How to use Intel® DAAL K-Means Clustering via SageMaker web interface

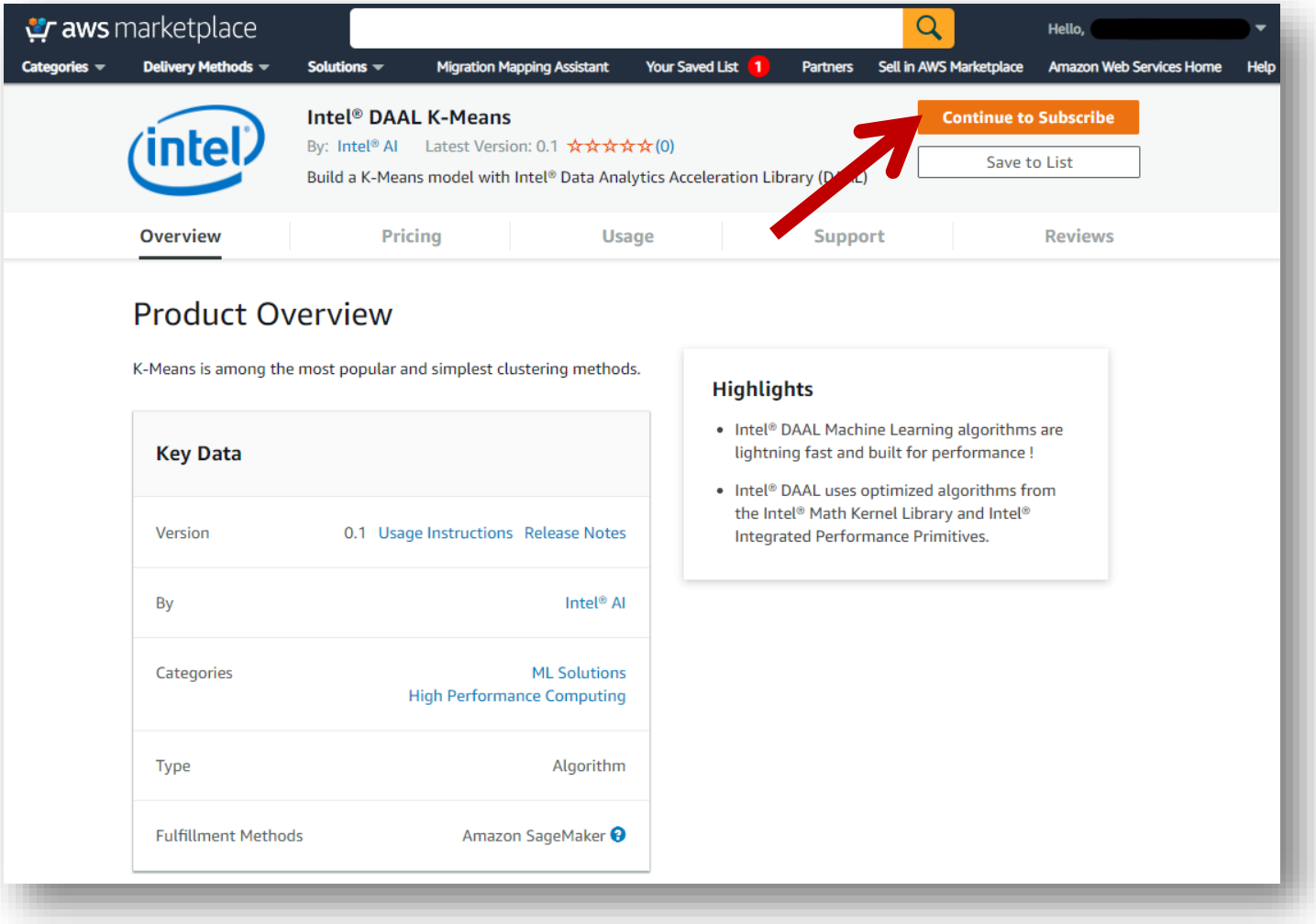
Description of algorithm:

K-Means is among the most popular and simplest clustering methods. It is intended to partition a data set into a small number of clusters such that feature vectors within a cluster have greater similarity with one another than with feature vectors from other clusters. Each cluster is characterized by a representative point, called a centroid, and a cluster radius. In other words, the clustering methods enable reducing the problem of analysis of the entire data set to the analysis of clusters. There are numerous ways to define the measure of similarity and centroids. For K-Means, the centroid is defined as the mean of feature vectors within the cluster.

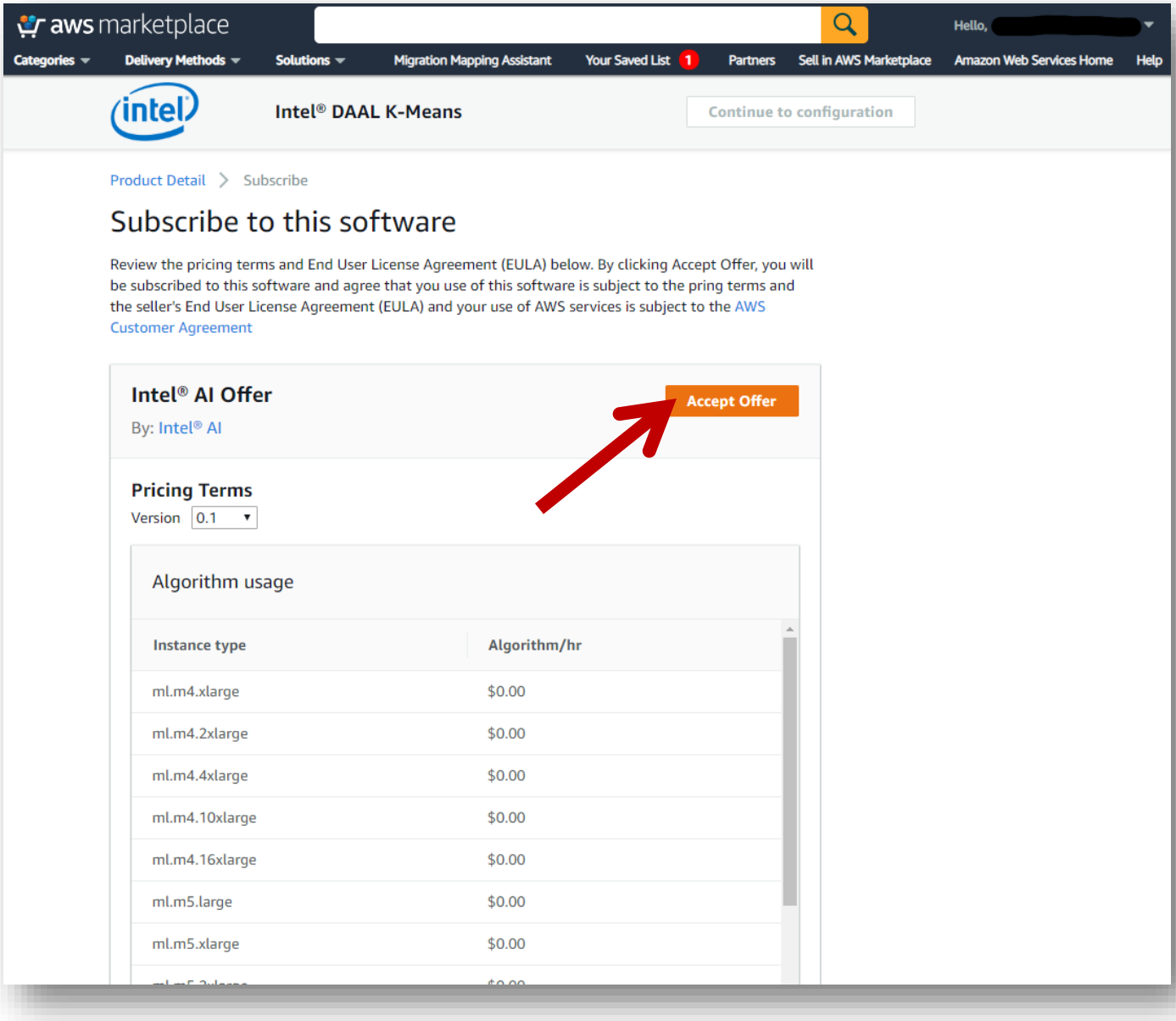
Intel® DAAL developer guide Intel® DAAL documentation for K-Means

