Version 1.00

J12/07/21

Part 1: Detect Ships in Images with Custom Vision

In this exercise, you will use the Custom Vision service to train an *object detection* model that can detect and locate ships in a satellite image. Overview Video
Note: These exercises are built with open-source imagery. Airbus or other imagery can be used in combination with the example code and fully customized for a complete end to end solution.

Clone the repository for this course

If you have already cloned **ShipDetector** code repository to the environment where you're working on this lab, open it in Visual Studio Code; otherwise, follow these steps to clone it now.

1. Start Visual Studio Code.
2. Open the palette (SHIFT+CTRL+P) and run a **Git: Clone** command to clone the `https://github.com/Microsoft/ShipDetector` repository to a local folder (it doesn't matter which folder).
3. When the repository has been cloned, open the folder in Visual Studio Code.
4. Wait while additional files are installed to support the Python code projects in the repo.
   Note: If you are prompted to add required assets to build and debug, select **Not Now**.

Create Custom Vision resources

If you already have **Custom Vision** resources for training and prediction in your Azure subscription, you can use them in this exercise. If not, use the following instructions to create them.

1. In a new browser tab, open the Azure portal at `https://portal.azure.com`, and sign in using the Microsoft account associated with your Azure subscription.
2. Select the **+Create a resource** button, search for *custom vision*, and create a **Custom Vision** resource with the following settings:
   - **Create options**: Both
   - **Subscription**: *Your Azure subscription*
   - **Resource group**: *Choose or create a resource group (if you are using a restricted subscription, you may not have permission to create a new resource group - use the one provided)*
   - **Name**: *Enter a unique name*
   - **Training location**: *Choose any available region*
   - **Training pricing tier**: F0
   - **Prediction location**: *The same region as the training resource*
   - **Prediction pricing tier**: F0
     Note: If you already have an F0 custom vision service in your subscription, select **S0** for this one.
3. Wait for the resources to be created, and then view the deployment details and note that two Custom Vision resources are provisioned; one for training, and another for prediction. You can view these by navigating to the resource group where you created them.

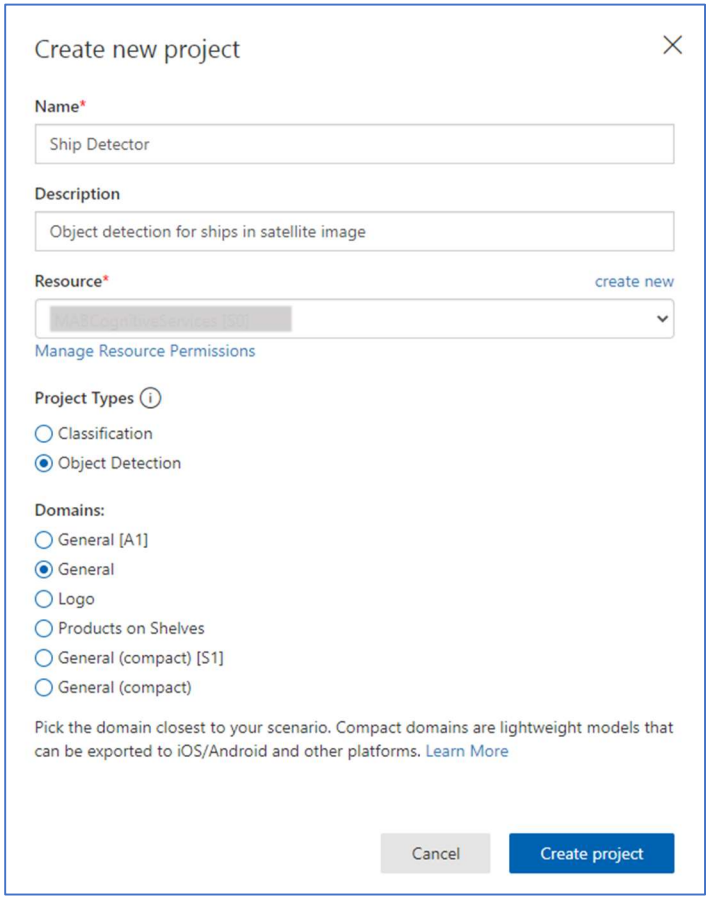
Important: Each resource has its own *endpoint* and *keys*, which are used to manage access from your code. To train an image classification model, your code must use the *training* resource (with its endpoint

and key); and to use the trained model to predict image classes, your code must use the *prediction* resource (with its endpoint and key).

Create a Custom Vision project

To train an object detection model, you need to create a Custom Vision project based on your training resource. To do this, you'll use the Custom Vision portal.

1. In a new browser tab, open the Custom Vision portal at `https://customvision.ai`, and sign in using the Microsoft account associated with your Azure subscription.
2. Create a new project with the following settings:
    - **Name**: Ship_Detector
    - **Description**: Object detection for ships in satellite image.
    - **Resource**: *The Custom Vision resource you created previously*
    - **Project Types**: Object Detection
    - **Domains**: General

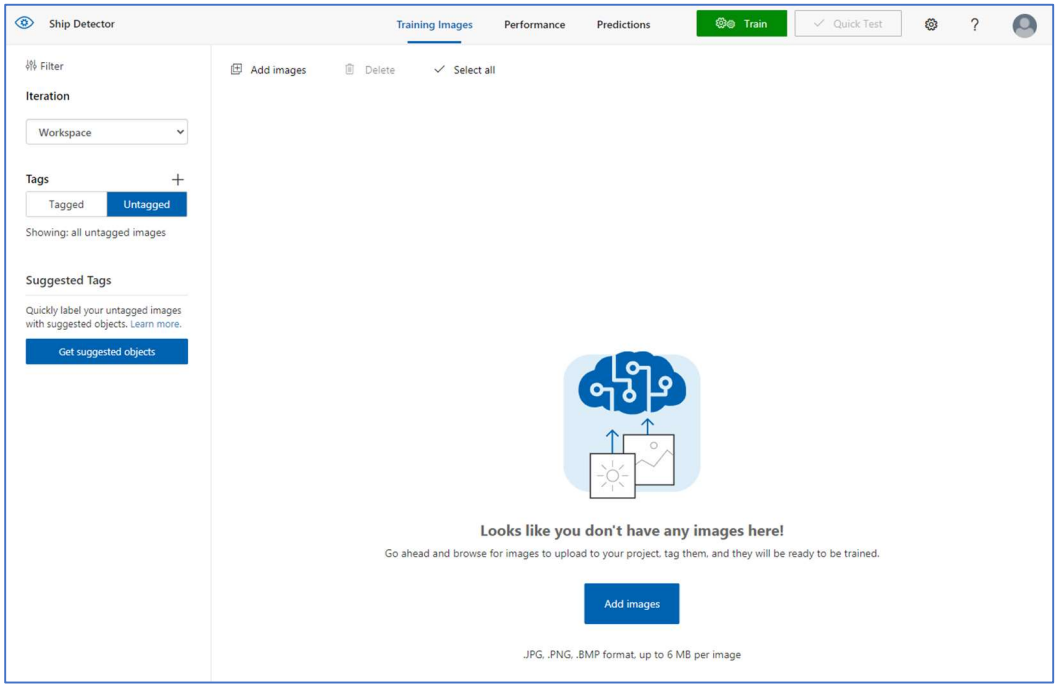


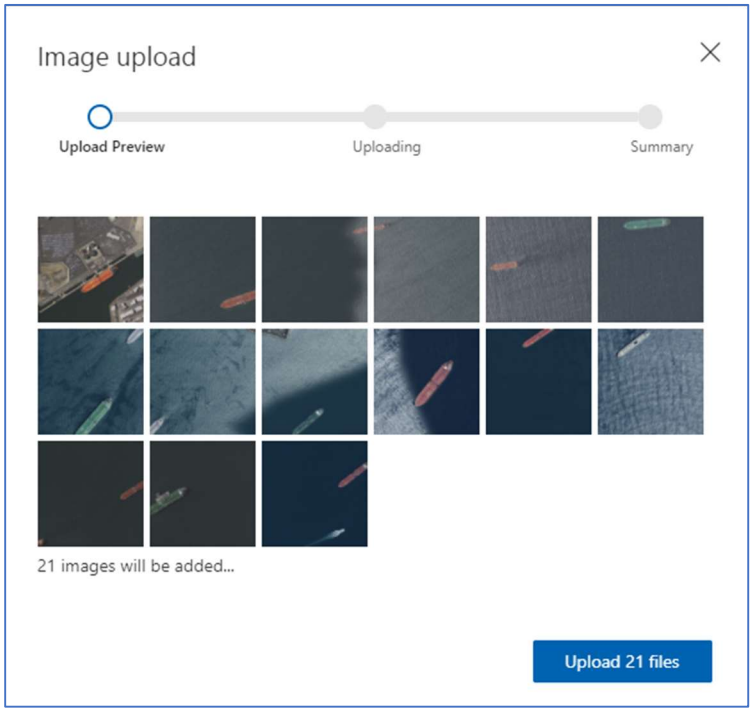
3. Wait for the project to be created and opened in the browser.

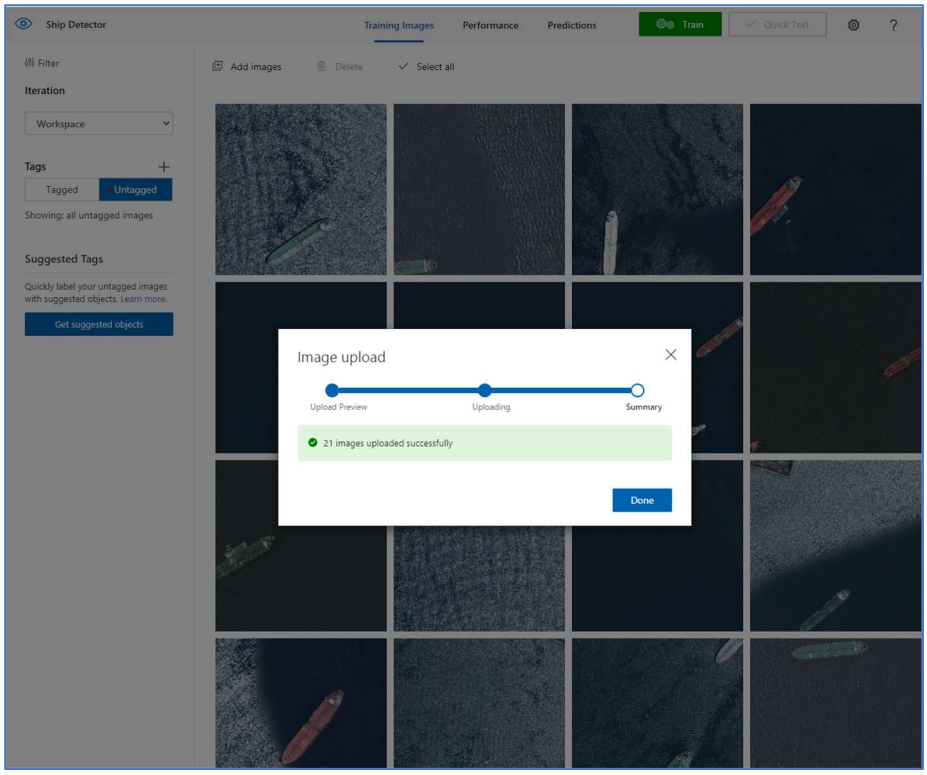
Add and tag images

To train an object detection model, you need to upload images that contain the classes you want the model to identify, and tag them to indicate bounding boxes for each object instance.

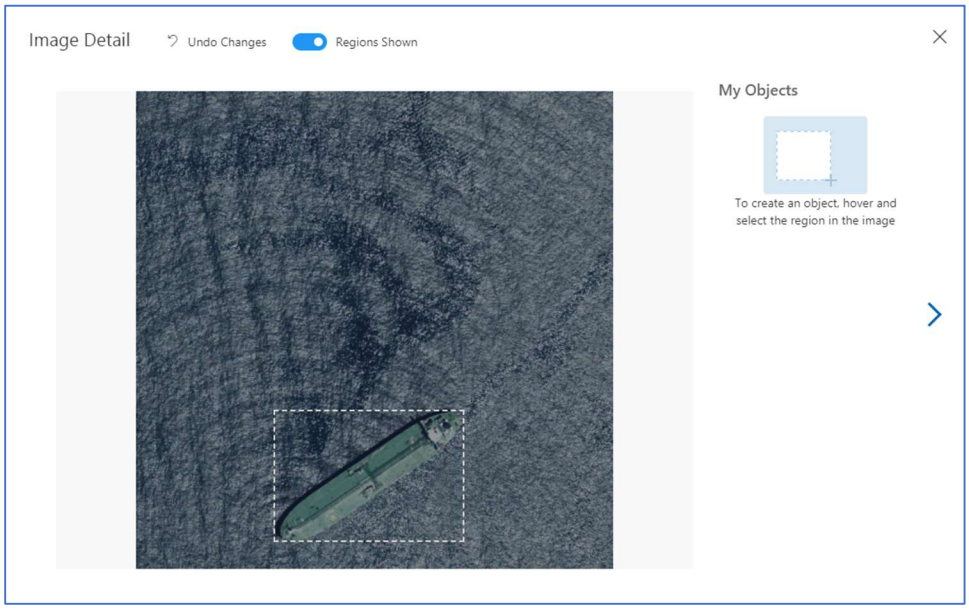
1. In Visual Studio Code, view the training images in the **/Training Images** folder where you cloned the repository. This folder contains images of ships.
2. In the Custom Vision portal, in your object detection project, select **Add images** and upload all of the images in the extracted folder.





