, simply run the localci.sh file using
bash to deploy and redeploy.

A stack named [site_name]-[domain_name]-[function_name]-[branch_name] will become
available inside your AWS Console CloudFormation dashboard.

2.3 CloudFrontWebsiteDeployment

Deploys the cloudfront components to add this application to the customer's portal.

2.3.1 README.md



CloudFrontWebsiteDeployment

Deploys the Cloudfront Distribution that hosts all the origins for the application.

CloudFormation/ contains the CF template to deploy the function into AWS.
.circleci/config.yaml contains the deployment steps.

Deployment steps

To deploy this function, define a *.deploy.env* file in the root directory
and export the following environment variables:

```
export "ROUTE53_DOMAIN_NAME=<domain_name for the cloudfront distribution>";
export "CLOUDFRONT_SITE_NAME=<host_name for the cloudfront distribution>";
export "JWTVALIDATOR_STACK_NAME=<stack name generated by CloudFrontOAuthCognito sta
export "SUBSPAINDEXMASK_STACK_NAME=<stack name generated by CloudFrontDirectoryInde
export "ACM_GENERATOR_STACK_NAME=<ACM stack name>";
```

Note that the ACM_GENERATOR function should already be deployed as per the prerequisites.

In order to deploy, install docker from https://www.docker.com/products/docker-desktop.

Then install CircleCI CLI from https://circleci.com/docs/2.0/local-cli-getting-started/#section=getting-started

Once the *.deploy.env* is defined, simply run the localci.sh file using bash to deploy and redeploy.

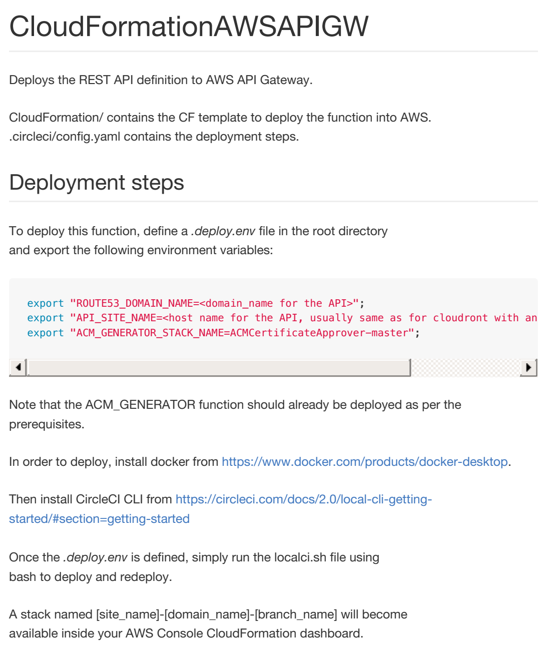
A stack named [site_name]-[domain_name]-[branch_name] will become available inside your AWS Console CloudFormation dashboard.

8

2.4 CloudFormationAWSAPIGW

Defines the REST API to host our REST function

2.4.1 README.md



CloudFormationAWSAPIGW

Deploys the REST API definition to AWS API Gateway.

CloudFormation/ contains the CF template to deploy the function into AWS.
.circleci/config.yaml contains the deployment steps.

Deployment steps

To deploy this function, define a *.deploy.env* file in the root directory
and export the following environment variables:

```
export "ROUTE53_DOMAIN_NAME=<domain_name for the API>";
export "API_SITE_NAME=<host name for the API, usually same as for cloudront with an
export "ACM_GENERATOR_STACK_NAME=ACMCertificateApprover-master";
```

Note that the ACM_GENERATOR function should already be deployed as per the prerequisites.

In order to deploy, install docker from https://www.docker.com/products/docker-desktop.

Then install CircleCI CLI from https://circleci.com/docs/2.0/local-cli-getting-started/#section=getting-started

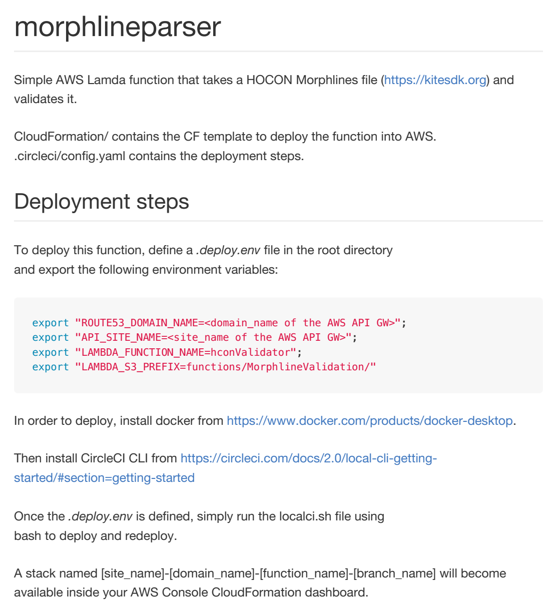
Once the *.deploy.env* is defined, simply run the localci.sh file using
bash to deploy and redeploy.