Instruction:

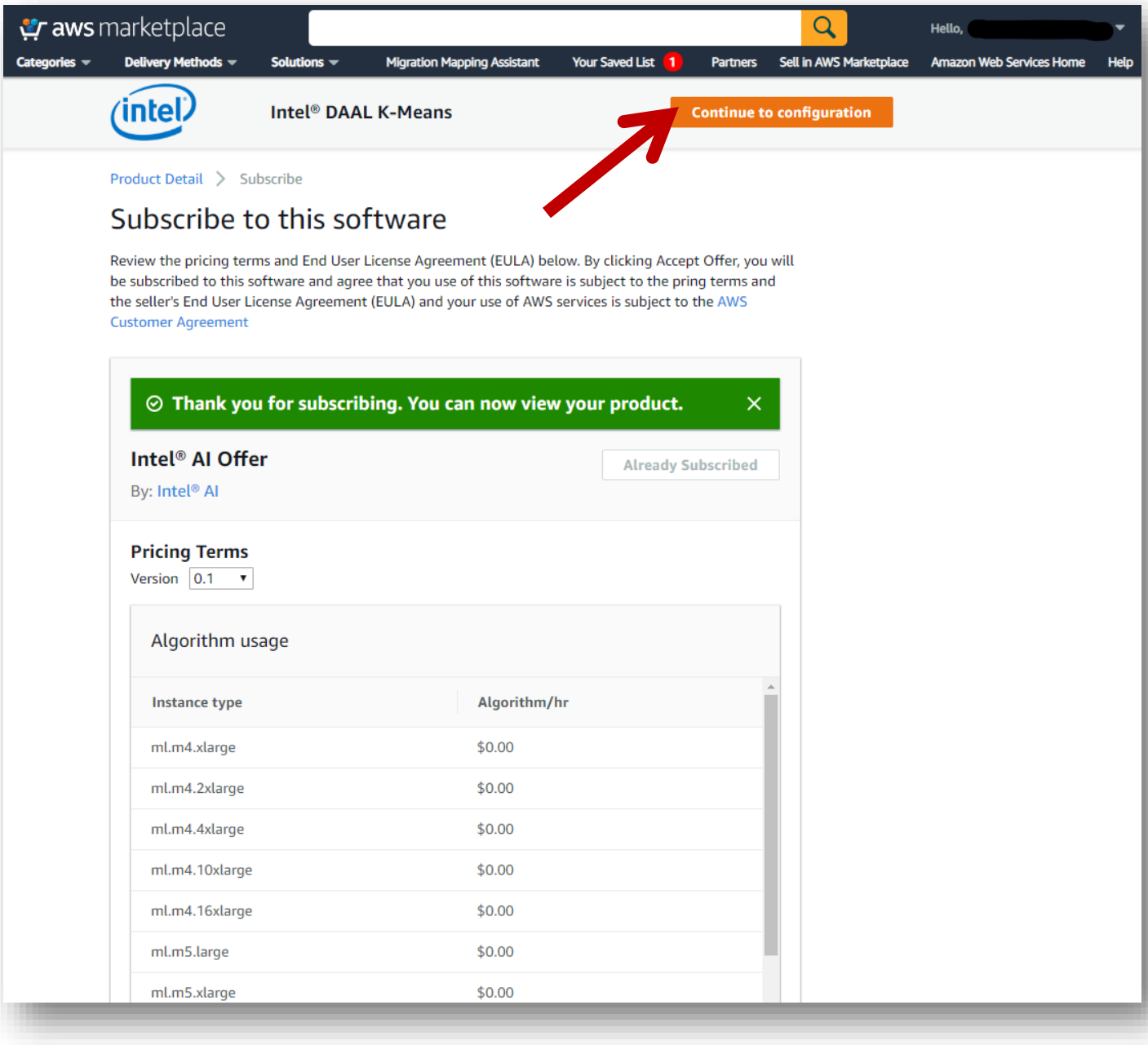
1. Visit page on SageMaker Marketplace and click “Continue to Subscribe”



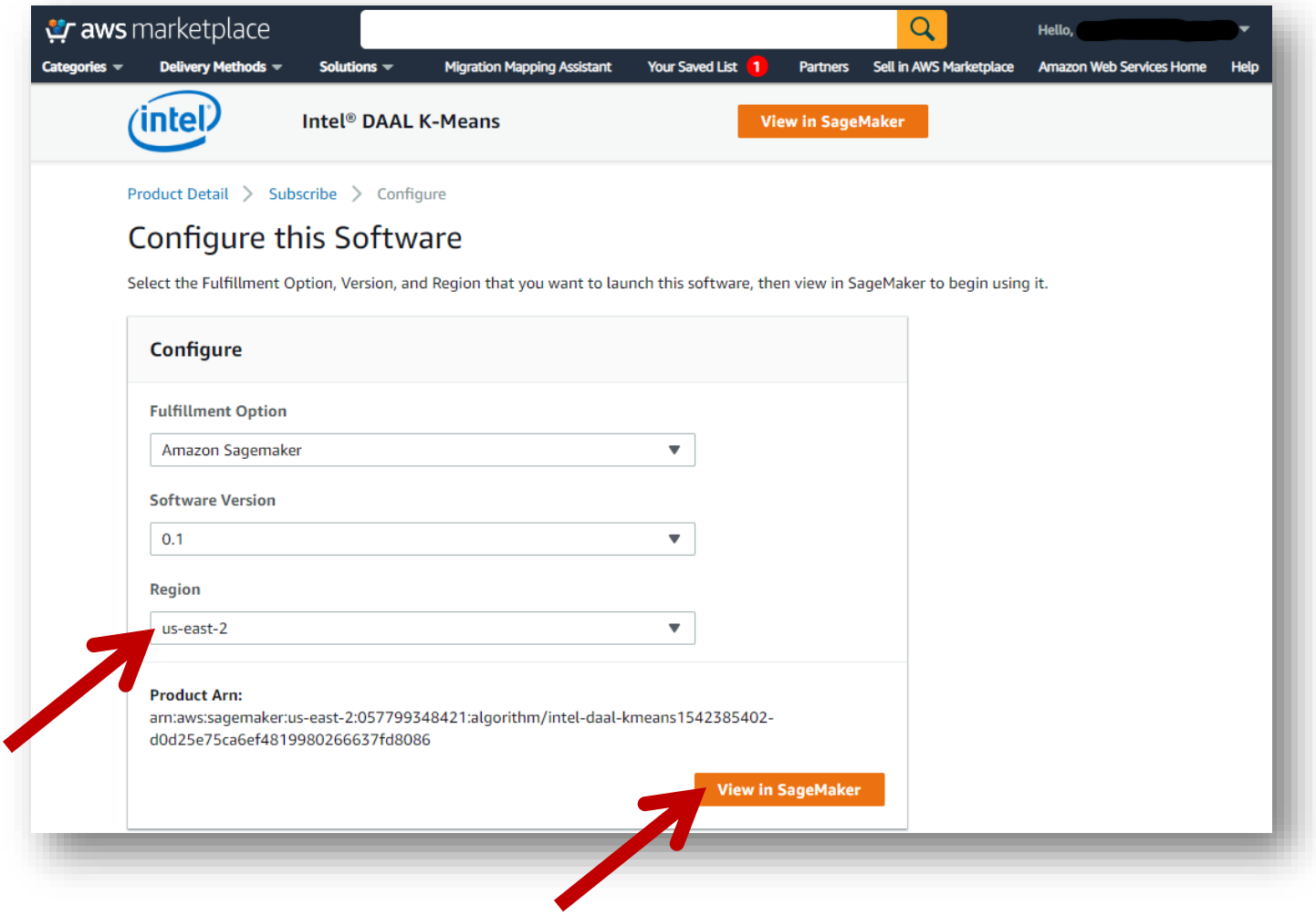
2. Click “Accept Offer” if you agree with EULA at end of page. If you already subscribed on algorithm on Marketplace this step will be skipped.