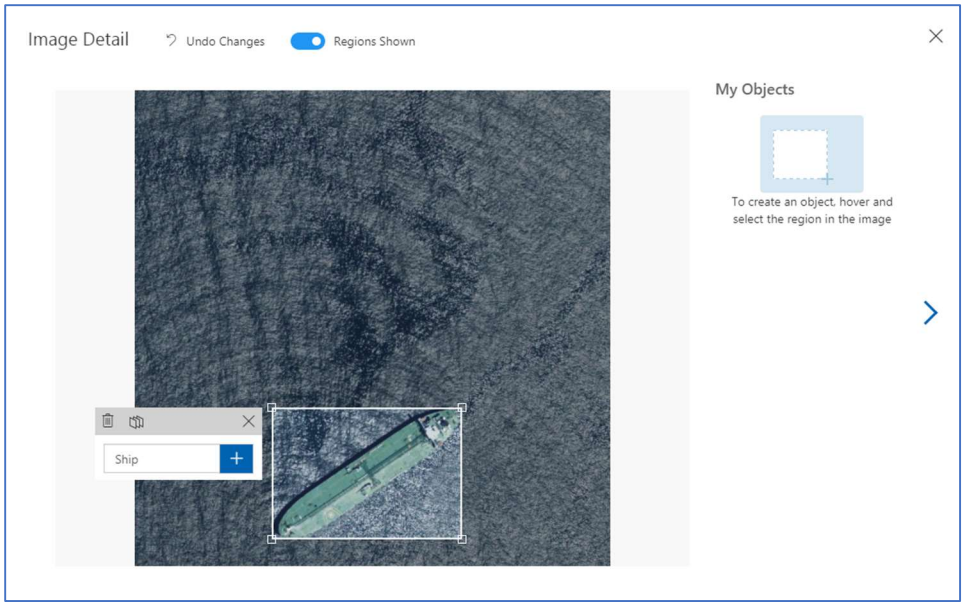


3. After the images have been uploaded, select the first one to open it.
4. Hold the mouse over any object in the image until an automatically detected region is displayed like the image below. Then select the object, and if necessary, resize the region to surround it.

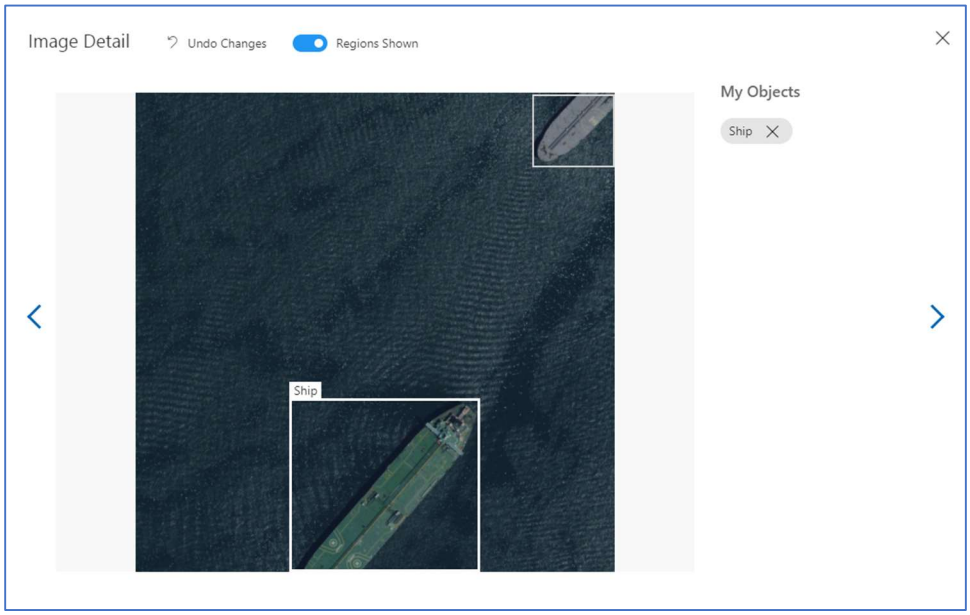


Alternatively, you can simply drag around the object to create a region.

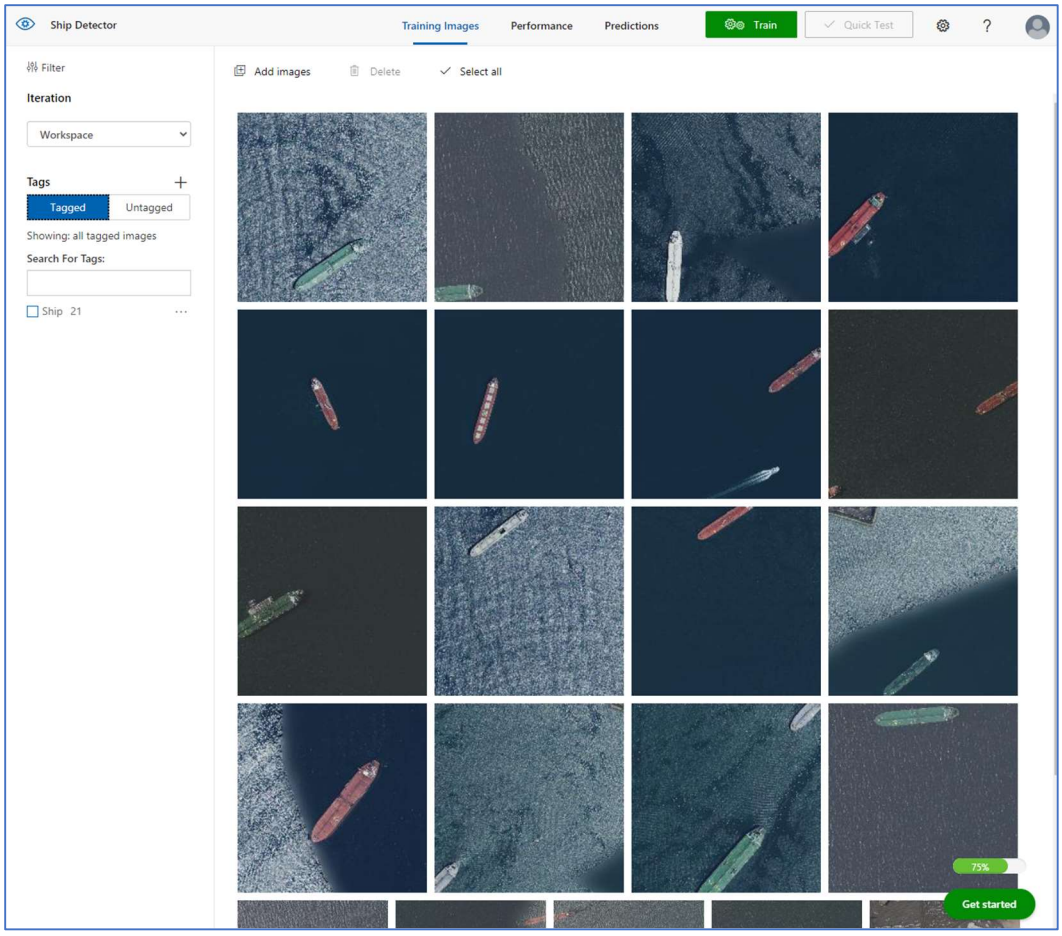
5. When the region surrounds the object, add a new tag with the appropriate object type (*Ship*) as shown here:



6. Select and tag each other object in the image, resizing the regions and adding new tags as required.



7. Use the **>** link on the right to go to the next image, and tag its objects. Then just keep working through the entire image collection, tagging each ship.
8. When you have finished tagging the last image, close the **Image Detail** editor and on the **Training Images** page, under **Tags**, select **Tagged** to see all of your tagged images:



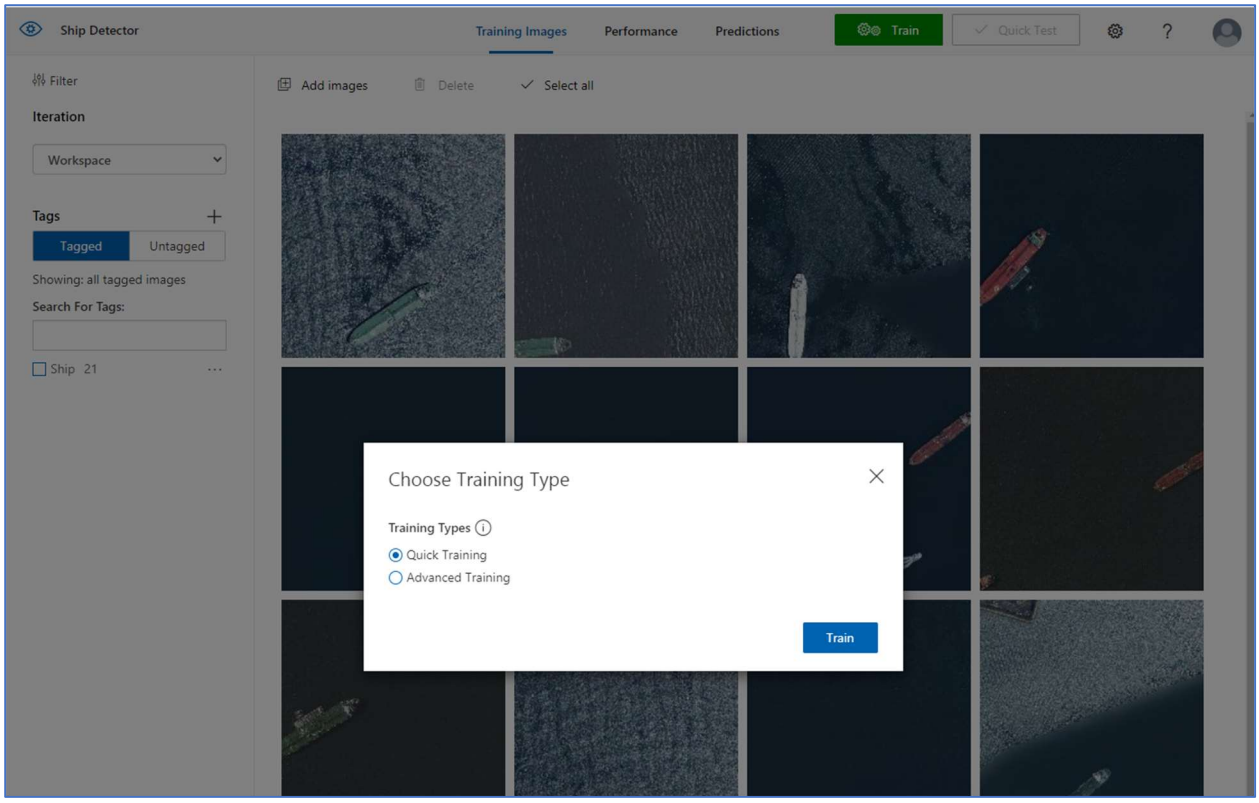
Use the Training API to upload images

You can use the graphical tool in the Custom Vision portal to tag your images, but many AI development teams use other tools that generate files containing information about tags and object regions in images. In scenarios like this, you can use the Custom Vision training API to upload tagged images to the project.

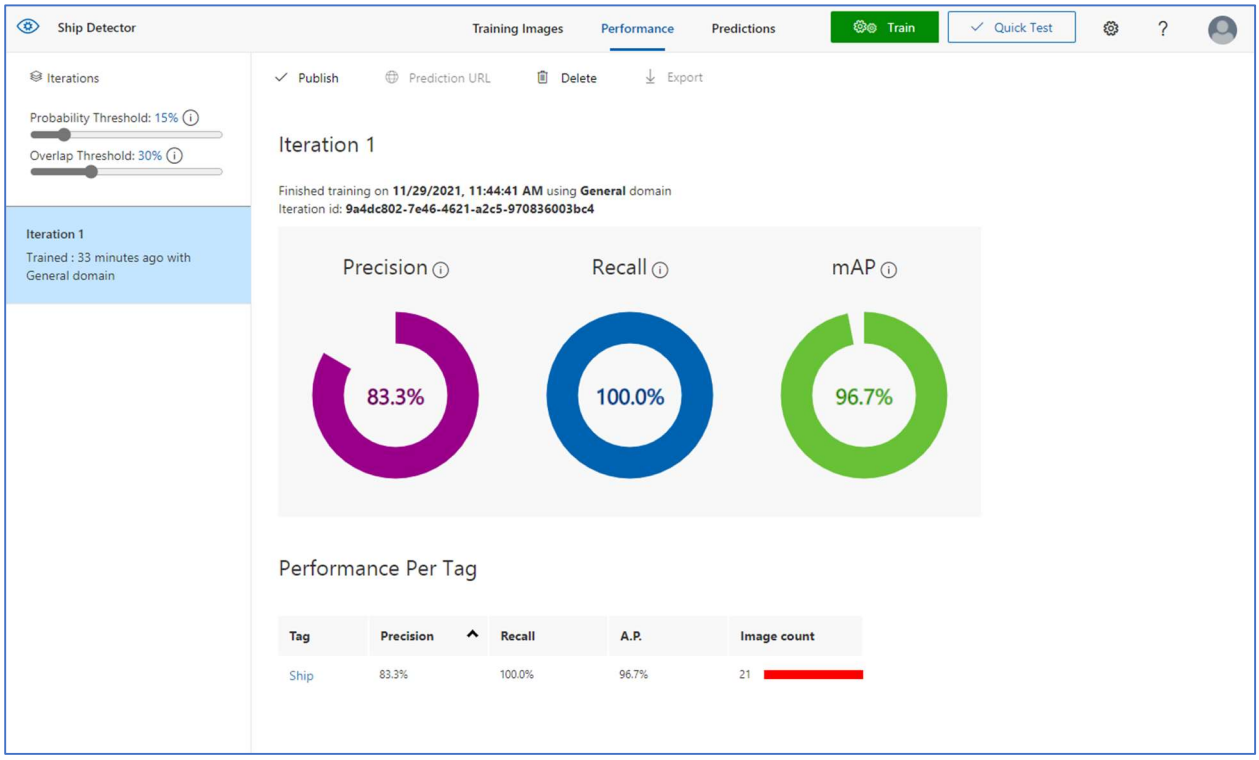
Train and test a model

Now that you've tagged the images in your project, you're ready to train a model.