A stack named [site_name]-[domain_name]-[branch_name] will become
available inside your AWS Console CloudFormation dashboard.

2.5 morphlineparser

REST function to validate Morphlines files.

2.5.1 README.md



morphlineparser

Simple AWS Lamda function that takes a HOCON Morphlines file (https://kitesdk.org) and validates it.

CloudFormation/ contains the CF template to deploy the function into AWS.
.circleci/config.yaml contains the deployment steps.

Deployment steps

To deploy this function, define a *.deploy.env* file in the root directory
and export the following environment variables:

```
export "ROUTE53_DOMAIN_NAME=<domain_name of the AWS API GW>";
export "API_SITE_NAME=<site_name of the AWS API GW>";
export "LAMBDA_FUNCTION_NAME=hconValidator";
export "LAMBDA_S3_PREFIX=functions/MorphlineValidation/"
```

In order to deploy, install docker from https://www.docker.com/products/docker-desktop.

Then install CircleCI CLI from https://circleci.com/docs/2.0/local-cli-getting-started/#section=getting-started

Once the *.deploy.env* is defined, simply run the localci.sh file using
bash to deploy and redeploy.

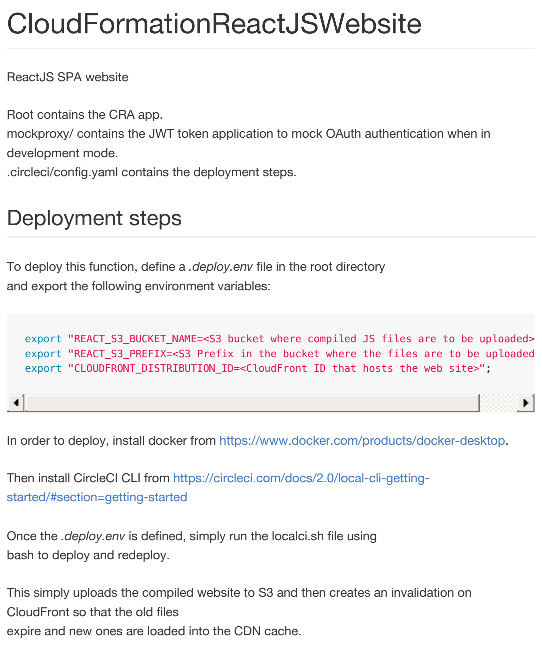
A stack named [site_name]-[domain_name]-[function_name]-[branch_name] will become available inside your AWS Console CloudFormation dashboard.

12

2.6 CloudFormationReactJSWebsite

This project only uploads the website itself to S3. The S3 bucket, prefix and CloudfrontID must be retrieved from the prerequisites in section 1.

2.6.1 README.md



CloudFormationReactJSWebsite

ReactJS SPA website

Root contains the CRA app.
mockproxy/ contains the JWT token application to mock OAuth authentication when in development mode.
.circleci/config.yaml contains the deployment steps.

Deployment steps

To deploy this function, define a *.deploy.env* file in the root directory
and export the following environment variables:

```
export "REACT_S3_BUCKET_NAME=<S3 bucket where compiled JS files are to be uploaded>
export "REACT_S3_PREFIX=<S3 Prefix in the bucket where the files are to be uploaded
export "CLOUDFRONT_DISTRIBUTION_ID=<CloudFront ID that hosts the web site>";
```

In order to deploy, install docker from https://www.docker.com/products/docker-desktop.

Then install CircleCI CLI from https://circleci.com/docs/2.0/local-cli-getting-started/#section=getting-started

Once the *.deploy.env* is defined, simply run the localci.sh file using
bash to deploy and redeploy.

This simply uploads the compiled website to S3 and then creates an invalidation on CloudFront so that the old files
expire and new ones are loaded into the CDN cache.

