Lab2: Amazon SageMaker Object Detection Algorithm

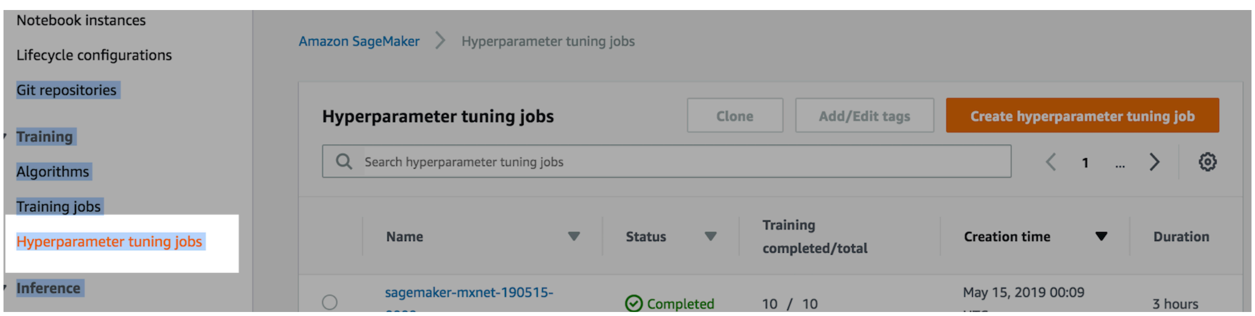
In this next lab we're going to train and deploy an object detection model using Amazon SageMaker's Object Detection algorithm.

This option has the benefit of allowing a data scientist to configure and tune an object detection model without having to write code. Amazon SageMaker provides a number of [built-in Algorithms](). In some cases, they provide a performance and scalability benefit over other alternatives through GPU and distributed training support.

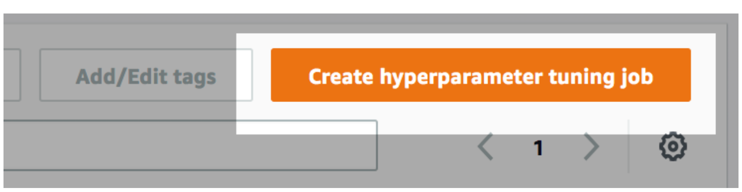
I. Launch an Automatic Model Tuning Job

Amazon SageMaker can automate much of the hyperparameter search process through its Automatic Model Tuning capabilities. In the following steps, we will provide Amazon SageMaker with a range of hyperparameters to search over. Amazon SageMaker will train multiple jobs on our behalf and in the process converge towards optimal hyperparameters within those ranges using [Bayesian Optimization](). Bayesian Optimization has been demonstrated to be more effective than naïve approaches like grid and random search. Thus, the time and cost of training can be reduced through this capability.

1. From the Amazon SageMaker console, select **Hyperparameter tuning jobs** from the navigational panel on the left hand side.