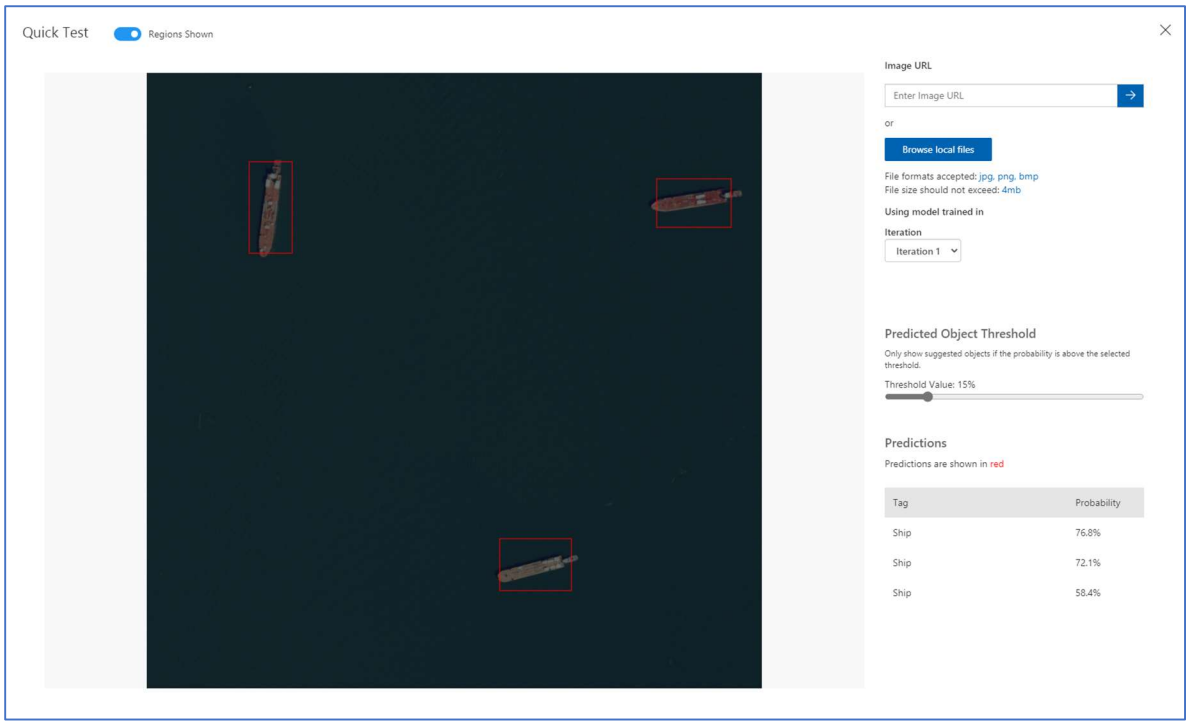
1. In the Custom Vision project, click **Train** to train an object detection model using the tagged images. Select the **Quick Training** option.



2. Wait for training to complete (it might take ten minutes or so), and then review the *Precision*, *Recall*, and *mAP* performance metrics - these measure the prediction accuracy of the classification model, and should all be high.



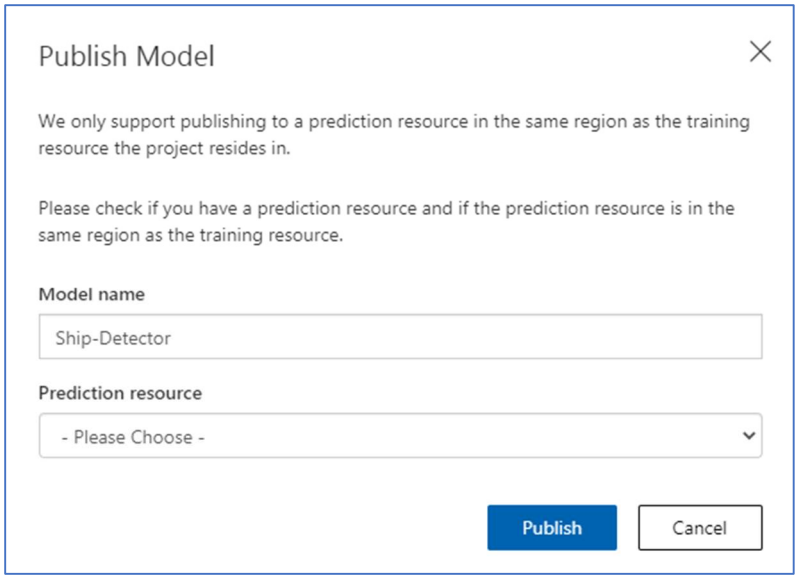
3. At the top right of the page, click **Quick Test**, and then in the **Browse local files** box, then select the Ship_Test.png file (or any other file in the folder) and view the prediction that is generated. Then close the **Quick Test** window.



Publish the object detection model

Now you're ready to publish your trained model so that it can be used from a client application.

1. In the Custom Vision portal, on the **Performance** page, click ✓ **Publish** to publish the trained model with the following settings:
    - **Model name**: Ship-Detector
    - **Prediction Resource**: *The **prediction** resource you created previously (not the training resource)*.

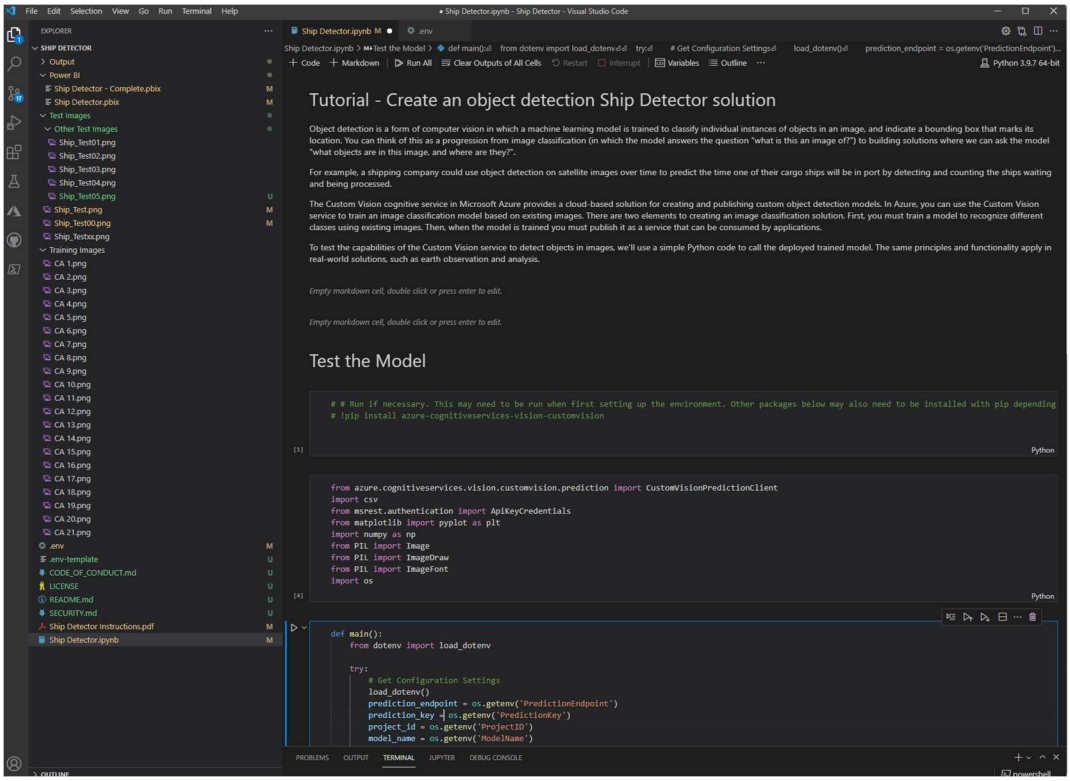


2. At the top left of the **Project Settings** page, click the *Projects Gallery* (👁) icon to return to the Custom Vision portal home page, where your project is now listed.
3. On the Custom Vision portal home page, at the top right, click the *settings* (⚙) icon to view the settings for your Custom Vision service. Then, under **Resources**, find your *prediction* resource (not the training resource) to determine its **Key** and **Endpoint** values (you can also obtain this information by viewing the resource in the Azure portal). Also note the *Published as*: **Ship-Detector** name on the Performance page as you will need this to test the model.

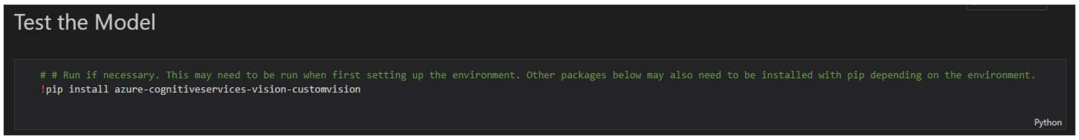
Use the image object detection model from a client application

Now that you've published the image classification model, you can use it from a client application. The example is developed in **Python**.

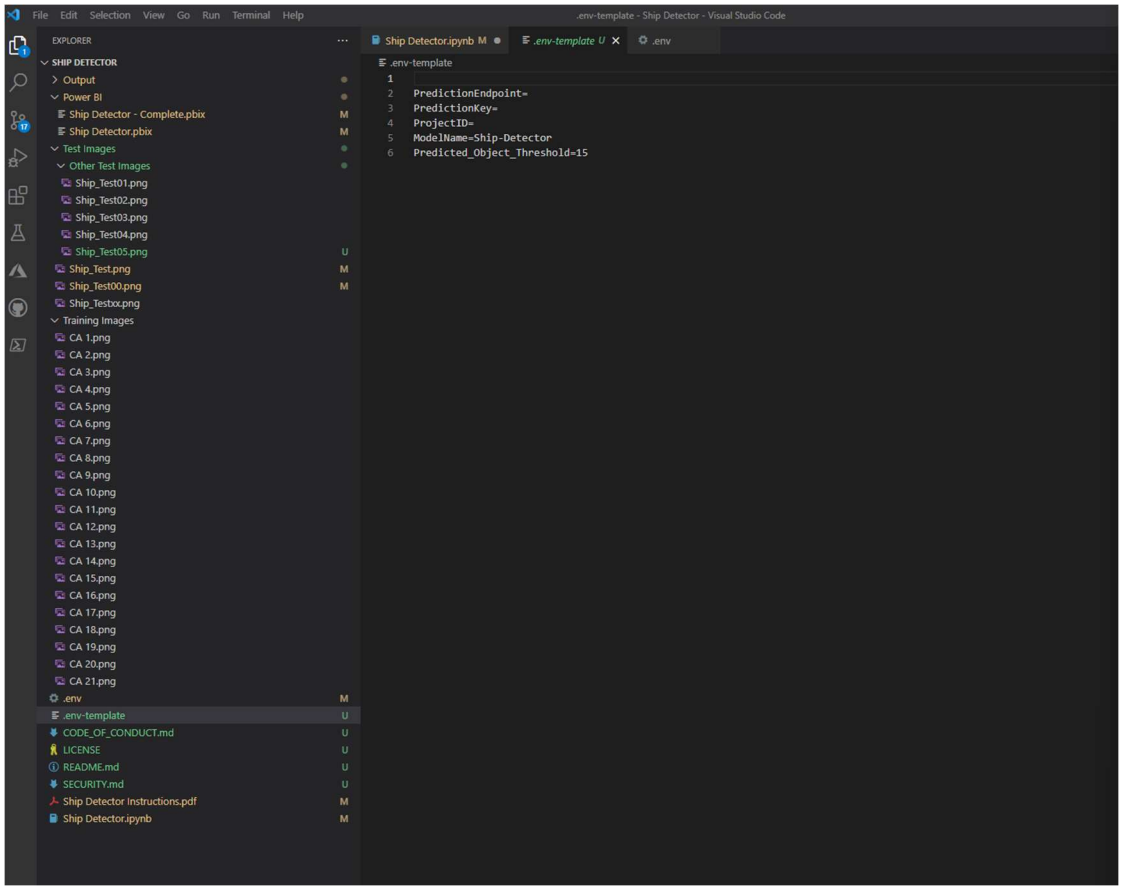
1. In Visual Studio Code, browse to the **Ship Detector** folder and find the **Ship Detector.ipynb** notebook file and open it.