List of Figures

List of Tables

Listings

References

AltexSoft (2018). *Angular vs React Compared.* URL: https://www.altexsoft.com/blog/engineering/react-vs-angular-compared-which-one-suits-your-project-better/ (visited on 10/30/2018).

Amazon AWS (Apr. 2019). *Use CloudFront Web Distribution To Serve Content From Multiple Origins*. Amazon AWS. URL: https://aws.amazon.com/premiumsupport/knowledge-center/cloudfront-distribution-serve-content/ (visited on 04/05/2019).

Balog, Noemi (2019). *React vs. Angular*. URL: https://www.codingdojo.com/blog/react-vs-angular (visited on 04/17/2019).

Chacon, Scott and Ben Straub (2019-11-01). *Pro Git*. Second Edition. 2.1.174. APress. URL: https://git-scm.com/book/en/v2/.

Fernando, Ashan (Apr. 2018). *5 Tips to SpeedUp Serverless Web Apps in AWS*. URL: https://hackernoon.com/speedup-serverless-web-apps-in-aws-12fa25b94600.

Github Inc. (2019). *octokit/rest.js*. URL: https://octokit.github.io/rest.js (visited on 11/11/2019).

Hamedani, Mosh (2018). *November 5th, 2018 Comments React vs. Angular: The Complete Comparison*. URL: https://programmingwithmosh.com/react/react-vs-angular/ (visited on 11/05/2018).

Kent, Karen and Murugiah Souppaya (2006). *Guide to Computer Security Log Management*. Tech. rep. NIST. URL: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-92.pdf.

Macquin, Goldy Benedict (2018). *Single Page Applications vs Multiple Page Applications — Do You Really Need an SPA?* URL: https://medium.com/@goldybenedict/single-page-applications-vs-multiple-page-applications-do-you-really-need-an-spa-cf60825232a3 (visited on 03/21/2018).

Mannion, Mike and Barry Keepence (Apr. 1995). “SMART Requirements”. In: *SIGSOFT Softw. Eng. Notes* 20.2, pp. 42–47. ISSN: 0163-5948. DOI: 10.1145/224155.224157. URL: http://doi.acm.org/10.1145/224155.224157.

MDN Contributors (Oct. 2019a). *JavaScript*. URL: https://developer.mozilla.org/en-US/docs/Learn/JavaScript.

– (May 2019b). *Server-side website programming*. URL: https://developer.mozilla.org/en-US/docs/Learn/Server-side.

– (Aug. 2019c). *Using Web Workers*. URL: https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage.

Patel, Ronak Patel Ronak Patel Ronak (2019). *Challenging the Traditional Web Development — Top JavaScript Trends 2019*. URL: `https://medium.com/@ronak8036/top-javascript-trends-2019-8b0c0d2fac3e` (visited on 07/02/2019).

Glossary

Amazon Amplify A mobile and web app library customized to integrate with other AWS services. 9, 10

`https://aws.amazon.com/amplify/`

Amazon API Gateway A web-api hosting gateway that is part of Amazon's AWS offering. 7, 15, 17, 18, 29, 59, 70, 148

`https://aws.amazon.com/api-gateway/`

Amazon AWS A cloud-based computing platform provided by Amazon Inc.. 1, 7, 9, 12, 15–17, 26, 147

`https://aws.amazon.com`

Amazon CloudFormation provides a common language for you to describe and provision all the infrastructure resources in your cloud environment.. 16, 17, 26, 28, 59, 124, 148

`https://aws.amazon.com/cloudformation/`

Amazon Cloudfront A content distribution network based on Amazon S3 storage. 1, 7, 15–19, 28–30, 59, 67, 146, 148

`https://aws.amazon.com/cloudfront/`

Amazon Cognito A web-based identity provider that is part of Amazon's AWS offering. 16, 28, 29

`https://aws.amazon.com/cognito/`

Amazon Lambda A serverless function execution environment provided as part of Amazon's AWS offering. 17, 26, 30, 59, 126, 148

`https://aws.amazon.com/lambda/`

Amazon Lambda@Edge Lambda@Edge lets you run Node.js and Python Lambda functions to customize content that CloudFront delivers, executing the functions in AWS locations closer to the viewer.. 15, 17, 18, 28, 30

`https://docs.aws.amazon.com/lambda/latest/dg/lambda-edge.html`

Amazon S3 Cloud based Simple Storage System provided as part of Amazon's AWS offering. 7, 15, 16, 59