3. Click “Continue to configuration”



4. Choose the Region and click “View in SageMaker”



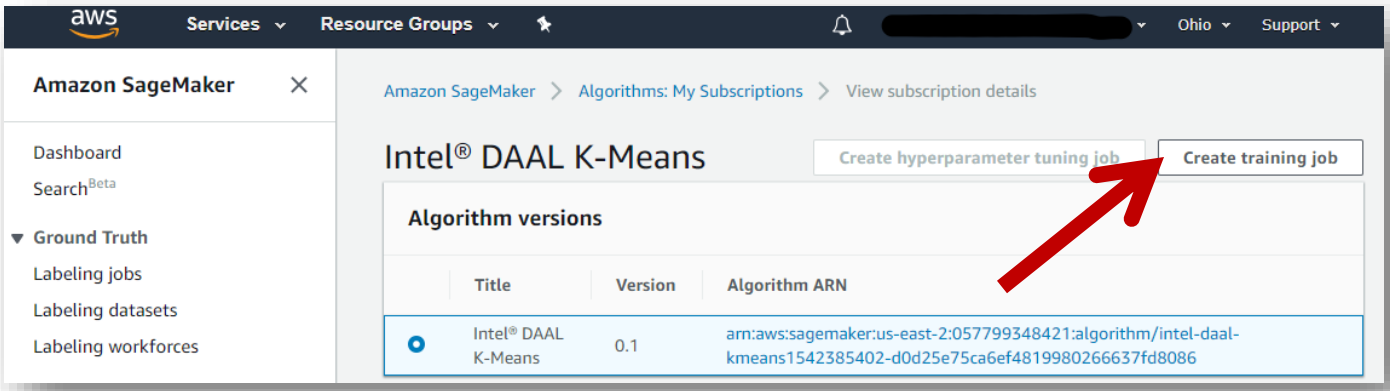
K-Means algorithm on AWS SageMaker is divided into two stages: training job and getting inference from endpoint.

Training job is computing clusters centroids and other values from provided training data.

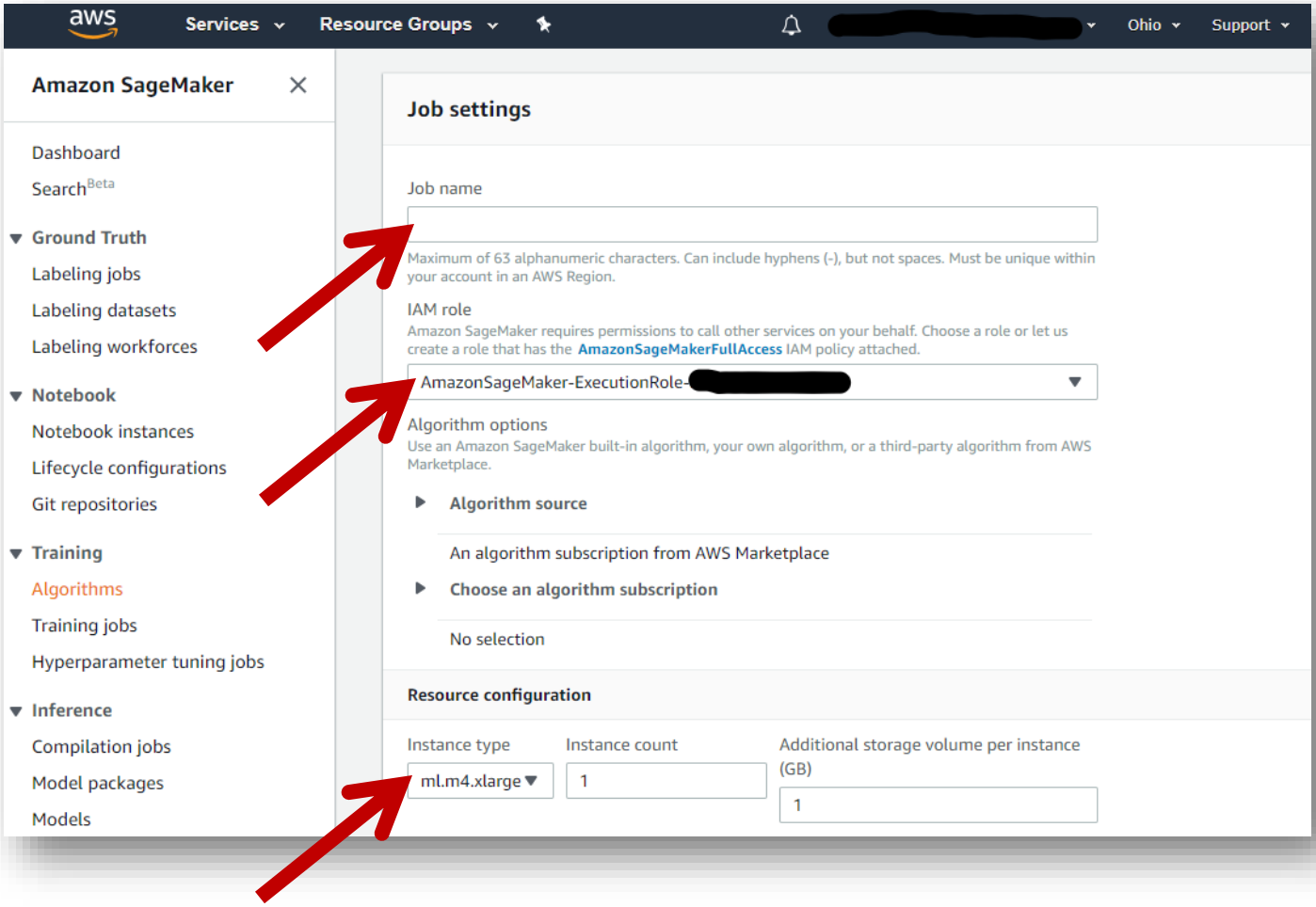
After that, you should create model with computed values and endpoint based on it.

Sending data to endpoint gives you numbers of clusters for provided data in response.