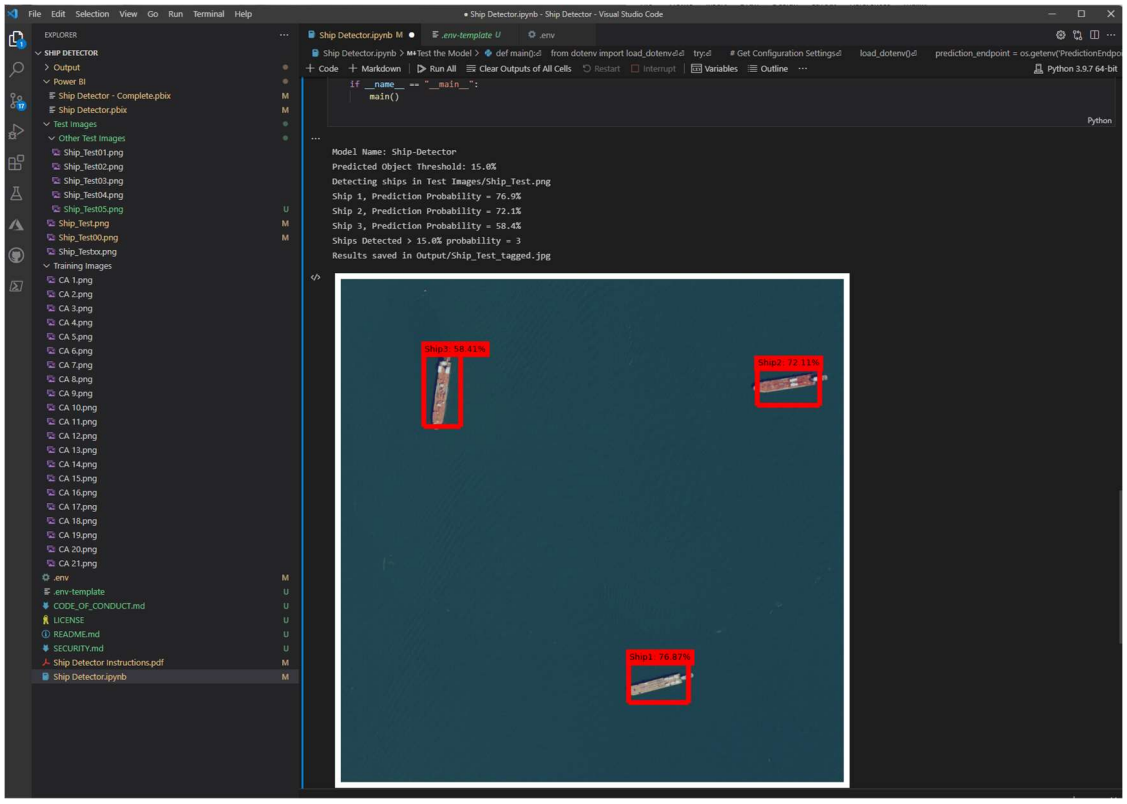
2. You may need to install required Python packages if you are setting up the environment for the first time. If you have not run Azure Cognitive Services in your environment, you will need to uncomment the "# !pip install azure-cognitiveservices-vision-customvision" line and run the cell. Then comment out the line as you will not need to run it again. You may also need to select a python version prior to running this cell.

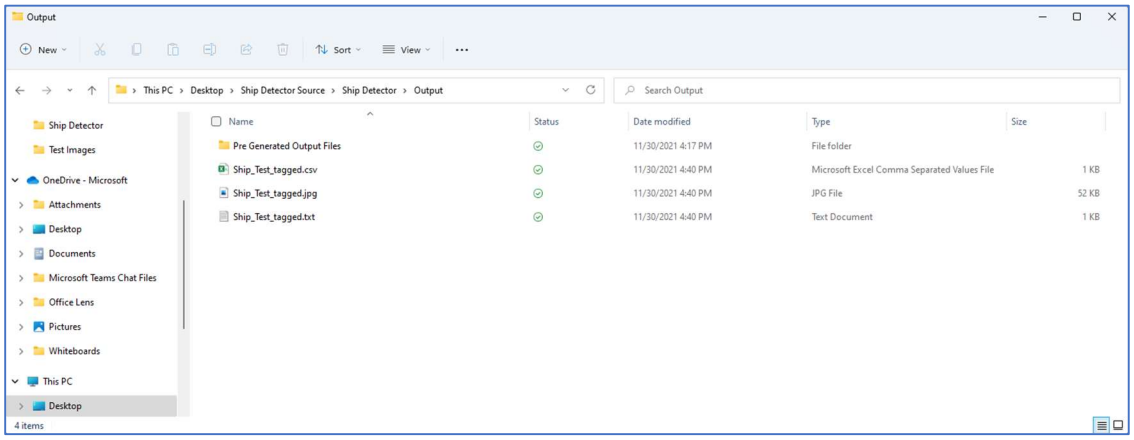


3. Open the configuration file for your client application (*template.env* for Python) and update the configuration values it contains to reflect the endpoint and key for your Custom Vision *prediction* resource, the project ID for the object detection project, and the name of your published model (which should be *Ship-Detector*). Save your changes and rename the *template.env* file to *.env* file.



4. Go back to view your notebook file (*Ship Detector.ipynb*) and review the code it contains, noting the following details:
    - The variable image_file = '**Ship_Test.png'** loads the desired image file to process. You can select one of the other images to test the code such as **Ship_Test00.png** or **Ship_Testxx.png** (no ships). Note that you will want to use the output of Ship_Test.png for the second part (Power BI) of this document.
    - The section labeled "Ship Location Information" shows the ship location data that is defined for this example. Code could be inserted to extract imagery ship latitude and longitudinal information from the ships detected after the detection loop. *The example code has the locations for the test image **Ship_Test.png**.*
    - The output files are saved to the Output subfolder after the code is executed.
5. Execute the remaining cells or Run All.
6. After the program has completed, view the resulting **Ship_Test_tagged.jpg** file to see the detected objects in the image. View the **Ship_Test_tagged.txt** to view the details of the objects detect. A third file **Ship_Test_tagged.csv** is created that holds the information for geospatial visualization in the next section.





More information on Custom Vision

For more information about object detection with the Custom Vision service:
https://docs.microsoft.com/en-us/azure/cognitive-services/custom-vision-service/

For more information about Azure Space:.
https://azure.microsoft.com/en-us/solutions/space/#overview

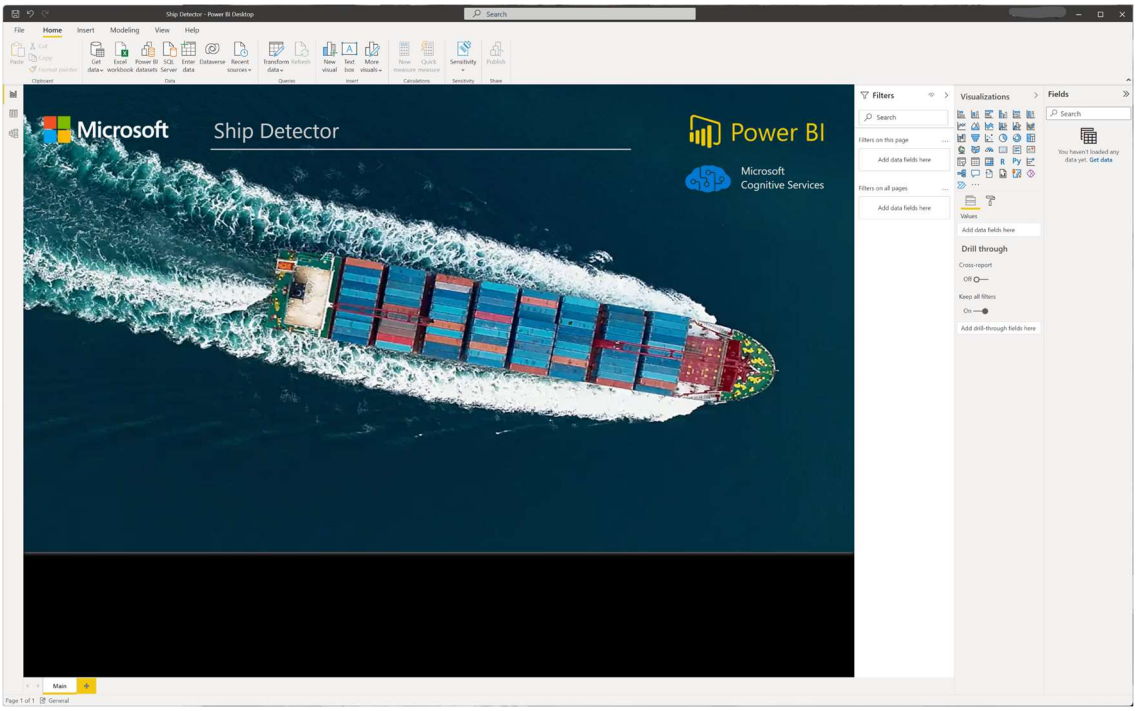
Part 2: Visualize Detected Ships with Power BI

In this exercise, you will use the Power BI to visualize the location of the detected ships. Note, this exercise is for demonstration purposes only with the latitude and longitudinal data defined in the code. This code can be modified to extract this information from the image and dynamically display on the map.

Open the Power BI Ship Detector Template

If you have already cloned **ShipDetector** code repository to the environment where you're working on this lab, open it in Visual Studio Code; otherwise, follow these steps to clone it now.

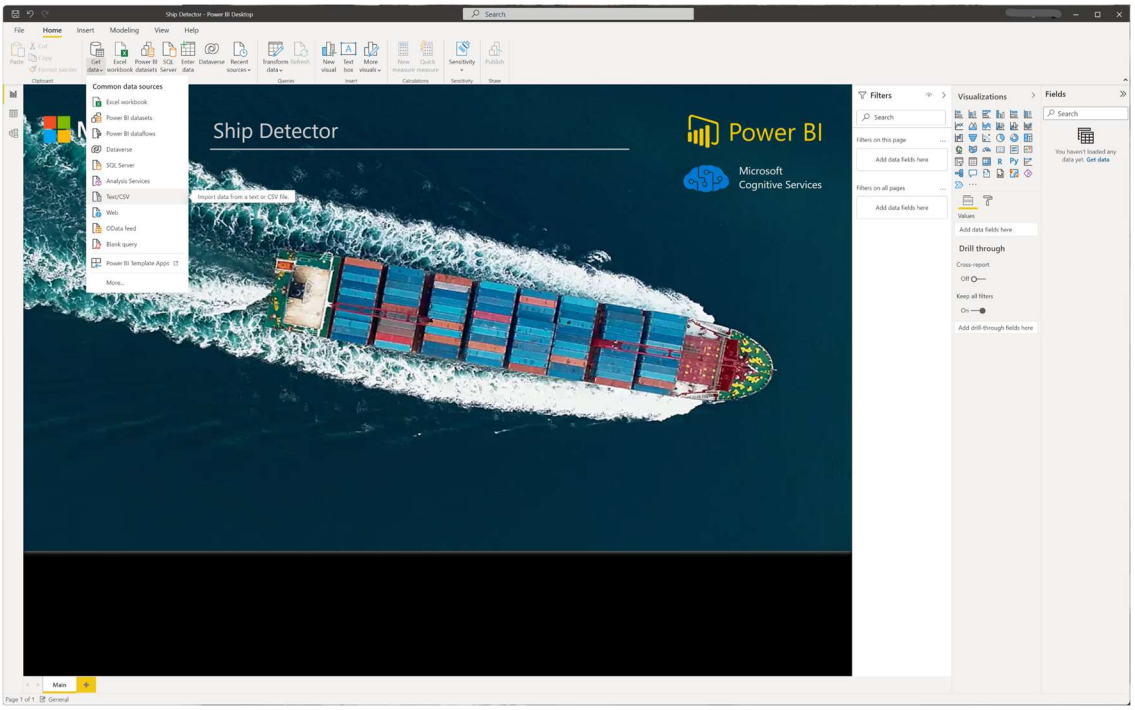
1. Login to GitHub with your account or
2. Open the palette (SHIFT+CTRL+P) and run a **Git: Clone** command to clone the `https://github.com/Microsoft/ShipDetector` repository to a local folder (it doesn't matter which folder).
3. When the repository has been cloned, open the Power BI folder under Ship Detector main folder.
4. If you do not have a Power BI account, you will need to create a free account first.
5. Download Power BI Desktop Power BI Desktop—Interactive Reports | Microsoft Power BI.
6. Find and open the file **Ship Detector.pbix** in the Power BI sub folder.