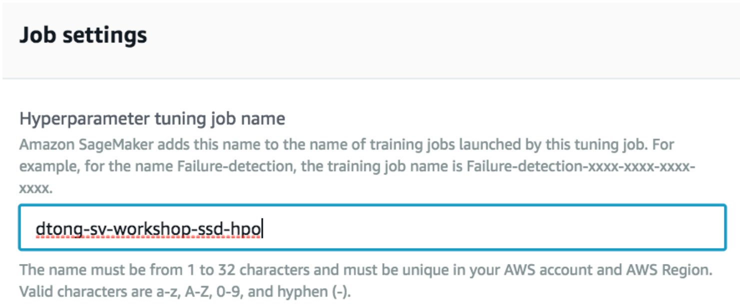


2. Click on the **"Create hyperparameter tuning job."**

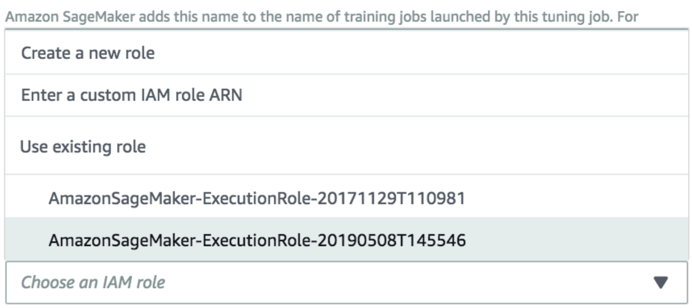


3. Work through the job configuration wizard.

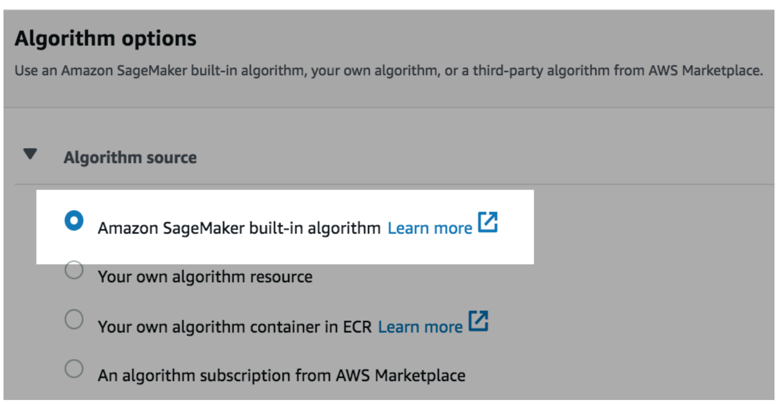
   Enter a **unique name** for the job.



If you completed Lab 1, you can reuse the IAM role that you created previously under "**Use existing role.**" Otherwise, choose **"Create a new role"** and give the role access to all S3 buckets (for the sake of keep this lab exercise simple).

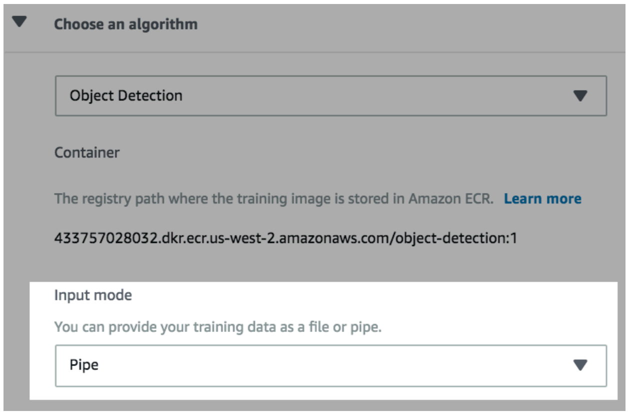


4. Under Algorithm options, select **"Amazon SageMaker built-in algorithm."**



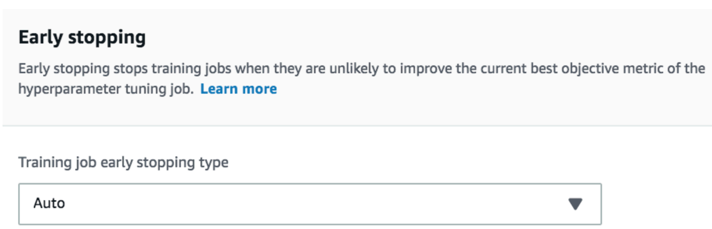
Search and select **"Object Detection"** from the list of algorithms.

5. Select **Pipe** under Input mode. This option allows the data to be streamed from S3 instead of staging the data on storage attached to the training machines. This mode reduces training start-up time, and is provides scalability necessary for large training sets.

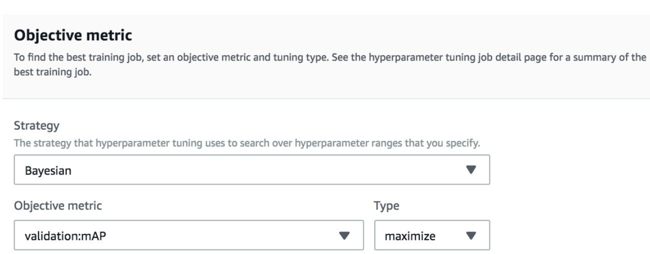


Click on **Next.**

6. You can optionally enable **early stopping**.



7. We need to specify an objective metric for Bayesian Optimization to optimize on. **Select validation:mAP and maximize**, so that the job seeks to maximizes mAP on our validation set.

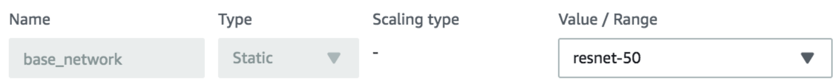


8. Next, configure the **parameters** for our Object Detection algorithm. Leave the parameters with their default values if they aren't explicitly mentioned below.

The parameters selected are practical for the purpose of demonstration. We're going to train the algorithm with only 10 images that were annotated in Lab 1.

In practice, we would use much more data, and require training time beyond the time available for a workshop.

Select resnet-50 as our base network.



Change **num_classes** to 1 ("bird" is are only class).

Set **epochs** to 60.

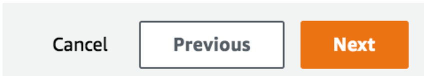
Set the **learning_rate** to the values below.

