AVERT C2 Development Environment Setup

October 18, 2021

Overview

The AVERT C2 application consists of NodeJS back-end services and ReactJS user interface applications. Each application in the AVERT C2 ecosystem consists of a back-end service and optionally, a client user interface. The core of AVERT C2 consists of the ecosystem application, which is a back-end service that supports most of the functionality across all AVERT C2 applications. Most real-time data is stored in a RethinkDB database and historical data is stored in an Elasticsearch database.

Prerequisites

Hardware

- Recommend minimum 16GB RAM (32 GB Preferred), At least equivalent of I7 CPU with 8 cores

Software

- Windows 10 Pro
- Visual Studio Code – For development
- NodeJS version 10.24.1 https://nodejs.org/dist/v10.24.1/node-v10.24.1-x64.msi
- Docker Desktop https://www.docker.com/products/docker-desktop – configure for windows subsystem for Linux

Repositories

All the repositories for AVERT C2 can be accessed from the bitbucket project at:

http://bitbucket.aresseccorp.com/projects/COM

The repositories you'll need to clone will depend on the development task at hand, but the following will typically always be required: app-gateway, client-app-core, commandbridge-compose, ecosystem, elasticsearch, integration-app, node-app-core, and orion-components.

Jenkins - Continuous Integration

All projects are setup in Jenkins to handle build and deployments. These build the applications from using a Jenkinsfile and the project Dockerfile. Access will be provided so that developers can troubleshoot build issues and/or trigger new builds as needed.

RethinkDB Setup

When the ecosystem service first starts up, it will run migration scripts to populate initial data. To view the data in RethinkDB, go to http://localhost:8080 in your browser. When prompted for credentials, use admin as the username and P@b6112759 as the password.

There is an open-source GUI for RethinkDB databases called ReQLPro that you may find useful. If you can't find the download online, we can provide an installer.

Elasticsearch Setup

When the ecosystem service first starts up, it will run migration scripts to populate initial data. You can use Kibana to view the data in Elasticsearch by going to http://localhost:5601 in your browser. When prompted for credentials, use elastic as the username and changeme as the password.

Running Locally***

Docker Stack

In the commandbridge-compose repository, there is a docker-compose.yml file that contains the Docker setup needed for development. The easiest way to get started is to open a command prompt, navigate to the commandbridge-compose repository, and enter `docker-compose up -d`.

Keep in mind that this file will have all containers configured, even if you don't need all of them for what you're currently working on. If you know a container won't be necessary for what you're working on, feel free to remove it from your local docker-comose.yml file so it is not used.

Another thing to note is that the first time running this command will take a bit of time to run because it is having to build the Docker images for the first time. If you are working with an AVERT C2 application that has a client, you will need to build that manually.

Client Development

If an AVERT C2 application has a client, that code can be found in the app-src directory of its repository. To build the application locally, you'll need to run `npm install` followed by `npm run build`, both in the app-src directory. This will compile the client code and put it in an app directory in the repository.

When working client code, it is easiest to run `npm run start` in the app-src directory of the repository. This will create a Webpack dev server that will run on port 3000 and rebuild the client whenever you make a change to a file and save it. Navigate to http://localhost:3000 in your browser to see the application and anytime the client code is modified, the browser will automatically refresh to show the changes.

Dependency Development

Node-app-core

You'll notice in the docker-compose.yml file in the commandbridge-compose repository, there are volume mappings for the node-app-core repository. This makes it easier for doing node-app-core development. When doing node-app-core development, simply change the main property value in its package.json file from ./dist/index to ./lib/index. Do NOT commit this change to the package.json file.

Since node-app-core is a dependency repository and not an application or service, you'll need to test your changes in another repository. You will need to restart the corresponding container for it to reflect the changes in node-app-core.

Orion-components

Since orion-components is used by the AVERT C2 applications with clients, changes need to be compiled with the application's code. The best way to do this is to run `npm run prelink` followed by `npm link` in the orion-components repository. Then, in the app-src directory of the client application's repository, run `npm link orion-components`. This will create a link so that changes to the orion-components repository will be reflected in the client applications node_modules directory.

Once you make a change in the orion-components repository, you'll need to run `npm prepublish` for the code to be compiled.

The prelink command you ran before setting up the link actually uninstalls the react, react-dom, moment-timezone, and react-mapbox-gl dependencies. Once you are done changing code and are ready to commit your changes, you'll need to revert those changes to the package.json file, run `npm install` followed by `npm prepublish`. Once this is done, you can add the changes and commit them.

Client-app-core

The easiest way to develop in the client-app-core repository is to make your changes, run `npm run build:node`, then copy the ./dist/client-app-core.js file from the client-app-core repository to the ./app-src/node_modules/client-app-core/dist directory in the AVERT C2 application repository. Once the file has been copied, you'll need to rebuild the client (unless running the Webpack dev server).

If the change you are making will be used in orion-components, you'll need to follow the same steps, but also rebuild orion-components.

Default Credentials

RethinkDB

URL: http://localhost:8080
Username: admin
Password: P@b6112759

Kibana

URL: http://localhost:5601
Username: elastic
Password: changeme

Minio

URL: http://localhost:9001
Username: SOME_ACCESS_KEY
Password: SOME_SECRET

AVERT C2

URL: https://localhost
Org Admin Username: orgadmin@aressecuritycorp.com
Org Admin Password: P@ssw0rd
Eco Admin Username: ecoadmin@aressecuritycorp.com
Eco Admin Password: P@ssw0rd