Cargoplane Documentation

Release 1.1.0

Onica Group

Feb 26, 2020

Contents


cargoplane
aws

CHAPTER 1

What is this?

Cargoplane is a toolset to help you quickly *transport message cargo* between webapp clients and a backend running in the AWS cloud.

Unlike other solutions, this one does not rely on a 3rd party (just you and AWS) and is entirely serverless. Also unlike some alternatives, you application controls what topics each client has access to subscribe to and which topics they may publish to.

Cargoplane is written in Typescript, but transpiled to Javascript. The Lambda code is compatible with Node.js 8 and 10. The client code is ES5, and so will work in any remotely modern browser.

This repository is developed and maintained by the Onica Cloud Native Development Practice.

This collection is part of Onica's commitment to give back to the open source community. Find this and other Onica open source repositories on GitHub.

CHAPTER 2

Example Uses

Chat The classic example for this is a chat ability between web site visitors and company support. In fact, a simple version of this serves as the *demo*.

Push notifications Web app users can be notified of events that have occurred in the cloud.

Data Refresh The classic problem with web apps is knowing when the data showing in the browser is outdated. Cargoplane was originally built to solve this problem. When data is changed by one user, a message can be published (by that client or by a Lambda processing the change) to subscribing clients that a change has happened. The other clients then know to refresh their content from the cloud. (Only small data changes should be sent directly through Cargoplane.)