Reduce the number of optimizers down to **adam**.

Set the **mini_batch_size** to be between 1 and 2 (since we only have 10 images).

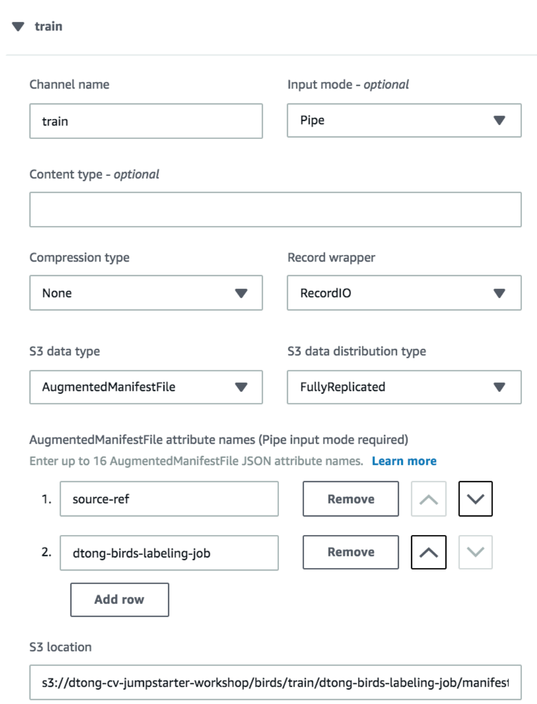
Set **num_training_samples** to 10.

Click on **Next** at the bottom of the page.



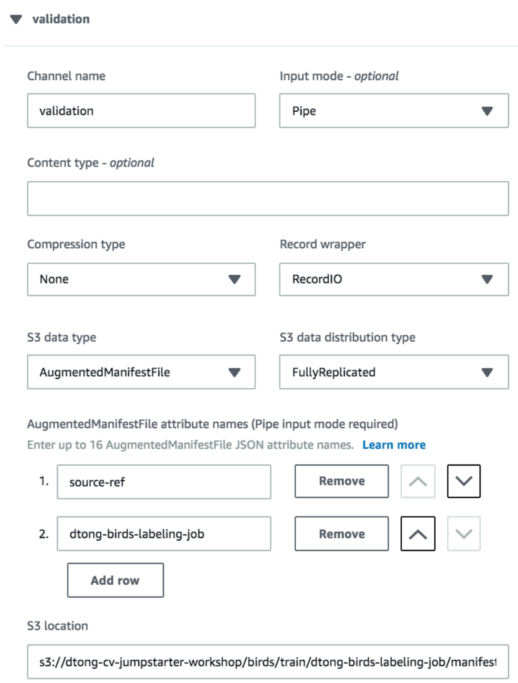
9. Configure the **train** channel, which specifies how the training data is fed into the training cluster.

    - Set the input mode to **Pipe**.
    - Set the **Record wrapper to RecordIO**.
    - Set the **S3 data type** to **Augmented ManifestFile**. With this setting we can use the output.manifest file created in Lab 1 to inform the training cluster about how to locate, parse and stream the images on S3.
    - Ensure you specify the attributes from the output.manifest file, so SageMaker knows how to parse the manifest:
        - The first attribute should be **source-ref** provides a mapping between image and annotations as well as the location in S3.
        - The second attribute should be the **name of your labeling job**. In the output.manifest file, the class labels and bounding box data are child elements. **This value will be different from the one in the screenshot provided below**.
    - Set the S3 location to the S3 URI of your output.manifest.

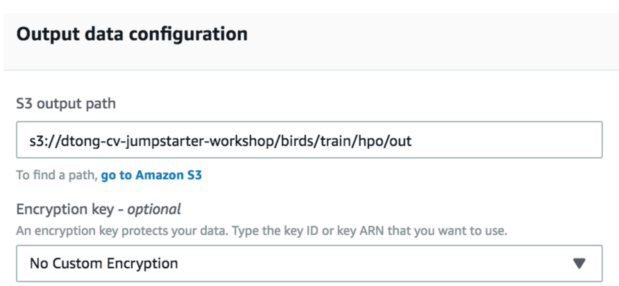


10. Create a validation channel, and replicate the configurations from the train channel.

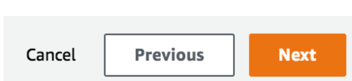
This will result in the training algorithm to use the training set for validation. In practice, we should have a separate data set for validation. For the sake of time and demonstration, we make do with the 10 images we annotated in Lab 1.



11. Under **Output data configuration**, specify a location in your S3 bucket where the tuning process will output the trained model artifacts.