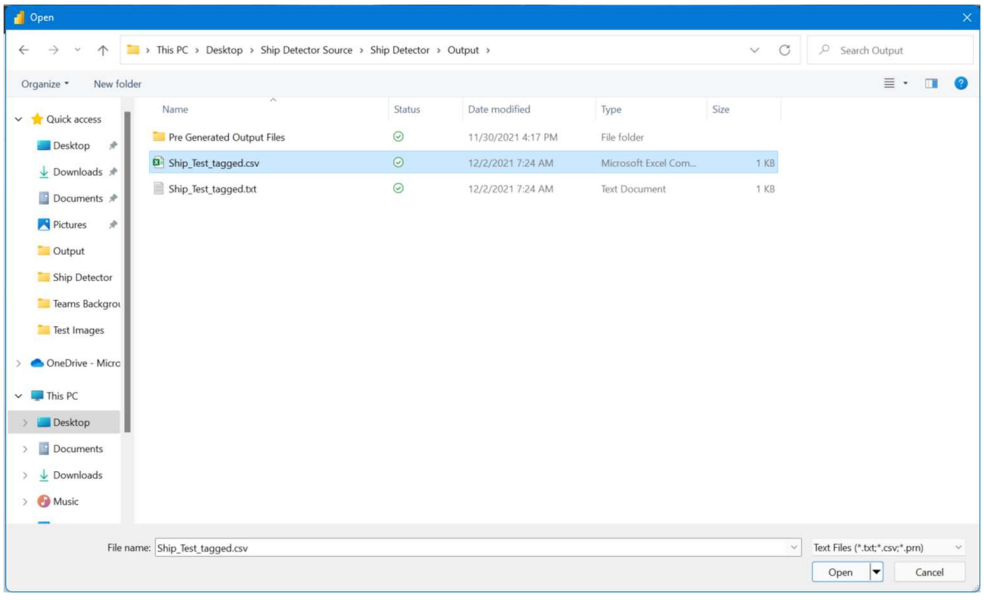


Build the Ship Detector Report

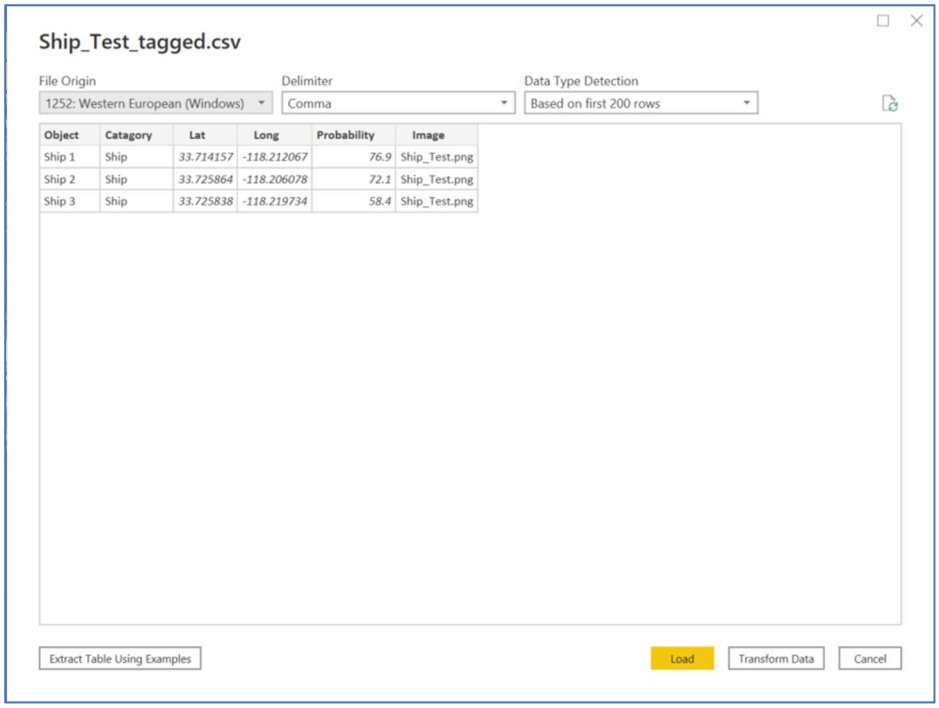
The first step in the process is to build a simple single page report with visualizations of the ship data.

1. Before we can add visualization, a connection to the desired data file must be made. On the menu bar, click on the **Get data** option and then select **Text/CSV**. Find the file **Ship_Test_tagged.csv** in you Output folder and select it. Click on Open. Note the Fields blade on the far right shows that you have not loaded any data yet.

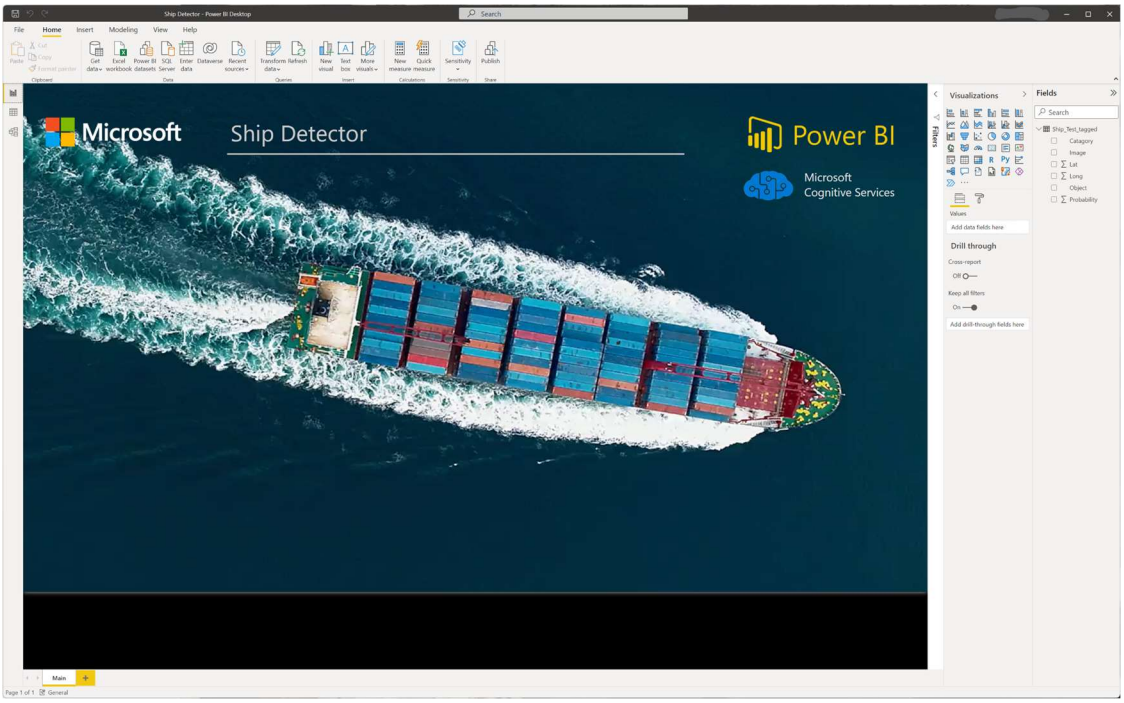


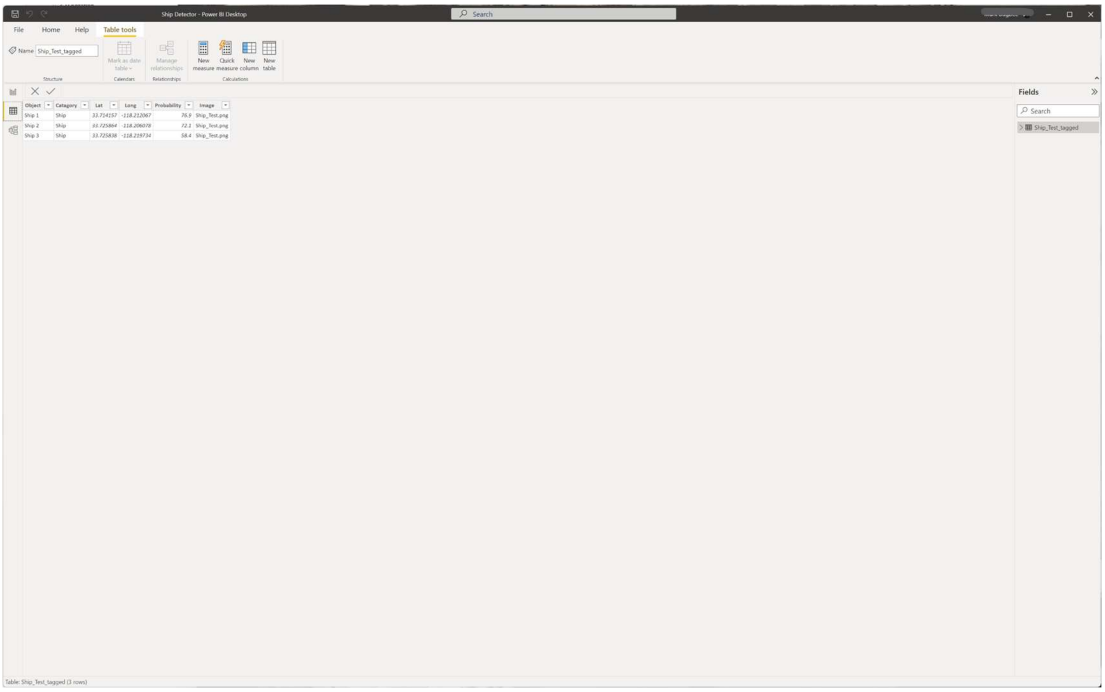


You will see the Load window and it will look similar to the following:



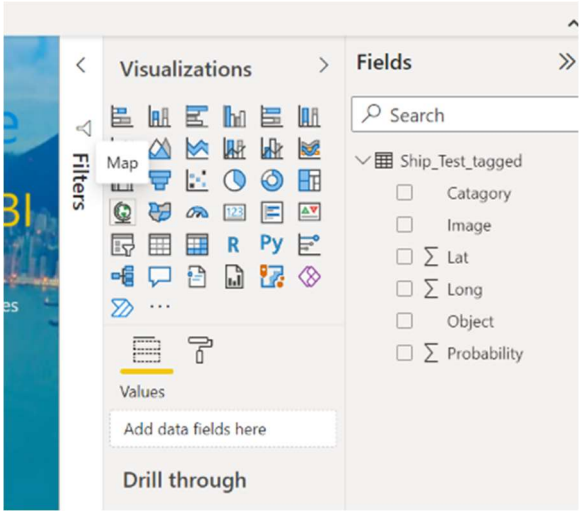
Select Load. You will now see the data fields on the Fields blade. You can also check your data by clicking on the Data tab on the far left side.