CHAPTER 3

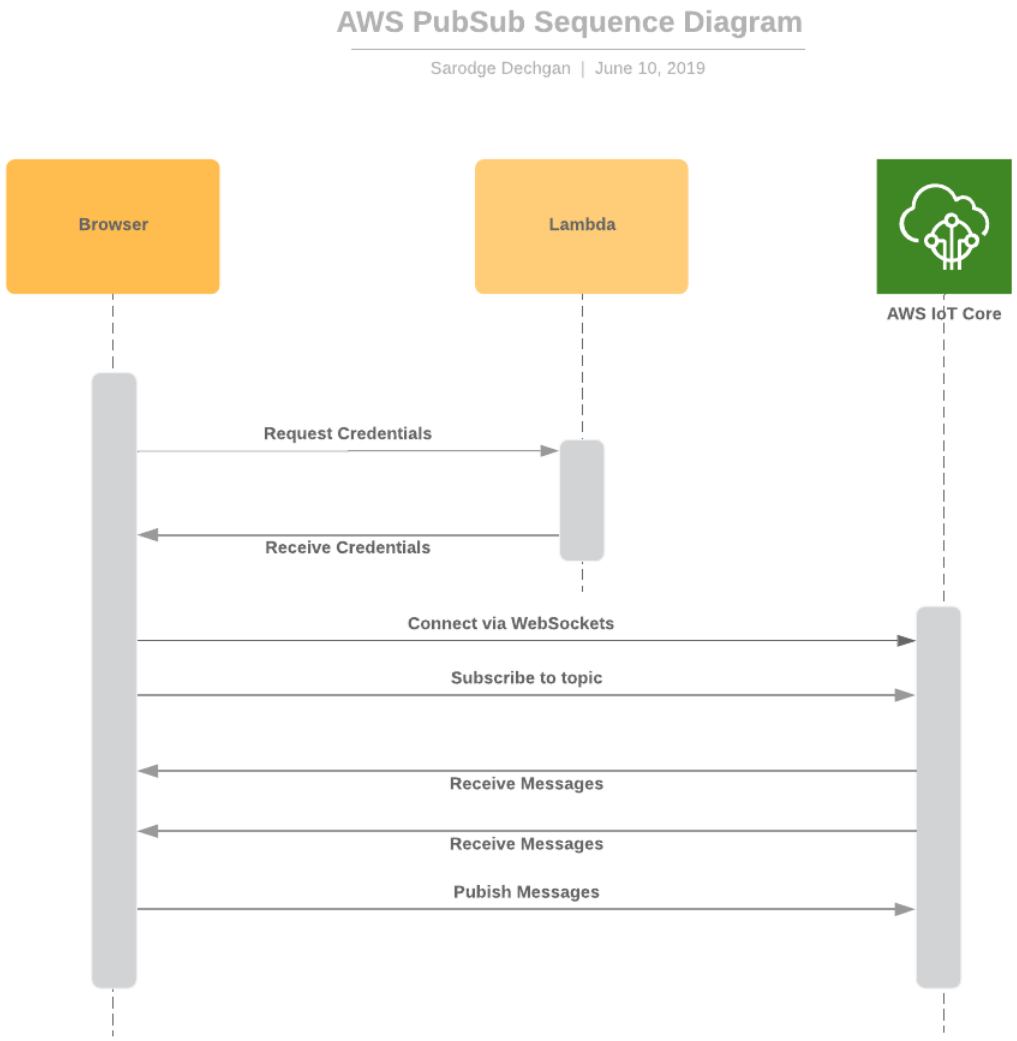
Content

3.1 Architecture and Design

3.1.1 Communication Flows

1. **Client to Client:** Web app clients can subscribe and publish to the same topics, allowing direct communications between them.
2. **Cloud to Client:** Your cloud code (ex: Lambdas) can easily publish to topics subscribed to by clients.
3. **Client to Cloud?** Generally, existing mechanisms such as API Gateway are better for this. However, it is possible to subscribe Lambdas to topics.
4. **Cloud to Cloud?** Only use Cargoplane this way if the cloud subscriber is in addition to web clients. For pure cloud to cloud, there are better options available within AWS such as SNS.

3.1.2 Design



In order to achieve the subscribe, publish, and receive capabilities there are two modules:

- cloud: Lambda to give the correct credentials.
- client: Browser client to send and receive messages.

Here are the steps:

1. Call AWS Lambda to get AWS IoT credentials and endpoint
2. Make MQTT Connection to AWS IoT with credentials

From there, you can:

1. Subscribe to topics and receive messages
2. Publish messages to topics

3.2 Cloud

Cargoplane consists of a *cloud* package and a *client* package, which must be used together in a solution.

The Cargoplane cloud package is for use in an AWS Lambda.

3.2.1 Install

```
npm i @cargoplane/cloud
```

3.2.2 Infrastructure

Cargoplane is powered by AWS IoT Core. The first time credentials are created in a region for your AWS account, the IoT endpoint is created and registered in DNS. This can take a few minute to create and still longer to DNS propagate. Rather than having your application fail on the first attempt, you can pre-create the endpoint:

1. Login to AWS Console
2. Ensure you are in the desired region
3. Navigate to the "IoT Core" Service
4. Click "Get Started" button. (If you see one.)
5. Click on "Settings" in the left nav-pane, and check on the status of your "Custom endpoint".

The cloud-side of Cargoplane can be used to publish messages, but it's primary purpose is to provide credentials for a client app. To do this, you must create an IAM role and a Lambda via API Gateway that uses this role to build credentials.

IAM Role

A role is needed to serve as the starting point for creating credentials for clients. The credentials created from this role will give clients limited access to connect, subscribe, publish and receive messages via AWS IoT. The actual credentials created for a client will be further limited by account, region, and topic.

```
IoTRole:
  Type: 'AWS::IAM::Role'
  Properties:
    RoleName: mycargoplaneapp-role-dev
    AssumeRolePolicyDocument:
      Version: 2012-10-17
      Statement:
        - Effect: Allow
          Principal:
            AWS: !Join
              - ':'
              - - 'arn:aws:iam'
                - ''
                - !Ref 'AWS::AccountId'
                - root
          Action: 'sts:AssumeRole'
    Policies:
      - PolicyName: mycargoplaneapp-role-dev-policy
        PolicyDocument:
```

(continues on next page)

(continued from previous page)

```
          Version: 2012-10-17
          Statement:
            Effect: Allow
            Action:
              - 'iot:Connect'
              - 'iot:Subscribe'
              - 'iot:Publish'
              - 'iot:Receive'
            Resource: 'arn:aws:iot:us-east-1:*:*'
```

You should of course replace the `RoleName` and `PolicyName` with appropriate names, best parameterized. Also, the final resource region needs to be changed if not `us-east-1` or to be parameterized.

Example: Cloud Demo Serverless Framework template

Lambda

Example: Cloud Demo Lambda handler

The Lambda needs the following additional policies in order to create client credentials, and to publish to topics:

```
- Effect: "Allow"
  Action:
    - iot:DescribeEndpoint
    - iot:Connect
    - iot:Publish
  Resource: "*"
- Effect: "Allow"
  Action:
    - 'sts:AssumeRole'
  Resource:
    Fn::GetAtt:
      - IoTRole
      - Arn
```

3.2.3 CargoplaneCloud class

Coding is managed by the class CargoplaneCloud. Simply create a new instance like so:

```
import {CargoplaneCloud, CargoplaneCredentialRequest} from '@cargoplane/cloud';
const cargoplane = new CargoplaneCloud();
```

Then use one of the public methods to either create credentials for a client app, or to publish a message.

createCredentials

```
async createCredentials(request:  CargoplaneCredentialRequest):
Promise<CargoplaneCredential>
```

Example Usage:

```
let credRequest: CargoplaneCredentialRequest = {
    roleName: "mycargoplaneapp-role-dev",
```

(continues on next page)

(continued from previous page)

```
    pubTopics: [
        "chattopic/general"
    ],
    subTopics: [
        "chattopic/*"
    ]
}

let credentials = await new CargoplaneCloud().createCredentials(credRequest);
```

publish

`async publish(topic: string, message?: any): Promise<void>`

Example Usage:

```
await cargoplane.publish(topic, message);
```

3.3 Client

Cargoplane consists of a *cloud* package and a *client* package, which must be used together in a solution.