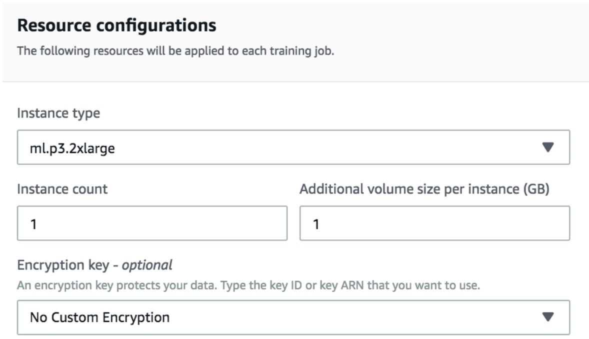


Click the **Next** button.



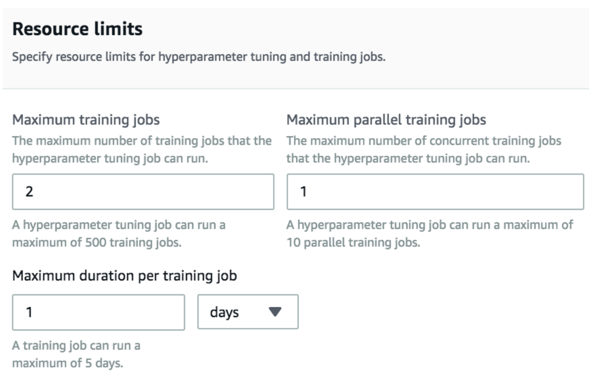
12. Under **Resource configurations**, select *ml.p3.2xlarge*. This provides us with 1 Nvidia V100 GPU. Leave the instance count to 1.

We utilize GPU for training because it will be magnitudes faster than CPU. We limit the instance count to and GPUs to 1 for the sake of managing the cost of this lab.

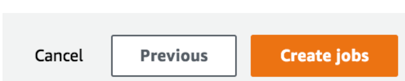


13. Set the **Maximum training jobs to 2**, and **parallel training jobs to 1**. In practice, you will run many more training jobs, so that Bayesian Optimization has an opportunity to converge towards optimal hyperparameters.

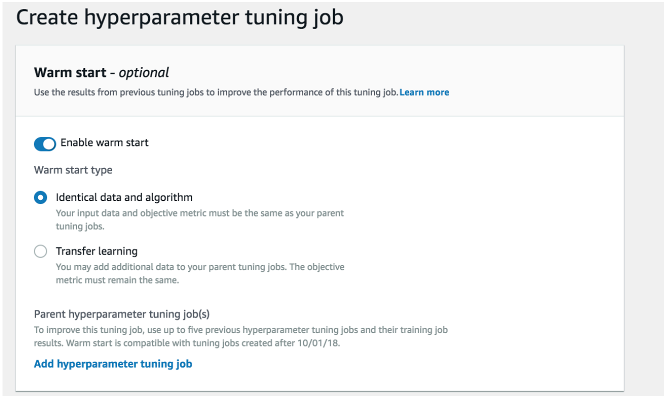
    In the lab we only run two training jobs for the purpose of demonstration.



Click on **Create jobs** to launch the tuning job.



In practice, you will run multiple automatic tuning jobs. With each job, narrowing down the range of hyperparameters will yield better results. Use the warm-start feature to leverage the results from previous automatic model tuning jobs.

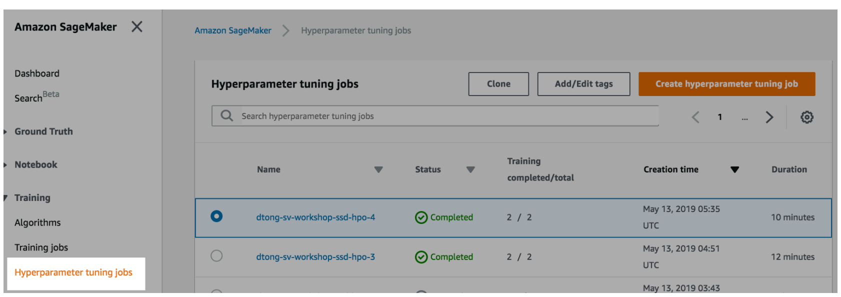


Once you hone in on some good parameters, you may switch to manual control and fine tune the model by running individual jobs with appropriate parameters.