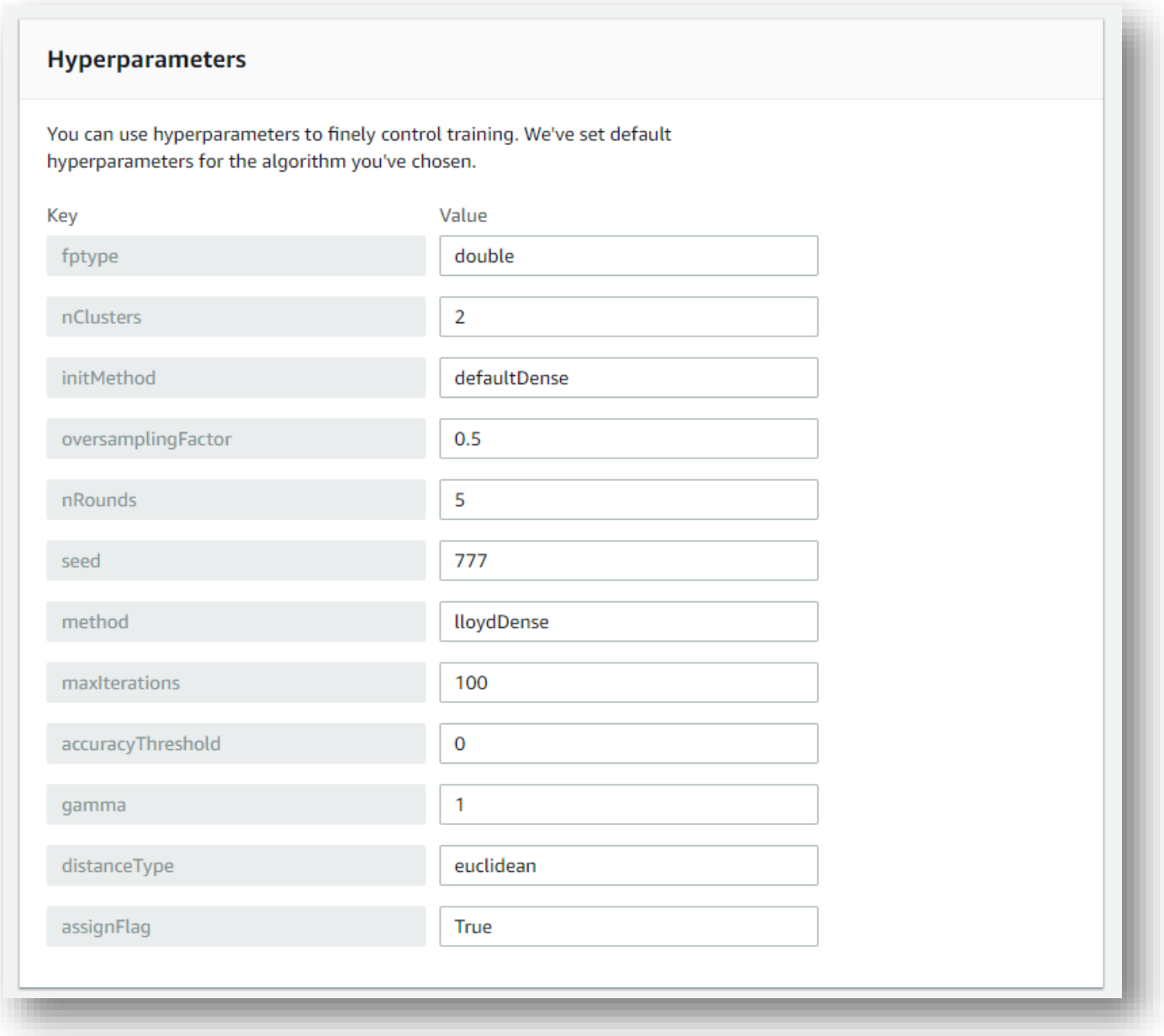
5. Select needed algorithm version and click “Create training job”