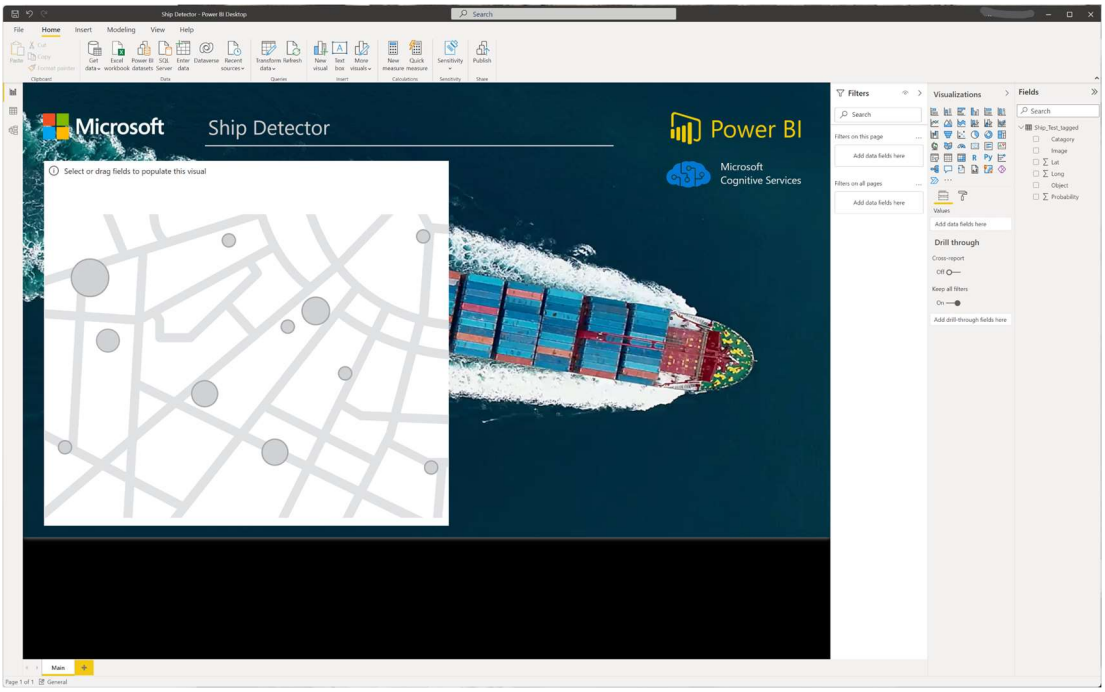




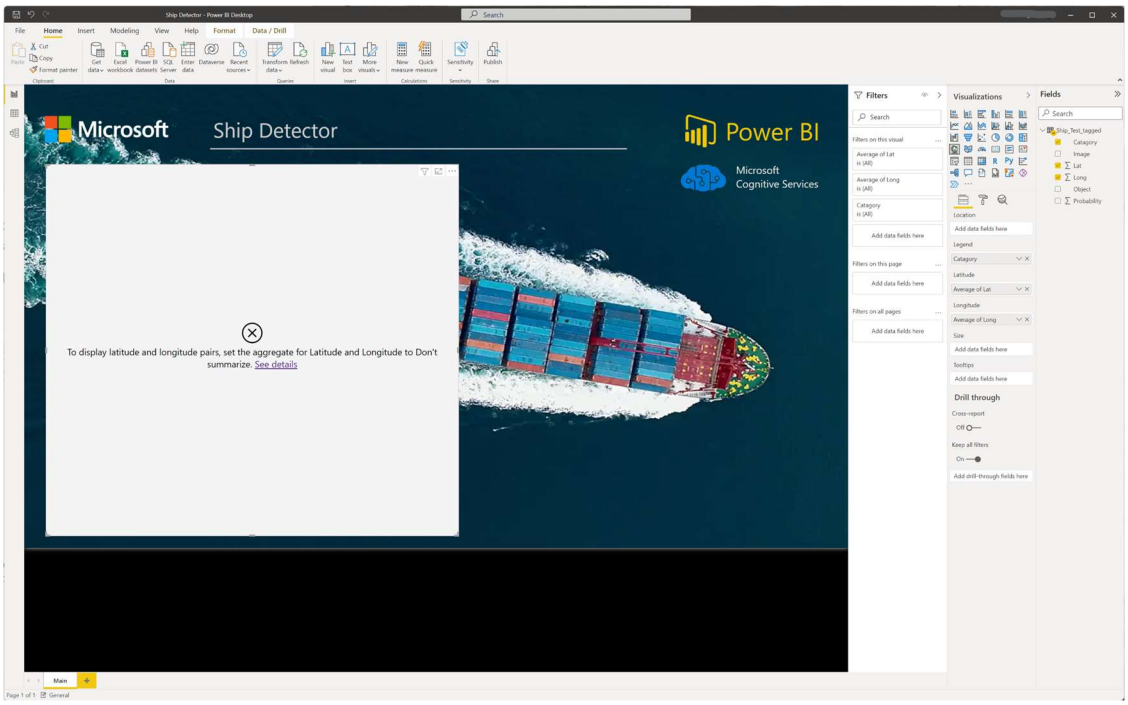
The data has successfully been added.

2. We are now ready to add our first visualization. Find the Map visualization and select it. Be careful not to choose the ArcGIS Maps for Power BI visualization. A blank map object will be added to your report page. Size the object as shown below.

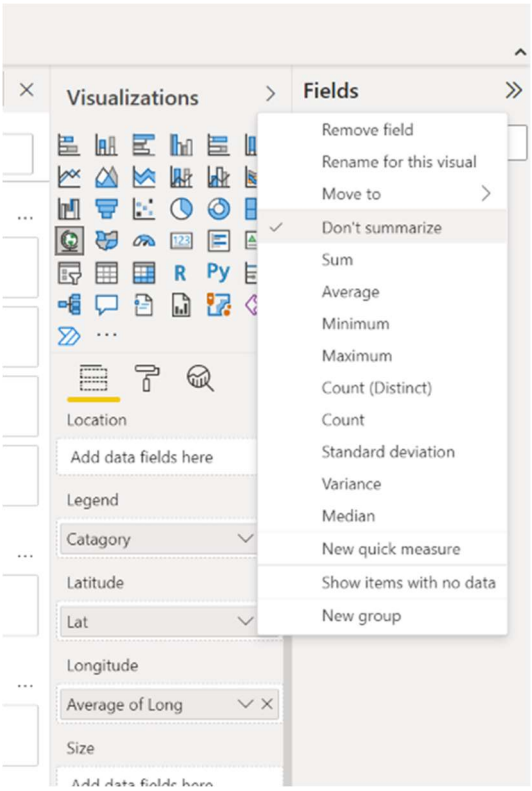


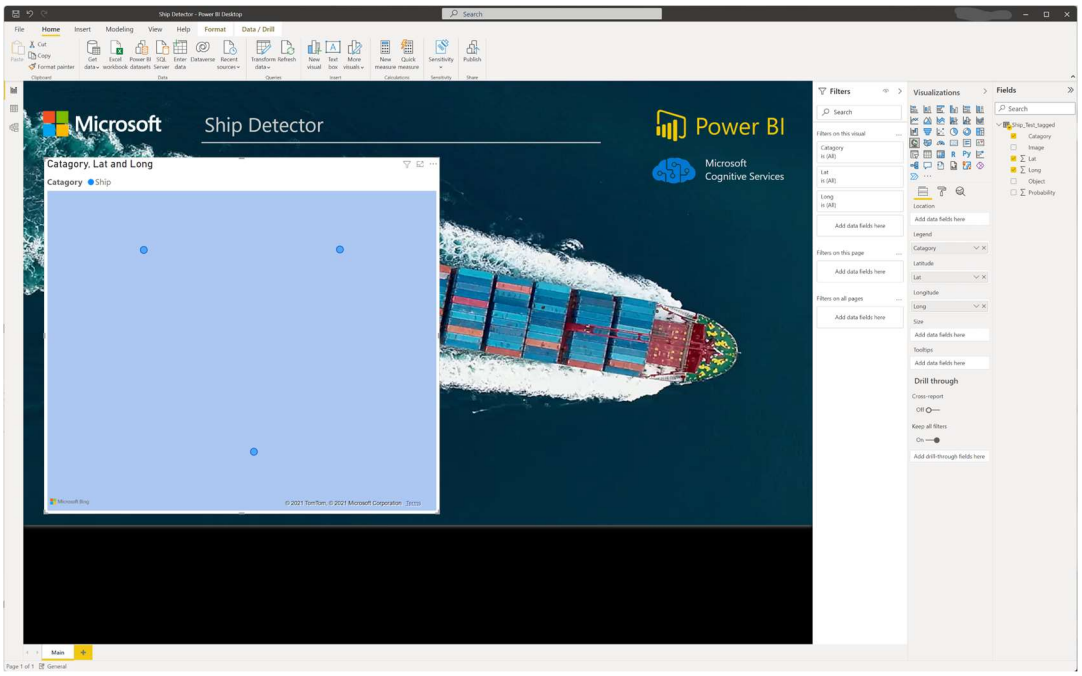


3. Select the map object by clicking anywhere on the object. Expand the Ship_Test_tagged fields in the Fields blade on the right. Drag the Category field to the Legend box on the Visualization. Drag the Lat to the Latitude box and the Long to the Longitude box. I will look as follows:

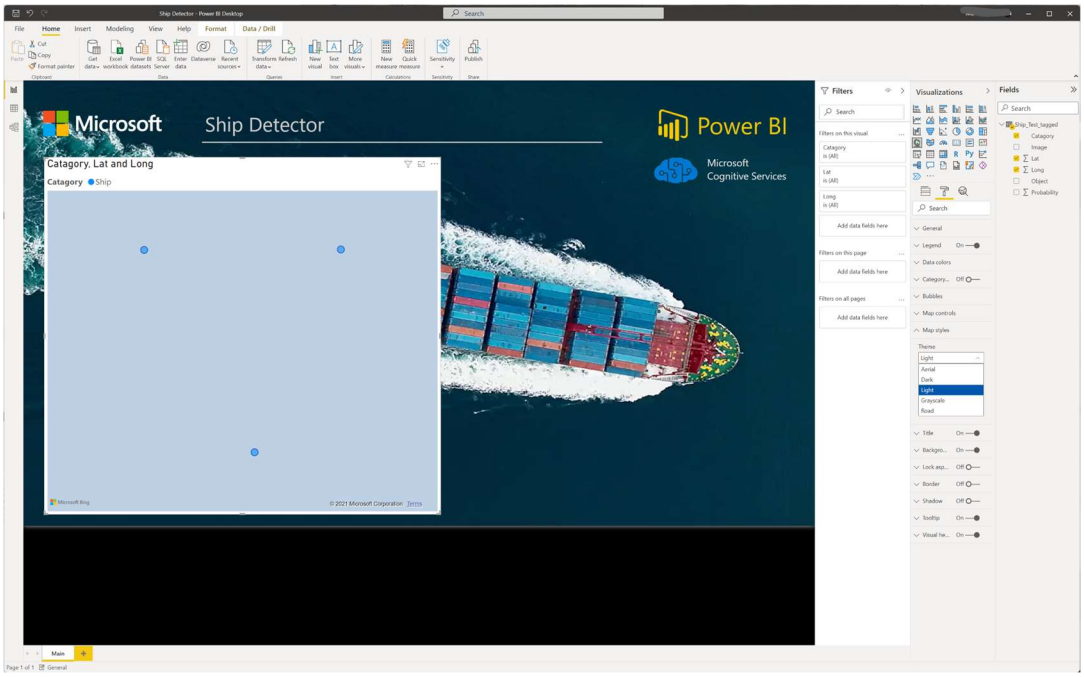


Note that the Latitude and Longitude boxes default to **Average of Lat** and **Average of Long**. Click on the fields and set them to **Don't summarize** as show. Your map should look like the one below.

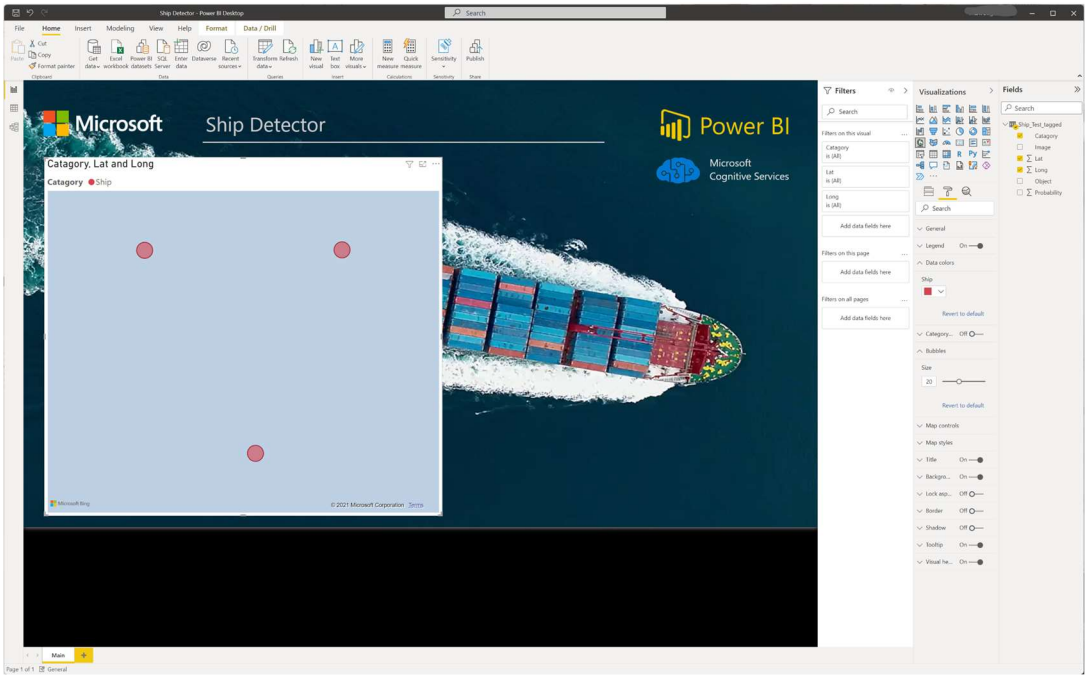




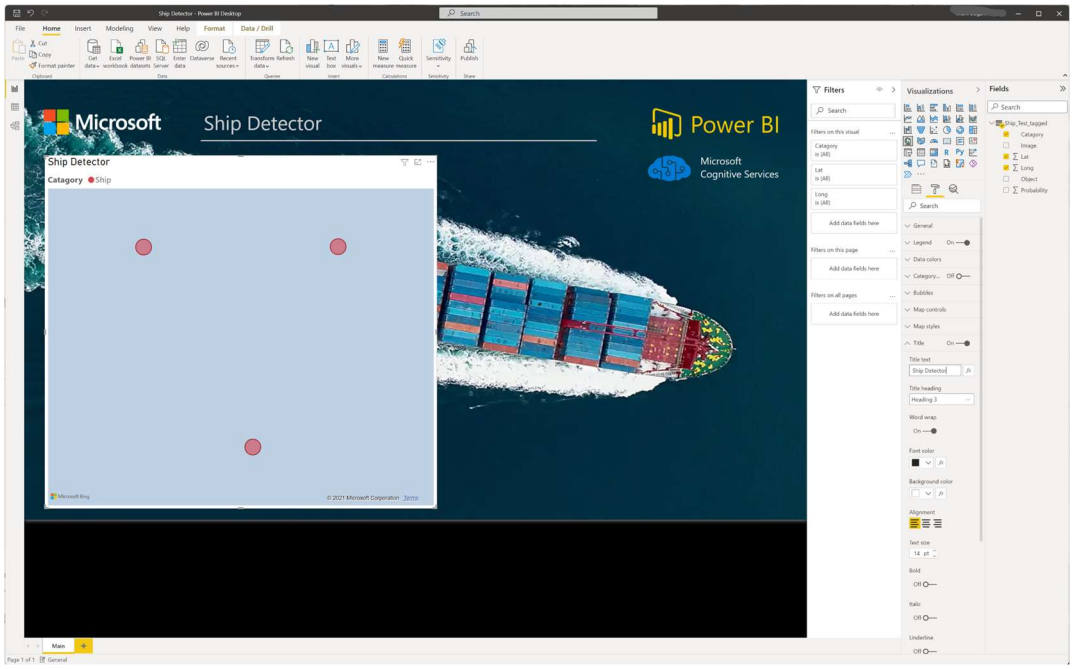
4. Now format the map by clicking on the Format tab (paint roller icon). Select Map styles and set the Theme to Aerial.