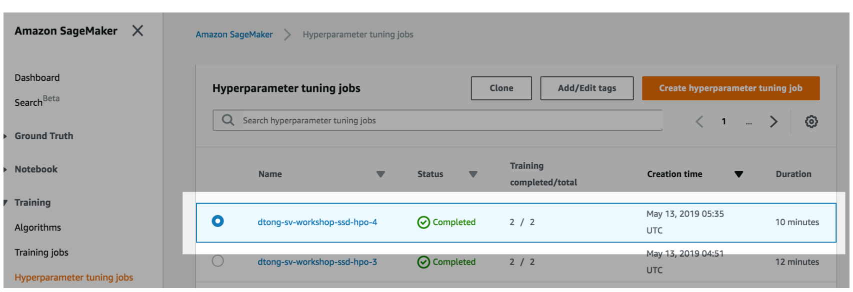
This tuning job will take 10-15 minutes to complete. This is a good time to take a break!

II. Deploy a Real-time Model Inference Endpoint

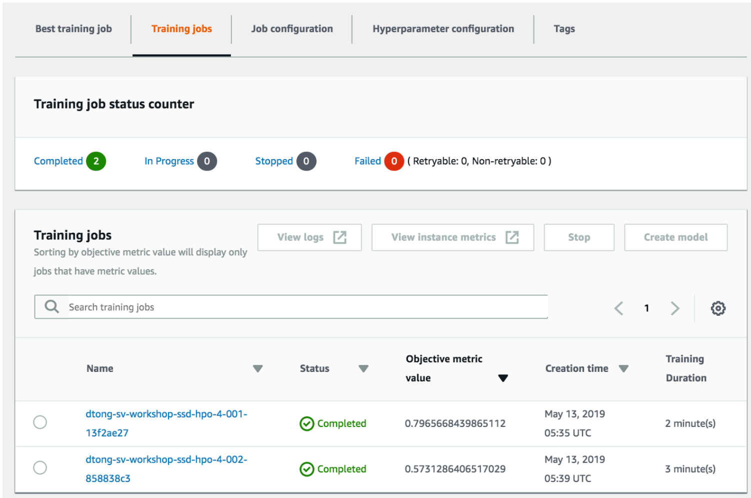
14. Select **Hyperparameter tuning jobs** on the navigation panel if you aren't already on the Hyperparameter tuning jobs page.



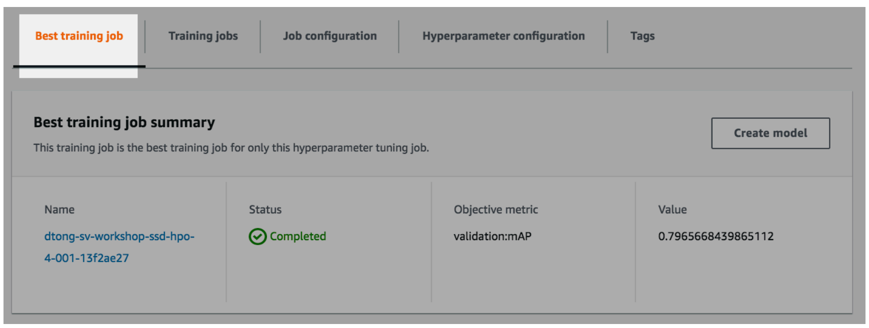
15. Select the hyperparameter tuning job that you created as shown in the screenshot below.



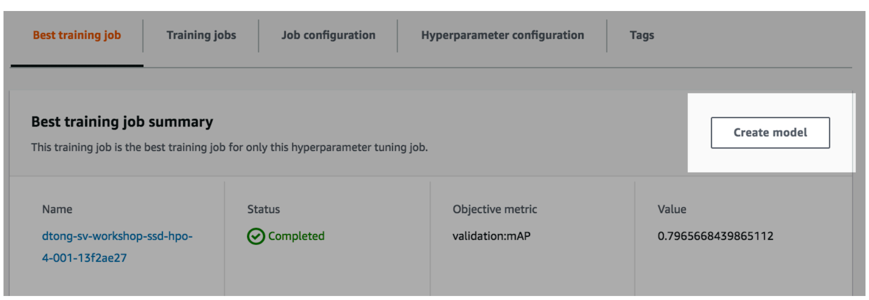
16. Scroll down. You should see something similar to the screenshot below. The tuning job should have two training jobs with corresponding objective metrics.



Click on **Best training job** on the sub tab.

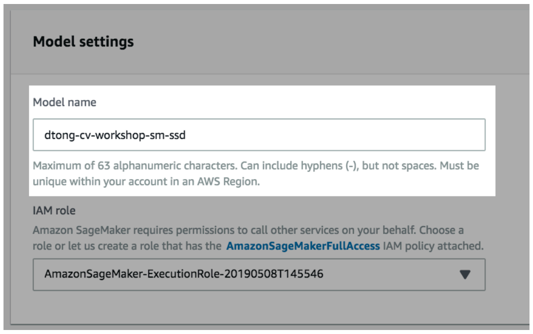


Click on the **Create model** button. This will register the best model produced by the hyperparameter tuning job, and make it available for deployment via Amazon SageMaker.