6. Type job name, select IAM role and instance type

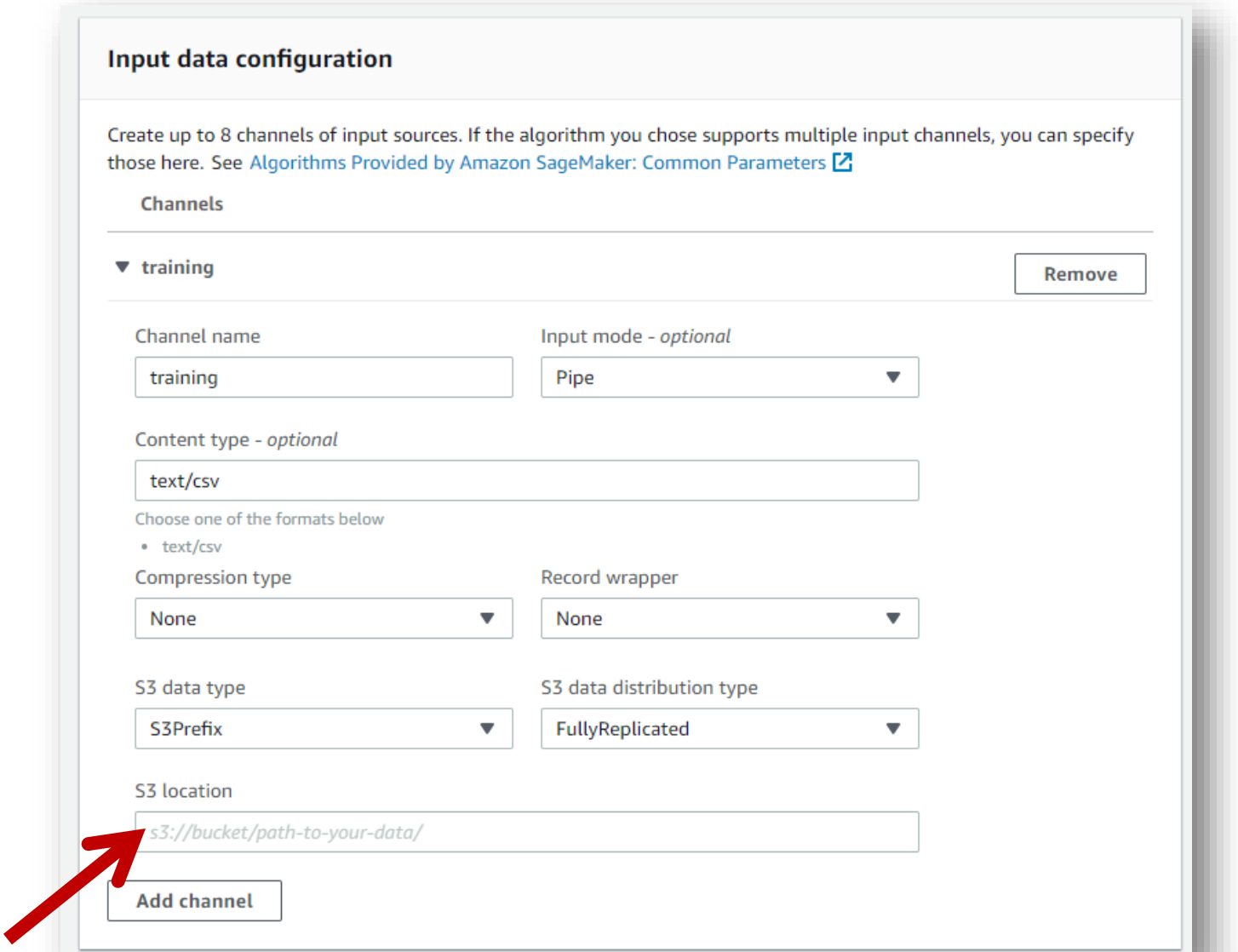


7. Choose hyperparameters

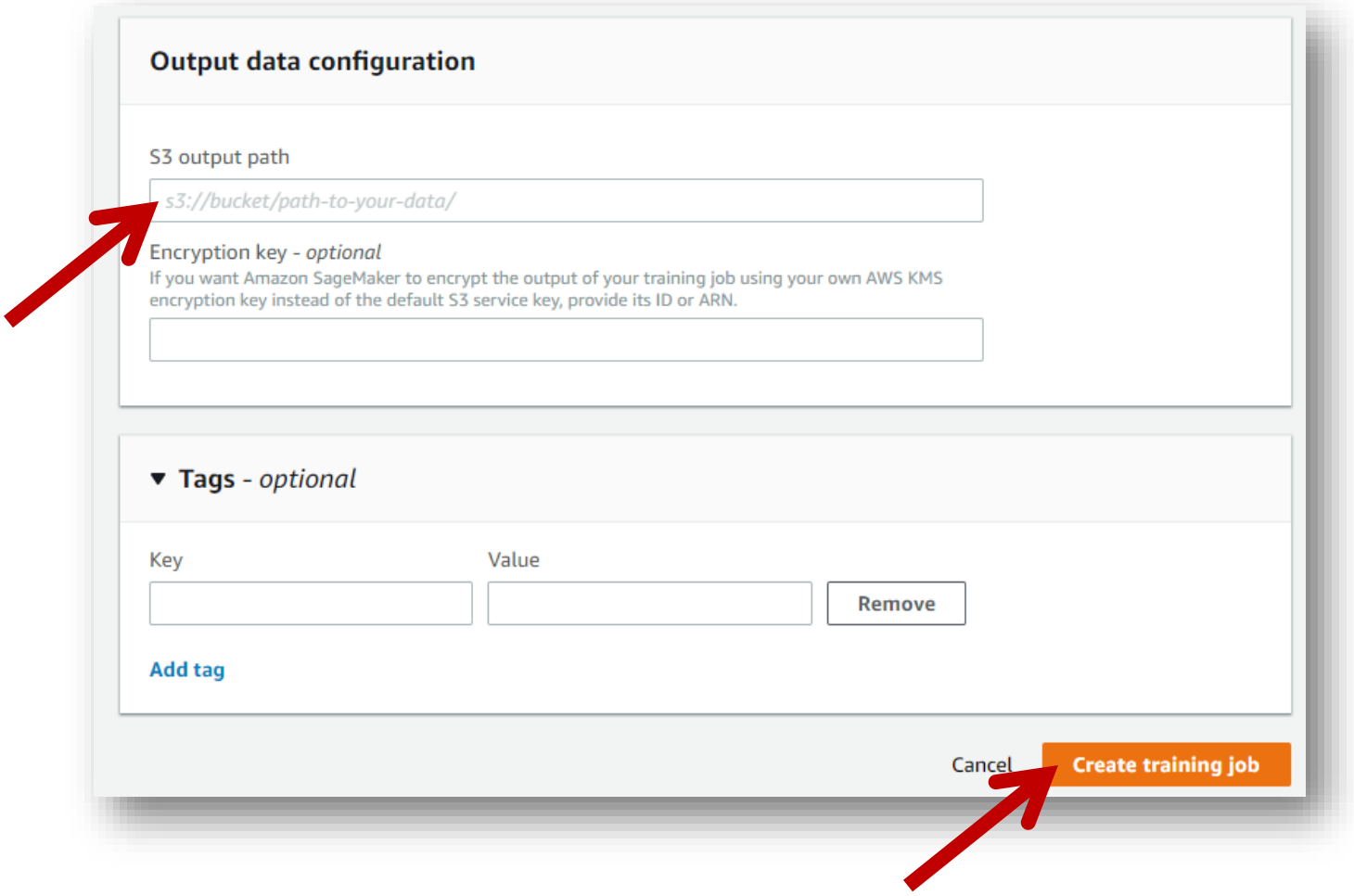


Parameter name	Type	Default value	Description
fptype	str	"double"	The floating-point type that the algorithm uses for intermediate computations. Can be "float" or "double"
nClusters	int	2	The number of clusters
initMethod	str	"defaultDense"	Available initialization methods for K-Means clustering: defaultDense - uses first nClusters points as initial clusters, randomDense - uses random nClusters points as initial clusters, plusPlusDense - uses K-Means++ algorithm; parallelPlusDense - uses parallel K-Means++ algorithm
oversamplingFactor	float	0.5	A fraction of nClusters in each of nRounds of parallel K-Means++. L=nClusters*oversamplingFactor points are sampled in a round
nRounds	int	5	The number of rounds for parallel K-Means++. (L*nRounds) must be greater than nClusters
seed	int	777	The seed for random number generator
method	str	"lloydDense"	Computation method for K-Means clustering
maxIterations	int	100	The number of iterations
accuracyThreshold	float	0	The threshold for termination of the algorithm
gamma	float	1	The weight to be used in distance calculation for binary categorical features
distanceType	str	"euclidean"	The measure of closeness between points (observations) being clustered. The only distance type supported so far is the Euclidian distance
assignFlag	bool	True	A flag that enables computation of assignments, that is, assigning cluster indices to respective observations

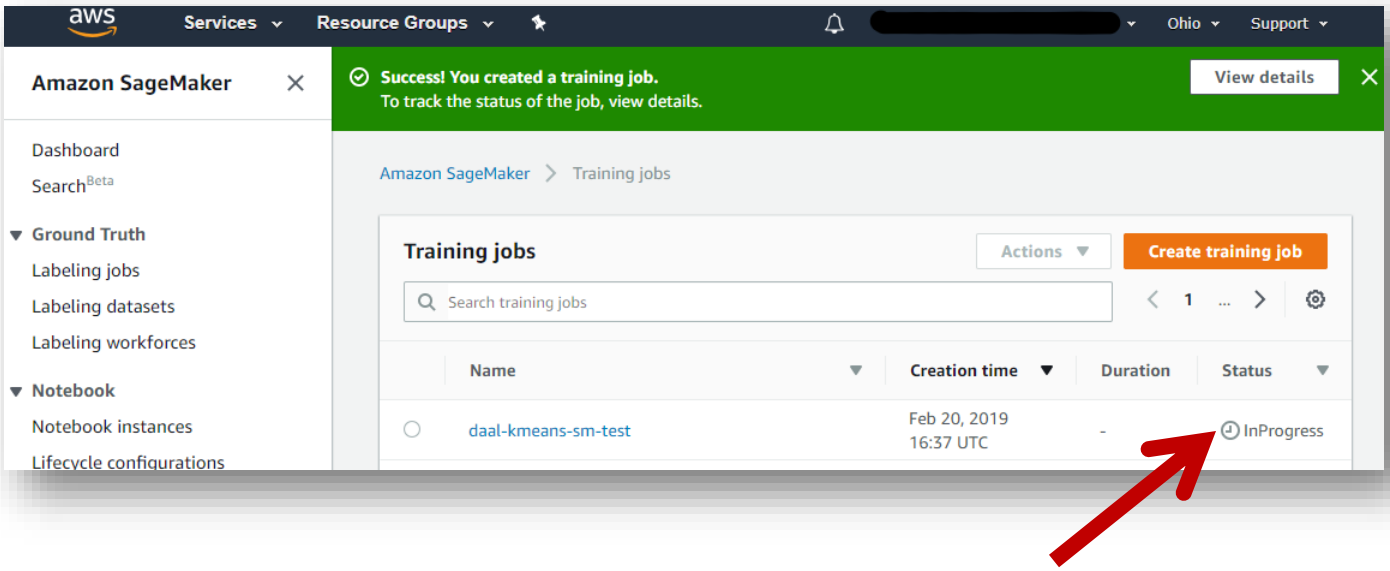
8. Specify S3 location of input data for training