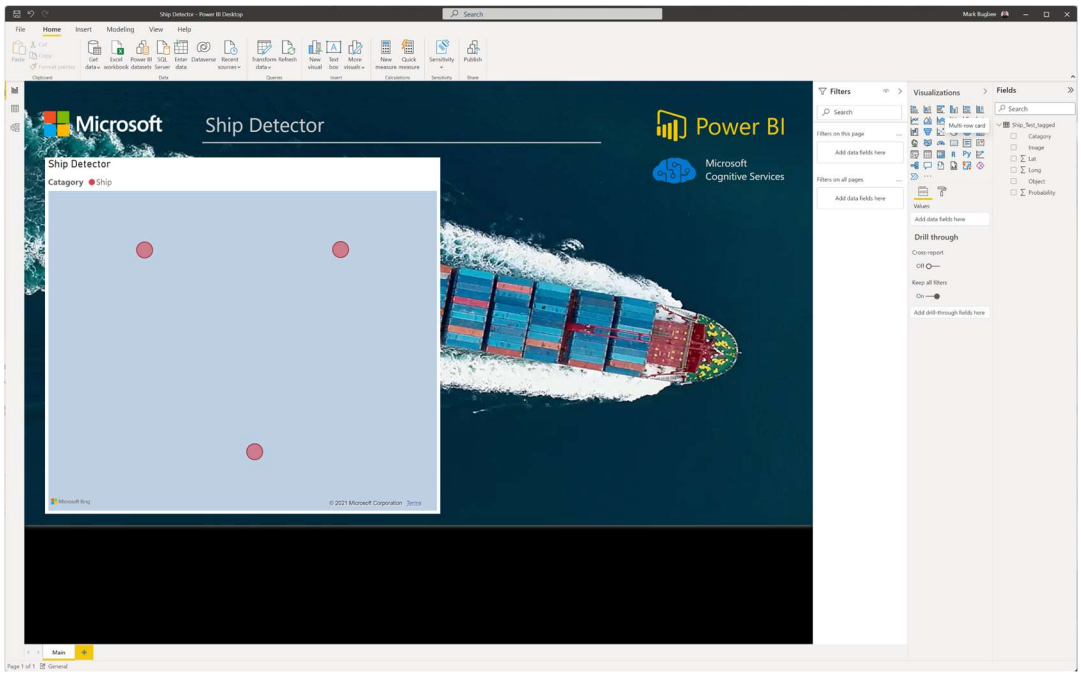
Set the Data color to red and adjust the Bubble Size as desired.

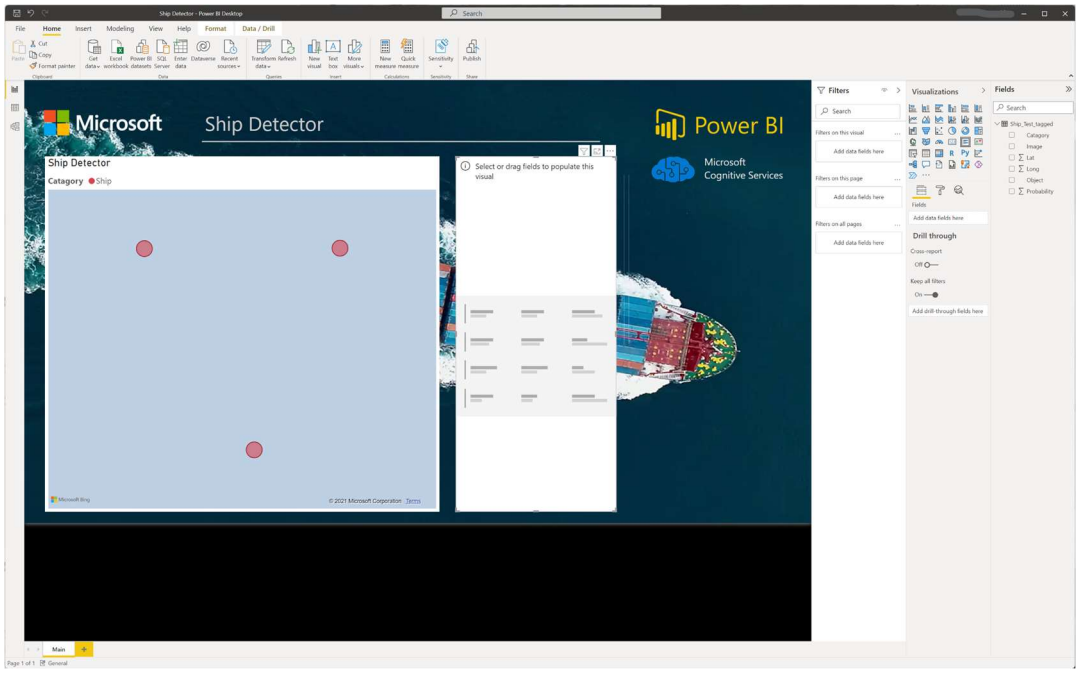


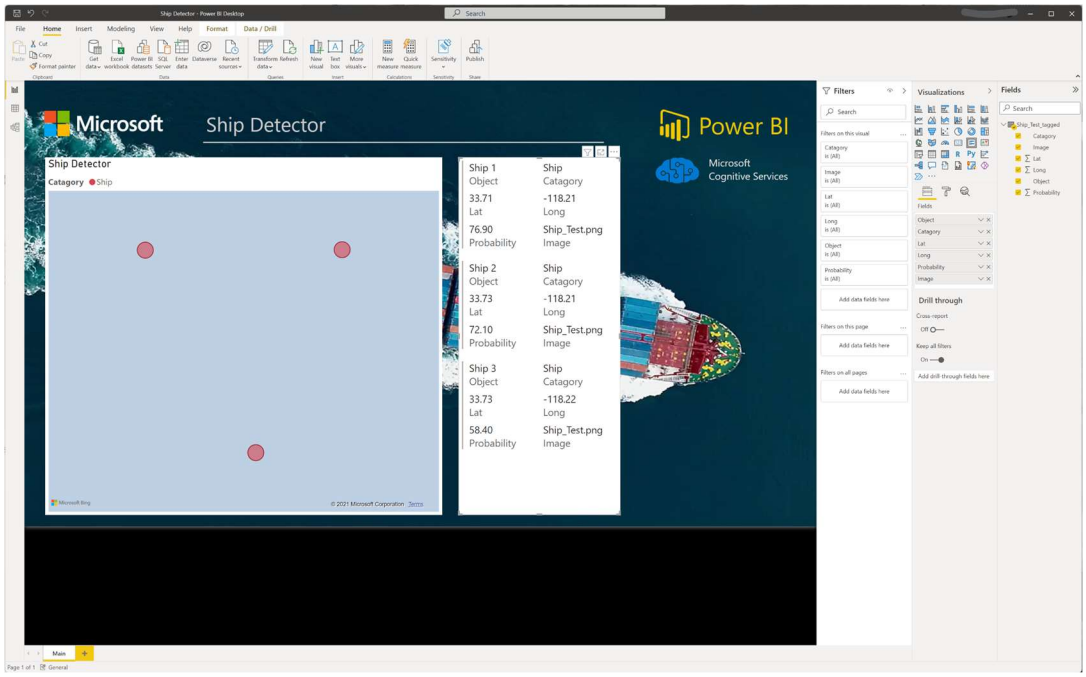
Set the Title to **Ship Detector**.



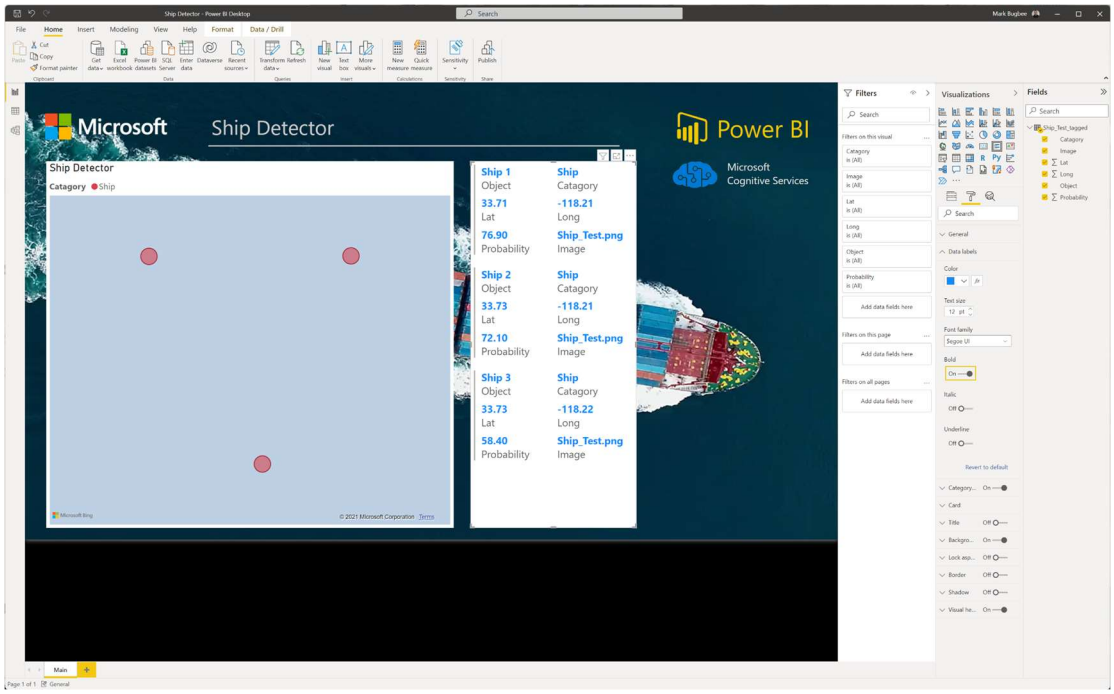
5. Now click anywhere outside the map control on the page and add a Multi-row Card. Align and size the card on the page. Then click on the object and select each of the fields on the Fields blade as shown below.