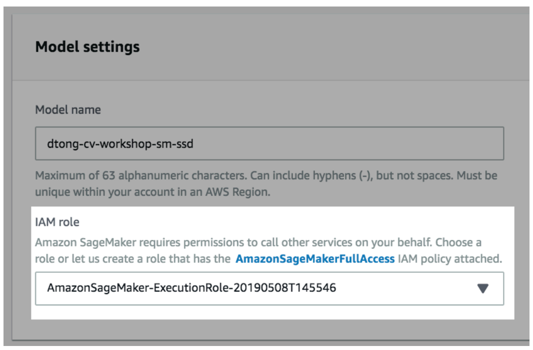


17. Work through the model configuration wizard.

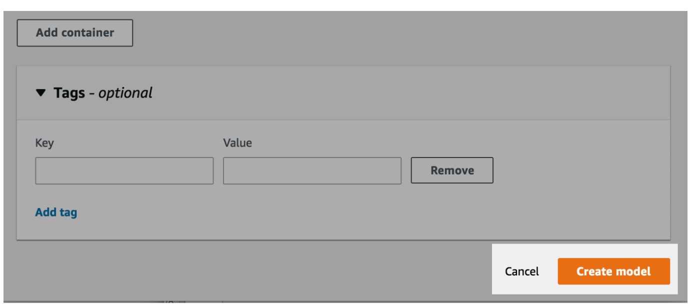
    Provide a **unique name** for your model.



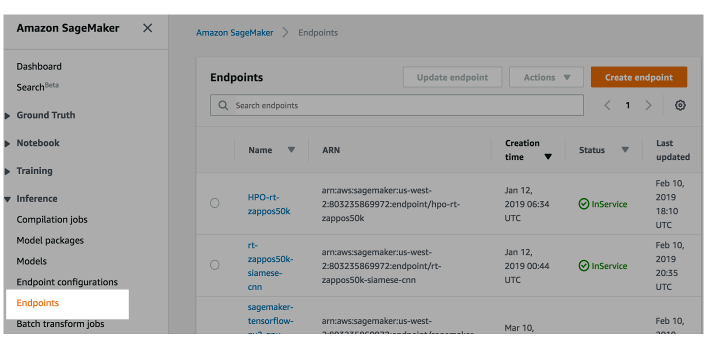
Select the IAM role that you've been using to provide the model access to Amazon SageMaker and S3 resources.



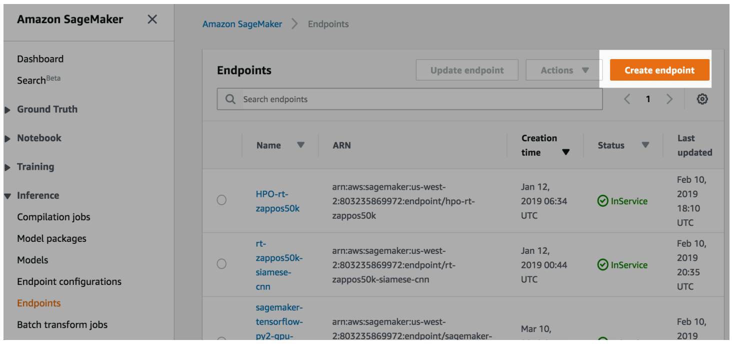
Click on the **Create model** button. The model is now available for deployment.