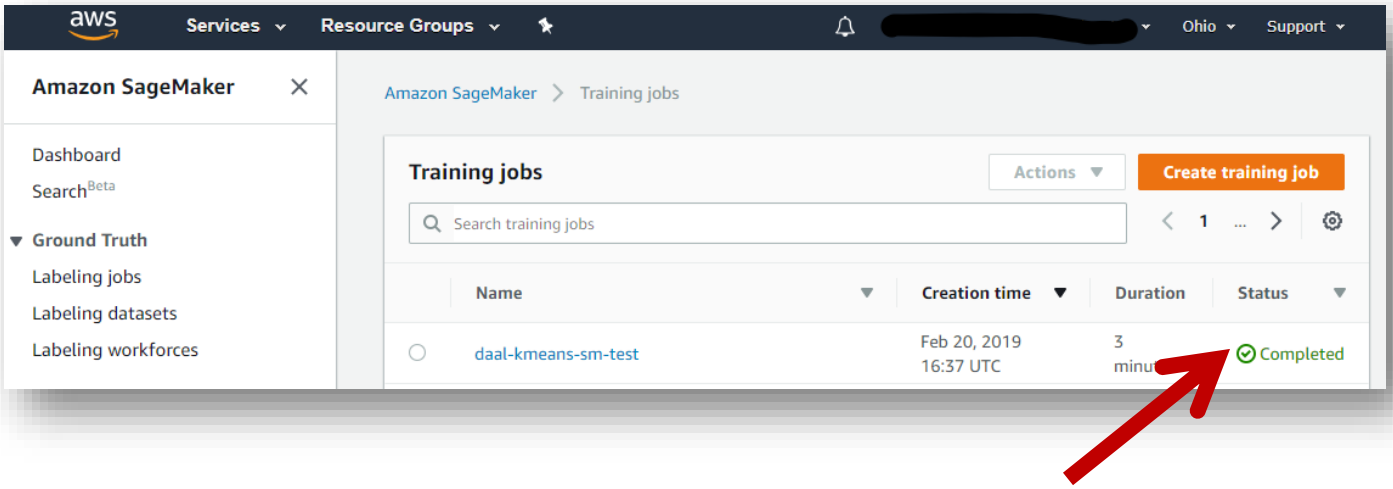


9. Specify S3 output path (model will be stored here) and click “Create training job”

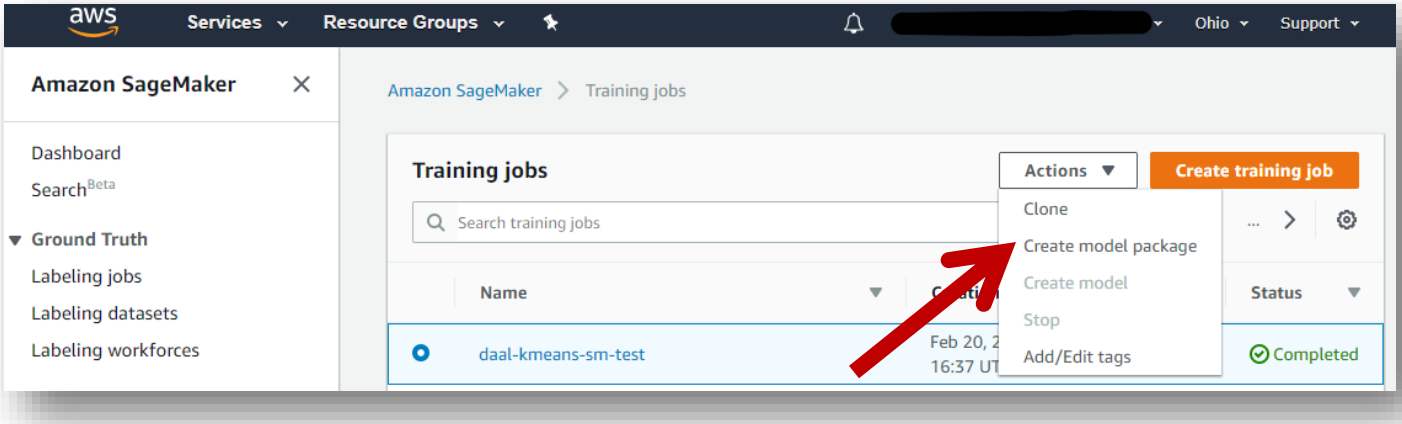


10. Wait until finish of training job

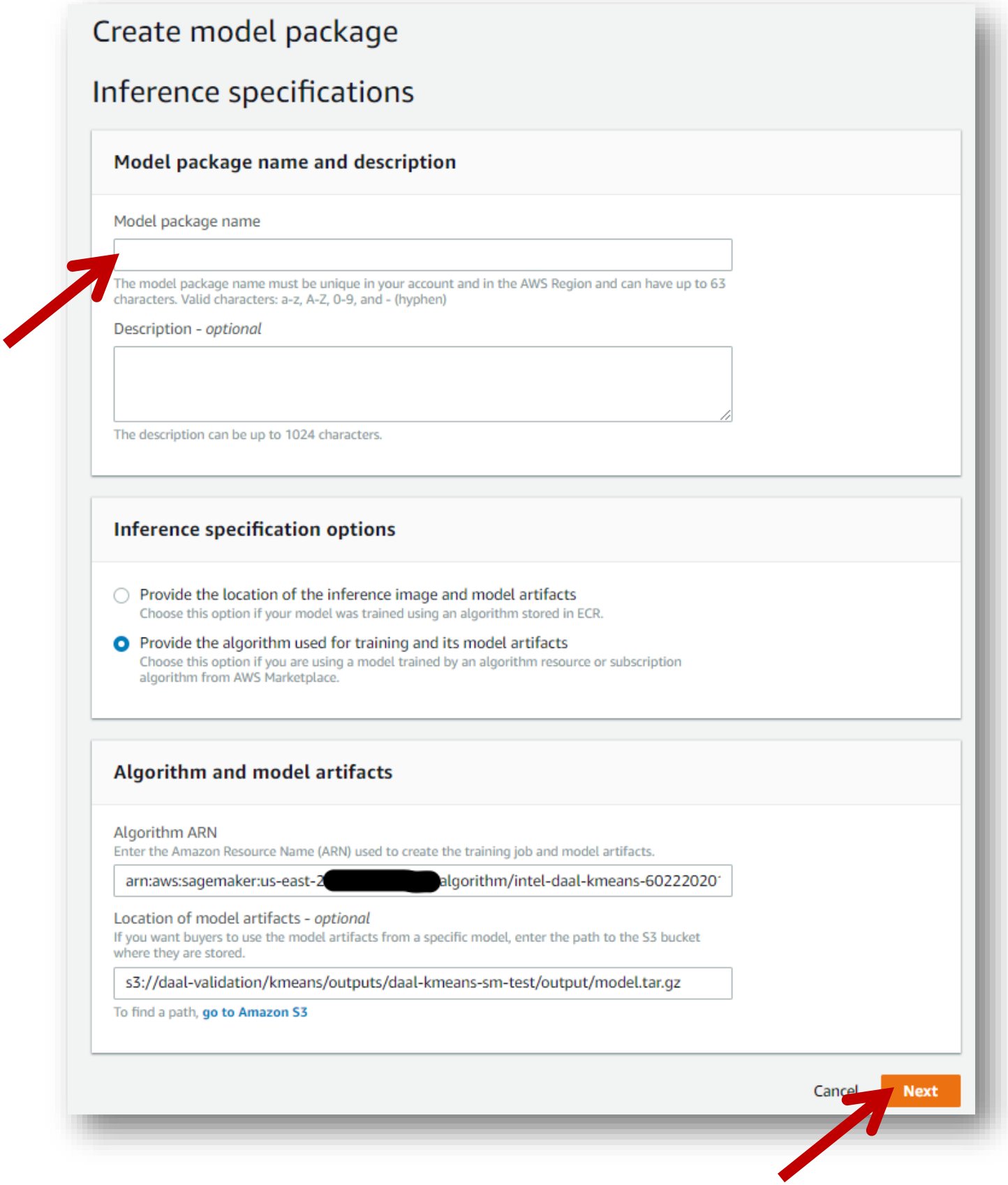




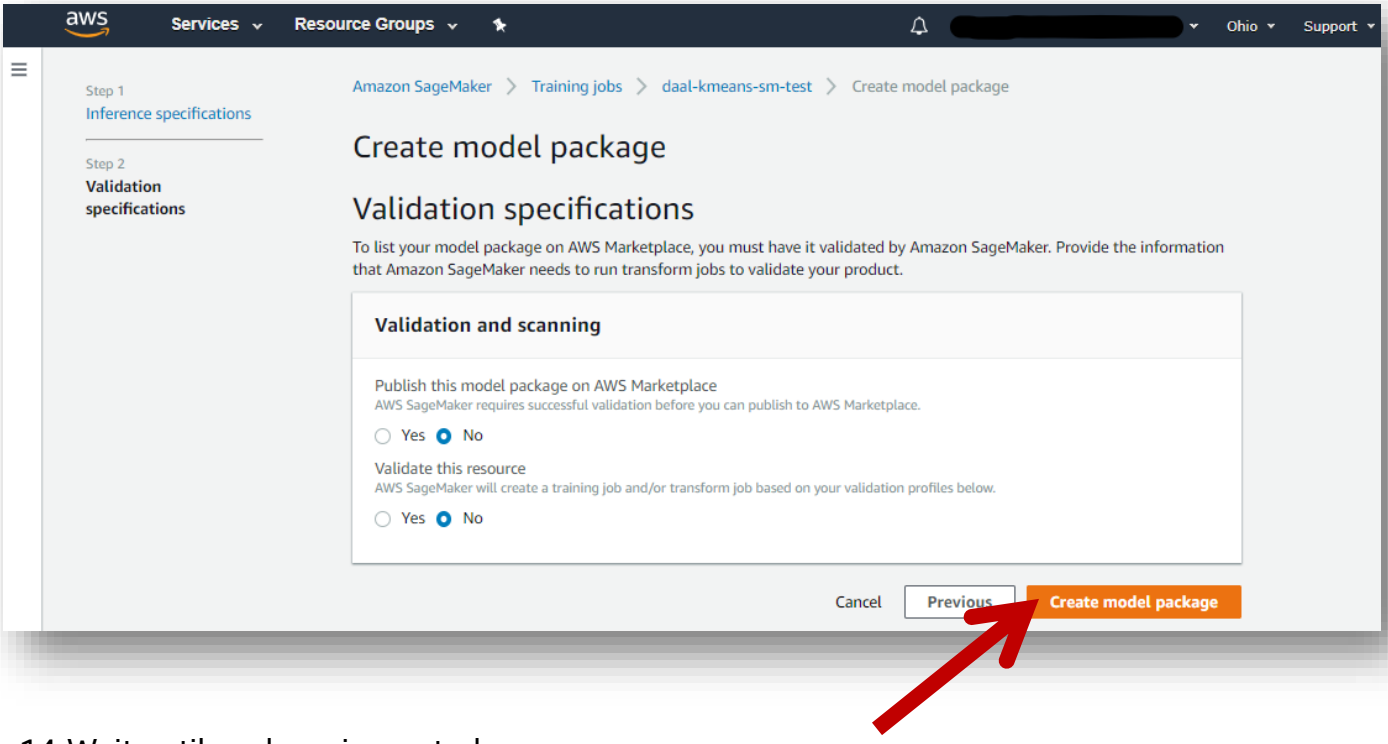
11. Select training job and take action “Create model package”



