caMicroscope Documentation

Release latest

Apr 11, 2019

Contents

1 nanoBorb **3**

2 Hosted Setup **5**
- 2.1 SSL . . . 5
- 2.2 Component Services . . . 5
- 2.3 Configuration . . . 6
- 2.4 Security . . . 6
- 2.5 PathDB . . . 7

3 Developer Guide **9**

caMicroscope is a tool to is a HTML5 image viewer optimized for large bio-medical image data viewing, with a strong emphasis on cancer pathology. This guide has sections for different kinds of use of the platform. “User Guide” covers the basics on how to use caMicroscope viewer. “Nanoborb” covers nanoBorb, the version of caMicroscope designed as a standalone application for individual users without a server. “Hosted Setup” covers how to set up caMicroscope for multiple users on a server. “Developer Guide” covers the broad strokes on how to add new functionality to caMicroscope.

UNDER CONSTRUCTION

CHAPTER 1

nanoBorb

To use caMicroscope as a standalone application, see nanoborb. The user guide screencast should explain the basics of Nanoborb.

Installation Instructions

Windows - 1) Download zip file. 2) Unzip. 3) Run “nanoborb.exe” in the unzipped folder

MacOS - 1) Download zip file. 2) Unzip. 3) Move Nanoborb app to Applications folder. 4) Double-click copied Nanoborb file to run.

CHAPTER 2

Hosted Setup

The full distribution repository for hosted caMicroscope is here. run with `docker-compose -f caMicroscope.yml up`

this will build all services and run in the foreground. Use `docker-compose -f caMicroscope.yml build` to rebuild the services.

Once everything is up, go to :4010/ to see the landing page.

2.1 SSL

To enable ssl, mount the private key and certificate files to elevate in /root/src/ssl/privatekey.pem and /root/src/ssl/certificate.pem respectively. HTTPS mode will only be enabled if both of these files are present.

2.2 Component Services

mongo - vanilla mongo container

idxMongo - ephemeral container to index mongo (that is, this container is *expected* to exit once it's done its job)

bindaas - api service for mongo (see https://github.com/sharmalab/bindaas)

iip - slide tile server (see https://github.com/camicroscope/iipImage)

viewer - hosts the viewer files and builds packages (see https://github.com/camicroscope/caMicroscope)

loader - extracts metadata needed for image loading (see https://github.com/camicroscope/SlideLoader)

elevate - security proxy (see https://github.com/camicroscope/Security)

auth - consumes external JWT and issues caMicroscope JWT (see Security Section)

2.3 Configuration

Logging - Container Logging is, for HIPAA reasons, disabled. Feel free to use a different logging engine if desired, especially for development.

Routes - This is handled by the elevate service/ca-security container. By default routes go the viewer, unless a specific pattern in routes.json is matched.

Image Volume - This is, by default, the images directory in this directory. If this is changed, please make the same change across all impacted services.

Packages - Packages are built in the viewer service using parcel, mount a different directory with packages.js to the package directory to overwrite or add functionality.

2.4 Security

For standard security, we use a combination of an external identity provider and an internal authorization service. The authorization service consumes Json Web Tokens (JWTs) from the identity provider, and then will issue JWTs which convey both authentication and authorization, which are consumed by the application.

2.4.1 Getting an Identity Provider and Setting up Login

There are many identity providers, but for testing and examples, we have been using auth0.

When selecting, an identity provider, note that we expect it to provide a JWT, and to have a certificate/public key/secret which can be used to verify such JWTs.

The example given in the Distro within config/login.html is set up for auth0; simply change the corresponding variables for your auth0 application if auth0 is used. If using another identity provider, then login.html, or equivalent, needs to, at least, set the JWT to a cookie called "token", and call the auth service's 'check' route, and save a successful result as the token. Follow the guide which your identity provider uses for further guidance.

2.4.2 Keys/Certificates

Add the following files; by default, they are mounted:

- –./jwt_keys/certificate or ./jwt_keys/jwk.json is the certificate/public key/secret or jwk (respectively) from the **identity provider**. (If both are included, the jwk takes precedence).
- –./jwt_keys/key and ./jwt_keys/key.pub are used as the signing and check keys for the **auth service**
- –These can (and should) be generated with ./kwt_keys/make_keys.sh

2.4.3 Deployment Configuration

Turn off disable security under the elevate service to block routes.

2.4.4 Adding Users to Database

Add users as in ./config/add_mongo_users.js. Attributes can be added to deny access to routes (e.g. allow only some users to post and delete)

The name field is the email field (or failing that, sub field) in that priority from the identity provider.

2.5 PathDB

To use pathdb, use pathDbCamic.yml instead of caMicroscope.yml, and replace routes.json with pathdb_routes.json. This deployment does not include the auth and loader as separate services, as this PathDB provides that functionality.

CHAPTER 3

Developer Guide

We are collecting feedback to write this section in more detail. Please give any feedback to this form.

caMicroscope is designed to be extended. Depending on the manner of extension, it may be a package or a new application. It is highly recommended to use a minimal methods of extension to avoid code duplication and issues with future upgrade compatibility. Changing Package on default viewer The viewer container builds package/package.js on build, and the resulting file is included in the viewer. This is used to add small custom functionality tweaks to the viewer. Adding more applications alongside the default viewer For more substantial modifications, it may make more sense to make a new application. These can be mounted somewhere (preferably somewhere in the apps folder) in the viewer service, and would include modification of the toolbar, or implementation of an entirely new functionality

caMicroscope is open source software. Any involvement and contribution with the caMicroscope project is greatly appreciated. Feel free to get directly involved in any of the repositories in the caMicroscope organization. The caMicroscope project has persistent branches for development and release. It is highly recommended to make any changes off of the develop branch of a repository, and, when ready, create a PR to the develop branch of the source repository. The creation of additional specializations, packages, or applications should, in most cases, live as a separate repository.