18. Switch to the **Endpoints** page by selecting **Endpoints** from the navigation pane:

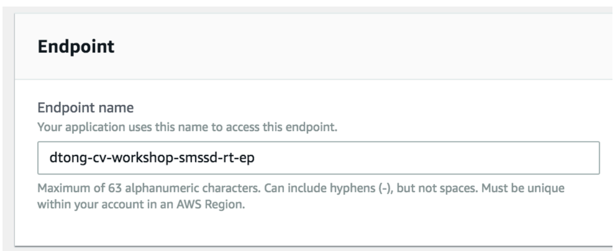


19. Click on the **Create endpoint** button.

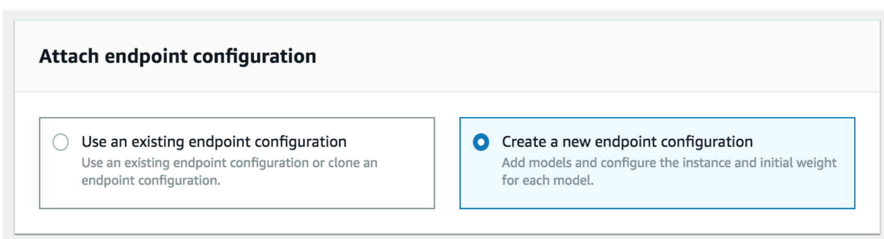


20. Work through the Endpoint configuration form.

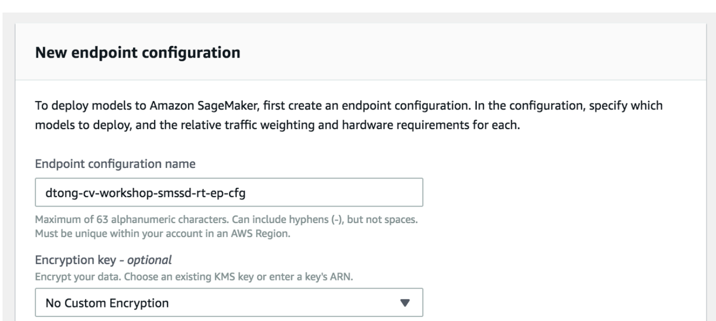
Provide a **unique name** for your endpoint.



Select **"Create a new endpoint configuration."**

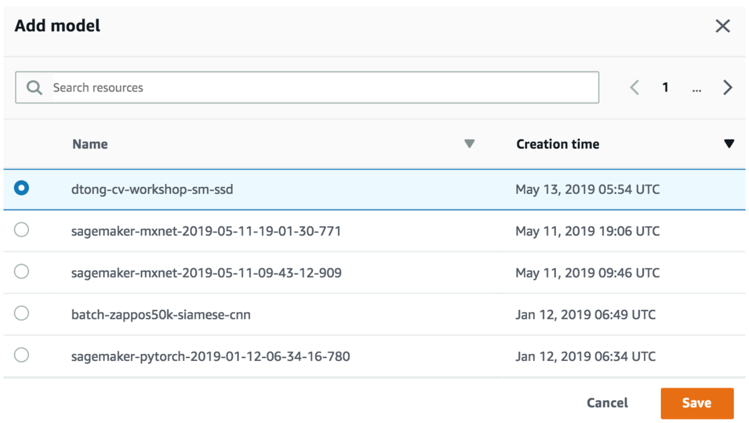


Provide a **unique name** for your endpoint configuration.

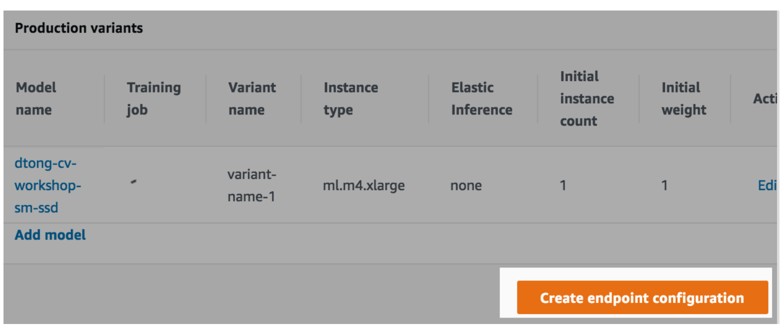


Click on the Add model link.

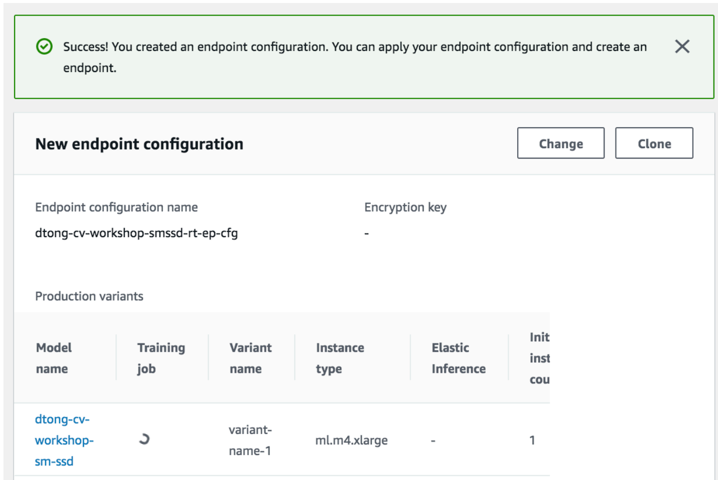
Select the model that you registered previously, and click on the **Save** button.