`https://aws.amazon.com/s3/`

CircleCI Online CI/CD hosting provider. 26, 28, 121, 148

`https://circleci.com/`

ElasticSearch A popular open source log centralization technology product. 45

`https://www.elastic.co`

ExpressJS Fast, unopinionated, minimalist web framework for Node.js. 39

`https://expressjs.com/`

Facebook A social networking pioneer and technology company. 8

`https://facebook.com/`

git A version control suite developed to manage the Linux kernel by Linus Torvalds. 4, 12, 20, 28, 30, 32, 42–44

`https://git-scm.com/`

GitHub Web-based version control system based on Git. 4, 7, 11, 12, 14–18, 20, 25, 28–32, 35, 37, 39, 40, 42, 59, 60, 67, 70, 72, 73, 78, 130, 146, 148

`https://www.github.com`

Google A cloud computing company providing web-based identity services. 8

`https://www.google.com`

GraphQL GraphQL is a query language for APIs and a runtime for fulfilling those queries with your existing data. 11

`https://graphql.org/`

Java Object oriented programming language and virtual machine platform supplied by Oracle Inc.. 4

`https://www.java.com`

JavaScript Object oriented programming language based on the Java Syntax. 7, 9, 11, 28–30, 32, 33

`https://www.ecma-international.org/publications/standards/Ecma-262.htm`

Monaco Editor Text editor component provided by Microsoft and used as the base for Visual Studio. 23

`https://microsoft.github.io/monaco-editor/api/index.html`

Morphlines Library and DSL providing ETL functionality used by Kite SDK and Field Extraction Library. 13, 21, 23, 45

`http://kitesdk.org/docs/current/kite-morphlines/index.html`

Node Package Manager Package manager used by node to install dependencies. 10, 33

`https://npm.org/`

NodeJS a JavaScript runtime built on Chrome's V8 JavaScript engine.. 15, 30, 35, 39

`https://nodejs.org/en/`

OAuth2 A web-based authentication protocol standard. 1, 12, 15, 16, 18, 28, 29, 35, 36, 39, 40, 59, 72, 146, 148

`https://tools.ietf.org/html/rfc8252`

Octokit NodeJS wrapper around the Github API. 11, 13, 29, 30, 32, 44

`https://octokit.github.io/rest.js/`

Open Source software for which the original source code is made freely available and may be redistributed and modified. 4

`https://www.lexico.com/en/definition/open-source`

PUG Templating language for HTML. 39

`https://pugjs.org/api/getting-started.html`

Python Interpreted programming language. 15

`https://www.python.org`

React Bootstrap ReactJS version of the popular bootstrap CSS library offered by Twitter. 25

`https://react-bootstrap.github.io/`

ReactJS A web application technology provided by Facebook. 9, 11, 17, 18, 39, 41, 59, 73, 146, 148

`https://reactjs.org/`

REST Representational State Transfer. 11, 30

`https://tools.ietf.org/id/draft-griffin-bliss-rest-00.html`

serverless A cloud-computing execution model in which the cloud provider runs the server, and dynamically manages the allocation of machine resources. 7, 12, 15, 16, 26

`https://en.wikipedia.org/wiki/Serverless_computing`

SHA-1 Secure Hash Algorithm 1, a having algorithm that produces a 160-bit hash. 12

`http://dx.doi.org/10.6028/NIST.FIPS.180-4`

Svelte Compiled SPA framework. 9

`https://svelte.dev/`

TypeScript a typed superset of JavaScript that compiles into plain JavaScript. 9

`https://www.typescriptlang.org/`

VueJS The Progressive JavaScript Framework. 9

`https://vuejs.org/`

Acronyms

API Application Programming Interface. 11, 13, 30

CI/CD Continuous Integration/Continuous Delivery. 26

ETL Extraction, Transformation and Loading. 4, 42, 45

HTTP Hyper Text Transfer Protocol. 11, 30

IDE Integrated Development Environment. 20

IS Information Security. 4

JWT Javascript Web Token. 29

MPA Multi Page Application. 7, 8, 16, 25

MVP Minimum Viable Product. 17, 43, 44

SMART Specific, Measurable, Attainable, Realisable, Traceable. 6

SPA Single Page Application. 7, 8, 11, 15, 16, 25

SWOT Strength, Weaknesses, Opportunities, Threat. 7, 8, 11

WYSIWYG What You See Is What You Get. 45