This Cargoplane client package is used in your web application.

3.3.1 Install

```
npm i @cargoplane/client aws-iot-device-sdk rxjs
```

Install with Angular

If integrating with Angular, some changes are needed to make the aws-iot package, which is intended for a Node.js environment, to work in a browser.

```
npm i @angular-builders/custom-webpack --save-dev
```

Add the file `./webpack-custom.config.ts` to the project's root folder with the following:

```
module.exports = {
  node: {
    fs: 'empty',
    tls: 'empty',
    path: 'empty'
  }
};
```

Similarly, if you use the default Karma unit test running, add the following in `karma.conf.js` as another property passed to `config.set`:

```
webpack: {
  node: {
    fs: 'empty',
```

(continues on next page)

(continued from previous page)

```
    tls: 'empty',
    path: 'empty'
  }
}
```

See also AWS Javascript SDK with Angular for other possible changes that may be needed.

3.3.2 CargoplaneClient class

The `CargeplaneClient` client is your API to integrate with Cargoplane. It must be used as a singleton. (If you have dependency injection, use that.)

Please see the *demos* for examples to follow.

connect

Connects/reconnects to Cargoplane Cloud with the given credentials.

`connect(credential: CargoplaneCredential, emitEventMsBeforeExpiration?: number): Observable<Event>`

The `credential` must be retrieved from the companion Lambda in the cloud package calling `CargoplaneCloud#createCredentials`.

`connect` will return a stream of Events about the connection. Check the Event `type` field for what happened. Unless otherwise stated, Cargoplane will log these events and manage them automatically.

- `type === 'connected'`: Connection has completed.
- `type === 'disconnected'`: Connection has been dropped.
- `type === 'offline'`: Network is offline.
- `type === 'expiring'`: The current credentials are expiring (or has already). Use this to trigger your application to obtain new credentials to call `connect` with again. Subscriptions are automatically re-applied upon reconnect. You can control how early the expiration warning comes by optionally passing in a value for `emitEventMsBeforeExpiration`. The default is one minute.
- `type === 'clock-resume'`: If the computer sleeps or the browser tab is suspended, this will be emitted when processing resumes. *Messages may have been lost while suspended* - you may need to take action to account for this. If the credentials expired during the suspension, a separate `expiring` event will follow this.
- `type === 'error'`: There was an error with the connection. (Cargoplane will try to reconnect if needed.)

disconnect

Disconnect and clear all subscriptions. (Ex: upon user logout)

`disconnect(): void`

isOnline

Is the service currently connected to the cloud?

If network access is lost, it will automatically attempt to reconnect when network access is restored provided that the credentials have not expired.

```
isOnline():  boolean
```

observe

Obtain an RxJs Observable of a topic.

This call will automatically subscribe to the topic if this is the first request to observe it.

```
observe(topic:  string):  Observable<any>
```

publish

Publish a message to a topic.

```
publish(topic:  string, message?:  any):  void
```

3.4 Demo

The demo directory contains a simple chat application to show how to use Cargoplane. It consists of:

1. A Serverless Framework based cloud stack.
2. An Angular webapp client.
3. A React webapp client.

3.5 Apache License

Version 2.0

Date January 2004

URL http://www.apache.org/licenses/

3.5.1 TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means *(i)* the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or *(ii)* ownership of fifty percent (50%) or more of the outstanding shares, or *(iii)* beneficial ownership of such entity.

"You" (or **"Your"**) shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License.

Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License.

Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution.

You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

- You must give any other recipients of the Work or Derivative Works a copy of this License; and
- You must cause any modified files to carry prominent notices stating that You changed the files; and
- You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and
- If the Work includes a `"NOTICE"` text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such `NOTICE` file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a `NOTICE` text file distributed as part of the Derivative Works; within the Source form or documentation,

if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the `NOTICE` file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the `NOTICE` text from the Work, provided that such additional attribution notices cannot be construed as modifying the License. You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions.

Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks.

This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the `NOTICE` file.

7. Disclaimer of Warranty.

Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an **"AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND**, either express or implied, including, without limitation, any warranties or conditions of **TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE**. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability.

In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability.

While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

CHAPTER 4

Why Cargoplane?

Onica's early OSS releases have had aviation themed names; this may or may not have something to do with the CTO being a pilot. Nobody really knows.

Cargoplane is visualized as a means to transport (fly) cargo (information) from one point to another. There isn't much more to it than that.

Also, the name was available. That was a big factor.