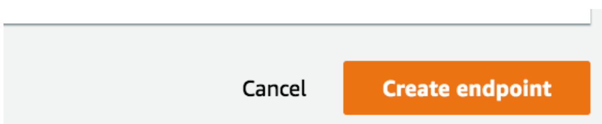
Click on the **Create endpoint configuration** button to finalize the creation of your endpoint configuration.



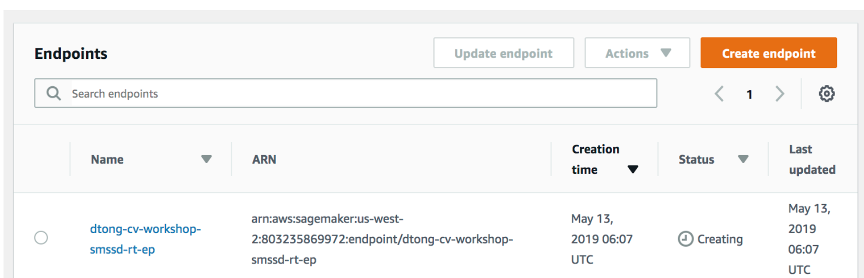
You should see confirmation that you endpoint configuration was successfully created.



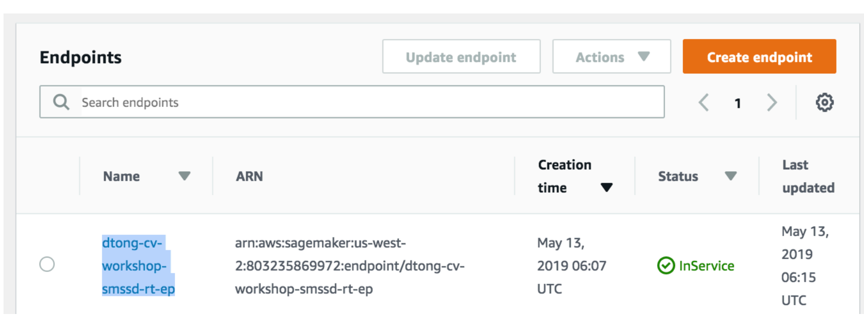
Click on **Create endpoint** to launch your inference endpoint.



You should see your endpoint in a “Creating…” status.



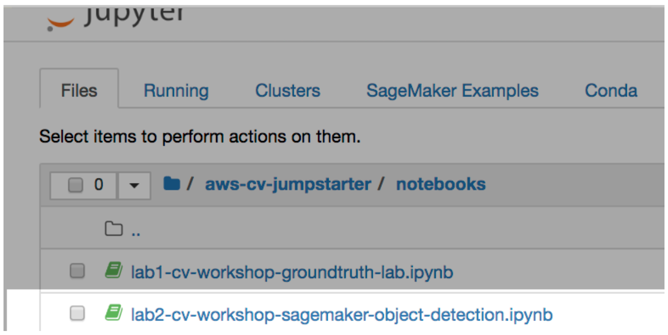
It takes approximately 5 minutes for the endpoint to launch.



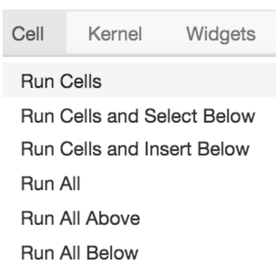
III. Test your Endpoint

In this section we're going to run some code in a Jupyter notebook to invoke our endpoint and visualize results.

21. **Log back on to** your SageMaker notebook instance. **Launch** the Jupyter notebook for **Lab 2** as shown in the screenshot below:



22. **Run** through each cell of the notebook.



There are a couple of places where you have to modify the script.

In the first cell, update the BUCKET variable with the name of the S3 bucket that you created in Lab 1.

```
BUCKET = '<<REPLACE WITH YOUR BUCKET NAME>>'
```

For instance,

```
BUCKET = 'dtong-cv-jumpstarter-workshop'
```

In the cell where you instantiate your real-time endpoint object, provide the name of the endpoint that you created in this lab.

```
RT_ENDPOINT_NAME = '<<REPLACE WITH THE NAME OF YOUR ENDPOINT>>'
```

For instance,

```
RT_ENDPOINT_NAME = 'dtong-cv-workshop-smssd-rt-ep'
```

23. **View the results.** The final cell will output images with predicted bounding boxes and classification.

    Results will vary. Again, due to the purpose of demonstration, don't expect production grade results. In practice, we will need more data, training time and tuning.

```
In [54]: from os import walk

         img_dir = './images'
         for (dirpath, dirnames, filenames) in walk(img_dir):
             for f in filenames :
                 show_bird_prediction(os.path.join(img_dir,f), od_endpoint.endpoint)
```

```
bird,0.6403394341468811
Number of detections: 1
```