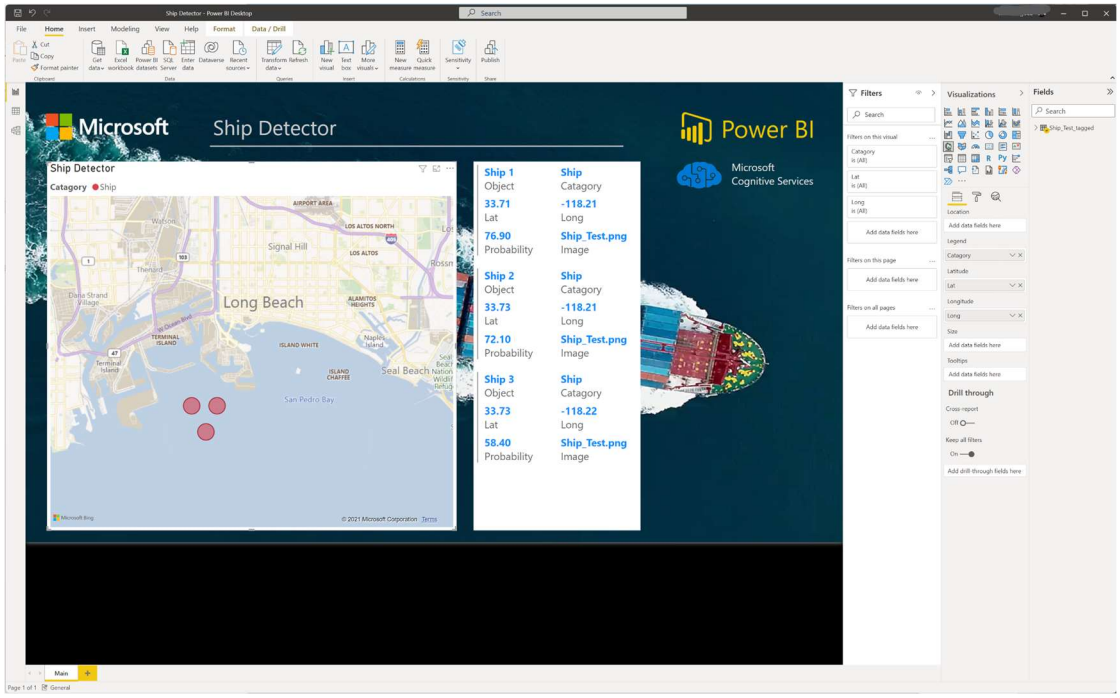






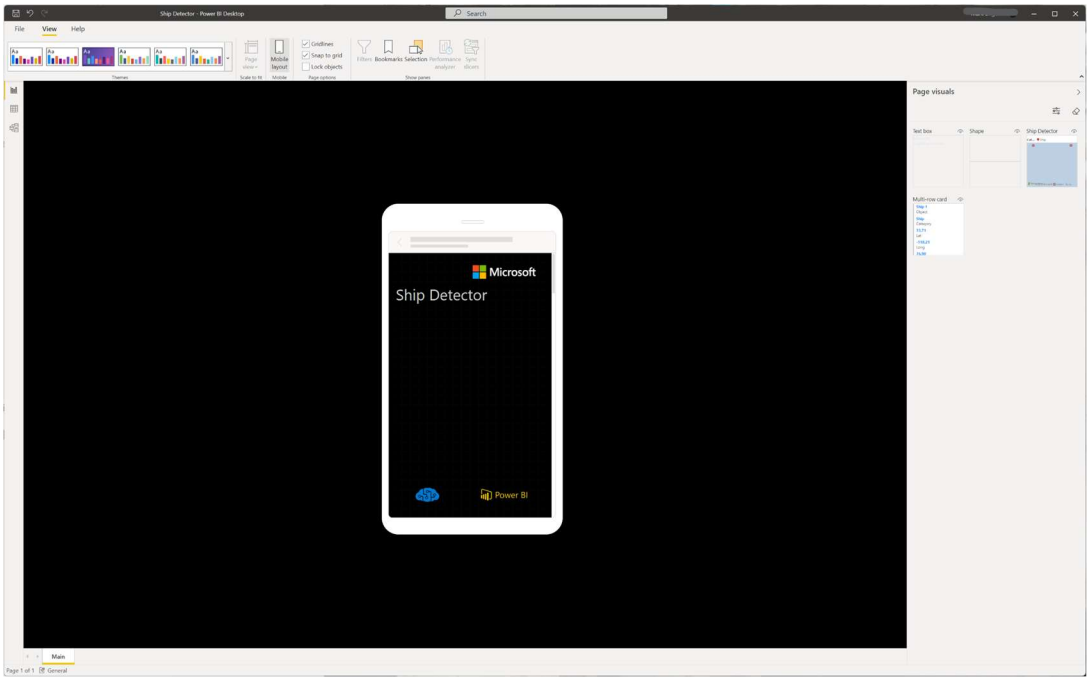
6. Format the card by selecting the Data labels color and setting it to bold. You are now finished with the page. Note you can zoom in or out on the map as it is interactive.



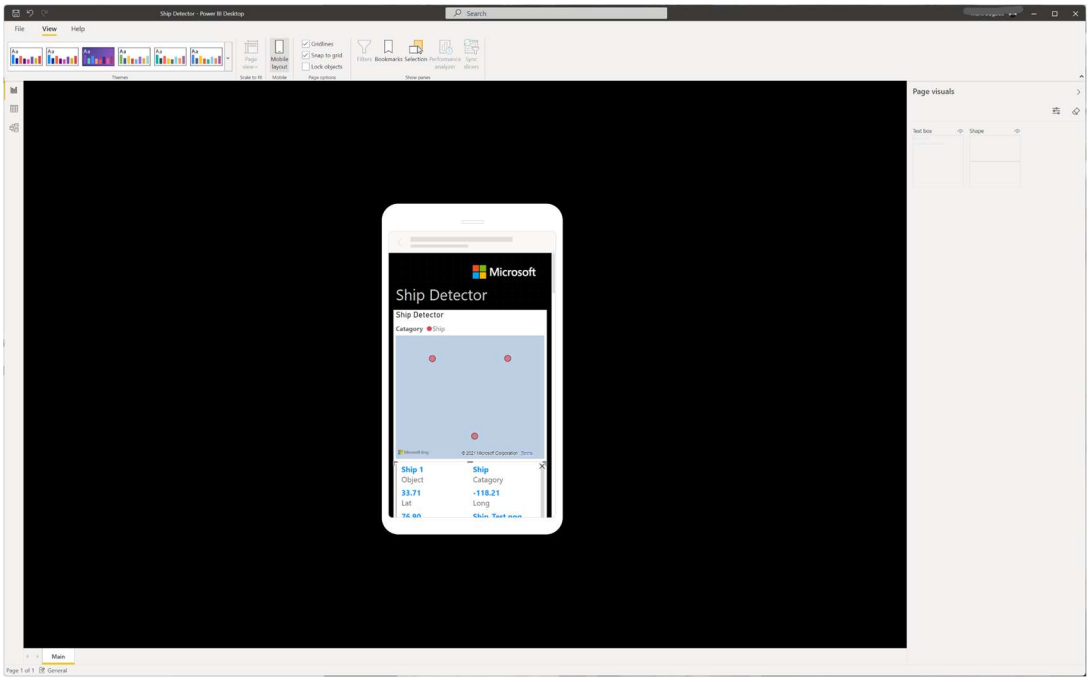


Build the Ship Detector Report Mobile Version

1. Open mobile view by selecting the View tab on the top menu bar. Then click on the Mobile layout. You will see the following.



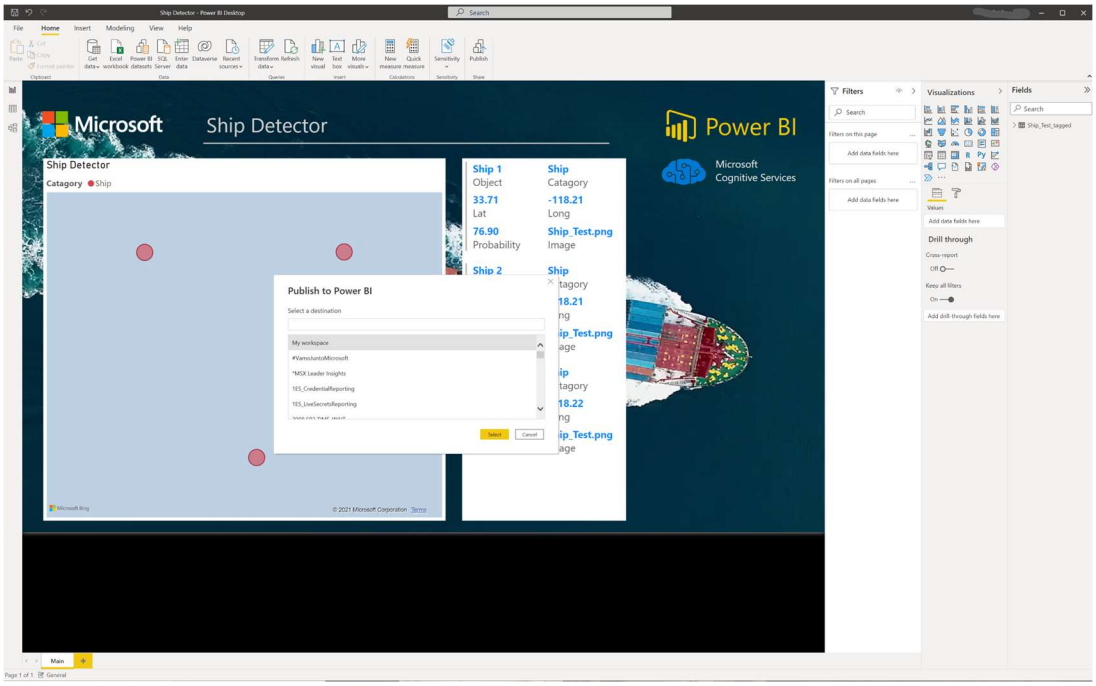
2. Now drag the items on the right in the Shape blade onto the phone page. Size them as show. That's it! Your mobile view is now complete. Save your file.

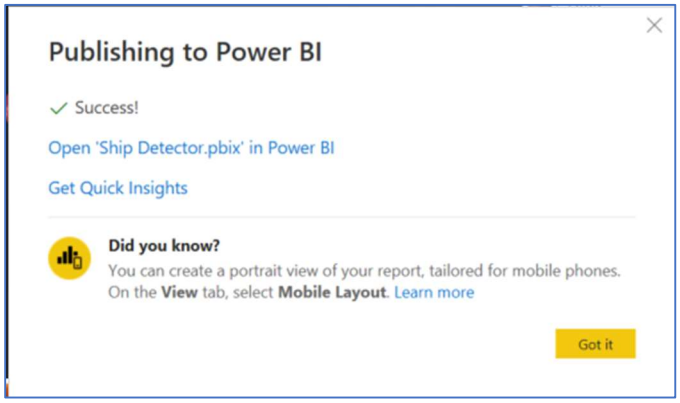


Publish the Ship Detector Report

Now you are ready to publish your report to the Power BI service for distribution.

1. Go back to the Home view by clicking Home on the top menu. Now click on the Publish button on the right of the top menu.



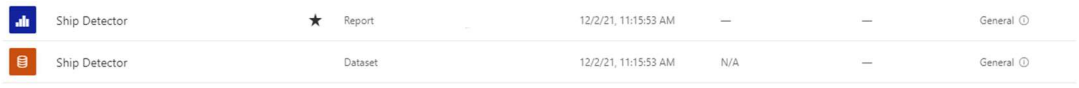


2. You are now ready to open the report in the Power BI service and on your mobile device.

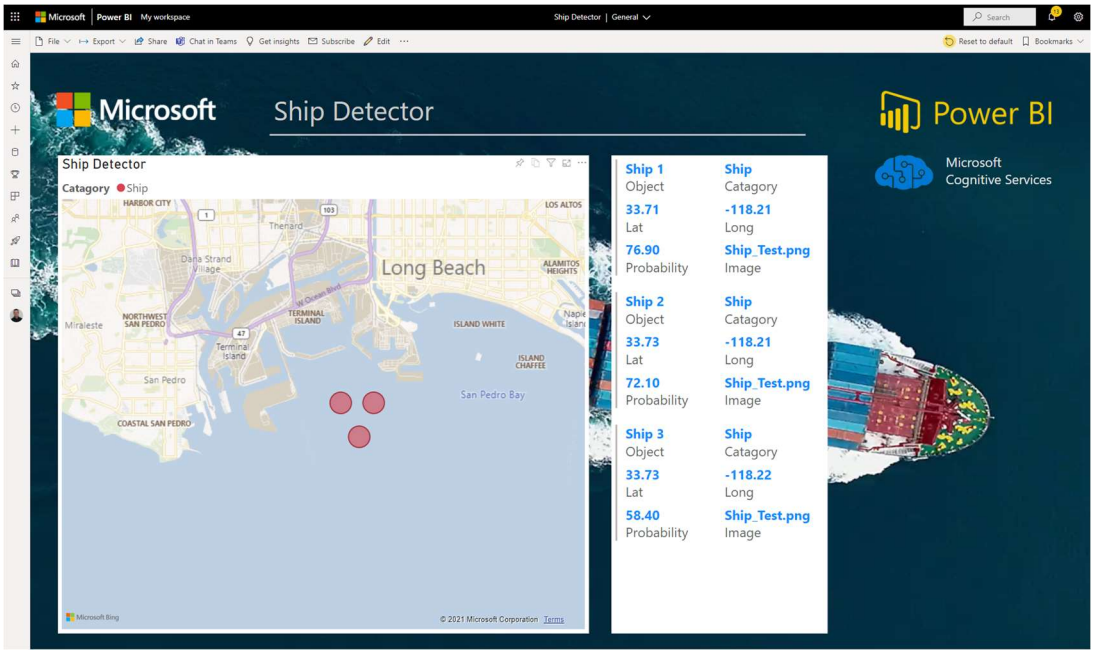
View the Ship Detector Report

To view your report in the Power BI browser, log in to the service and go to My workspace.

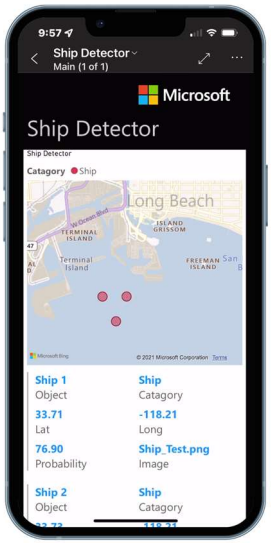
1. Find the Ship Detector Report and verify the Refreshed date. You should see a date and time that reflects when you last refreshed the data and published the report.



2. Click on the Ship Detector report. You now should see your report.



3. Now go to your mobile device. Download the Power BI application for your mobile phone and login using the same account as the service. Find the report in the mobile menu and open it. You should see your report. Congratulations on building your Ship Detector!



More information on Power BI

For more information about Power BI:
https://docs.microsoft.com/en-us/power-bi/

For more information about Azure Space:.
https://azure.microsoft.com/en-us/solutions/space/#overview