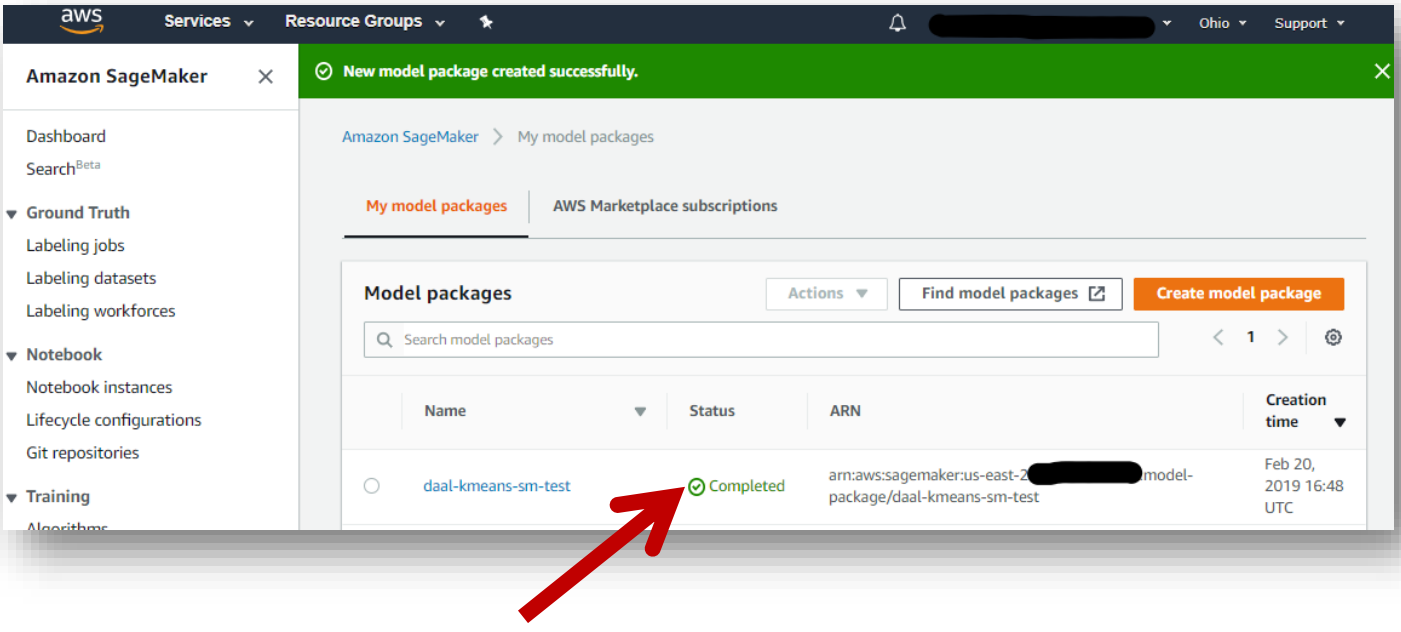
12. Type model package name and click “Next”



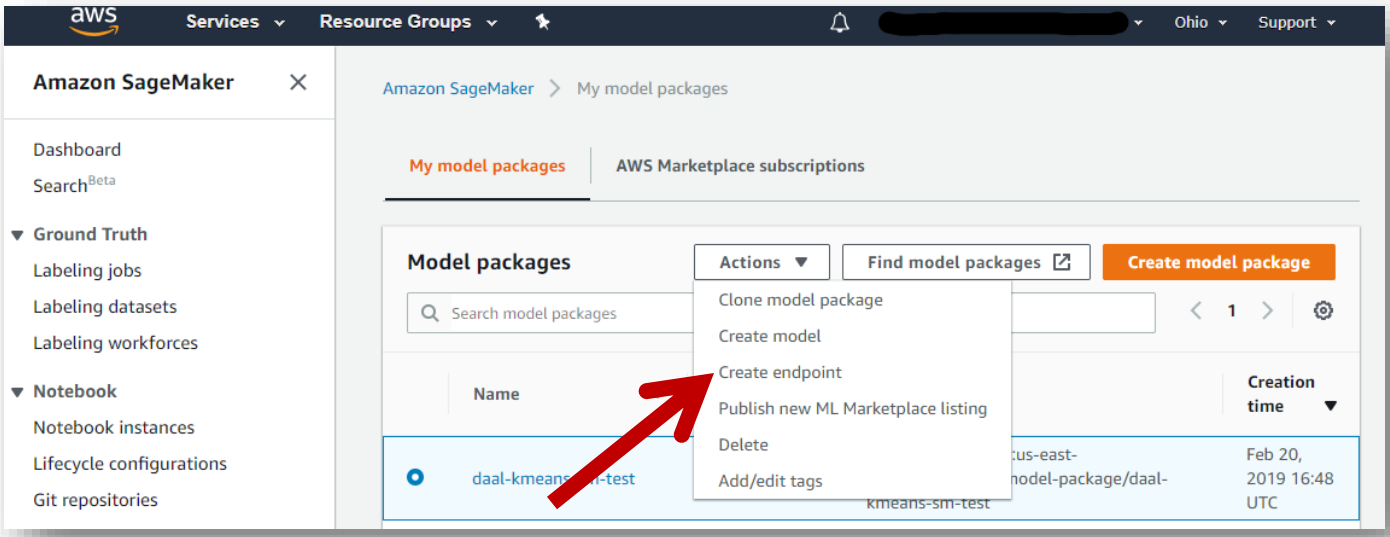
13. Click “Create model package”



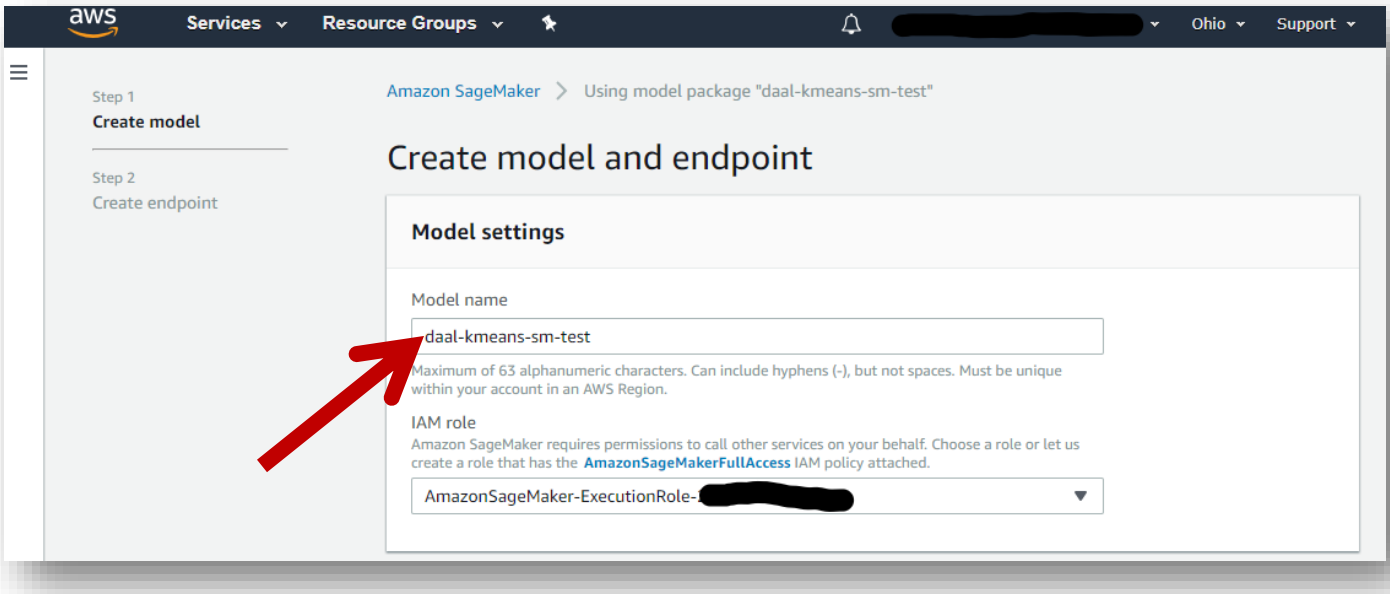
14. Wait until package is created



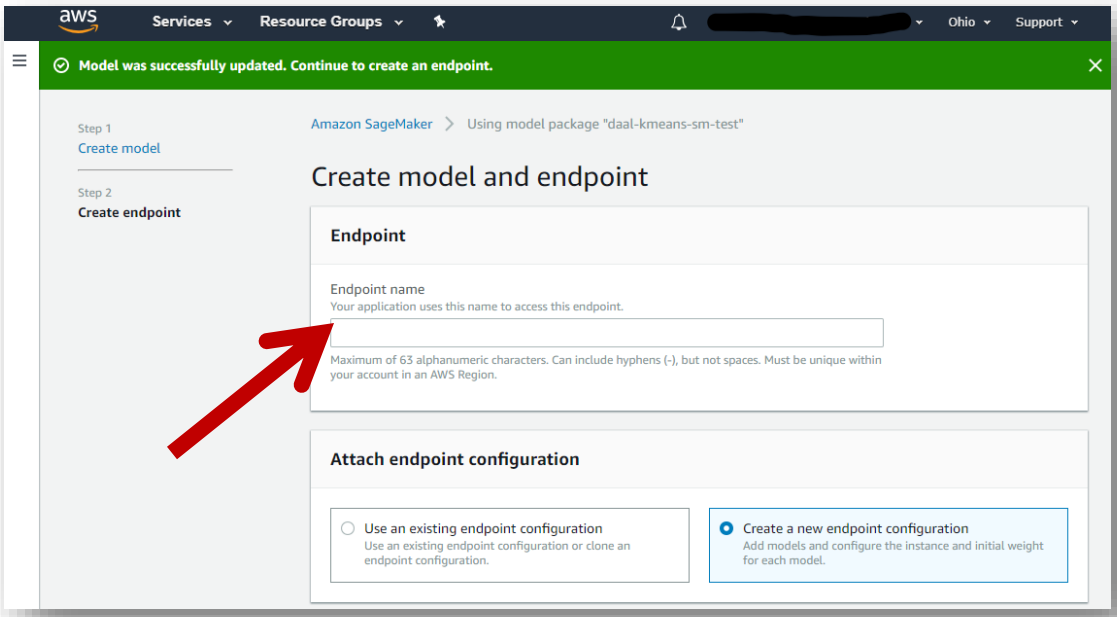
15. Select package and take action “Create endpoint”

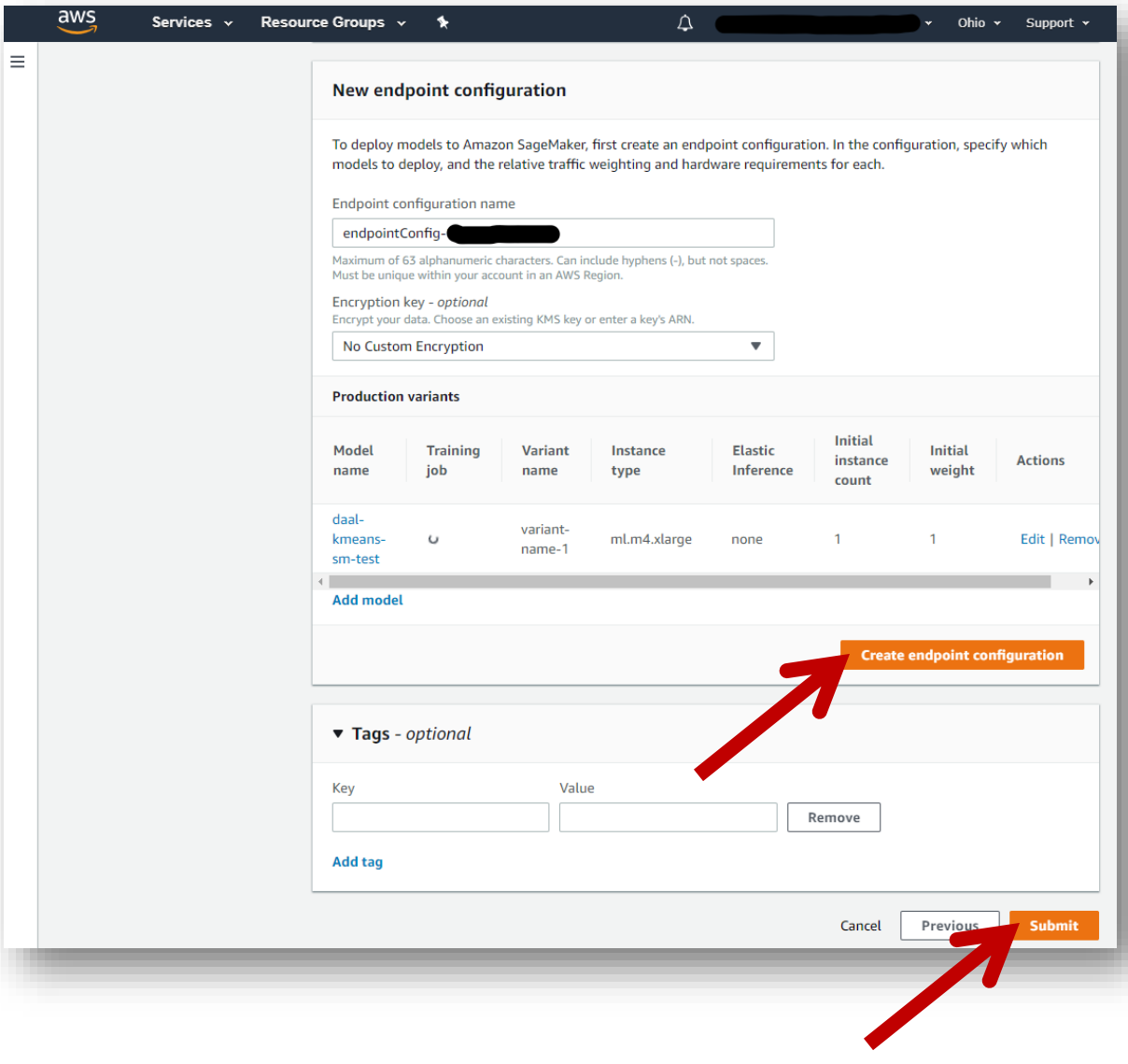


16. Type model name and click “Next”

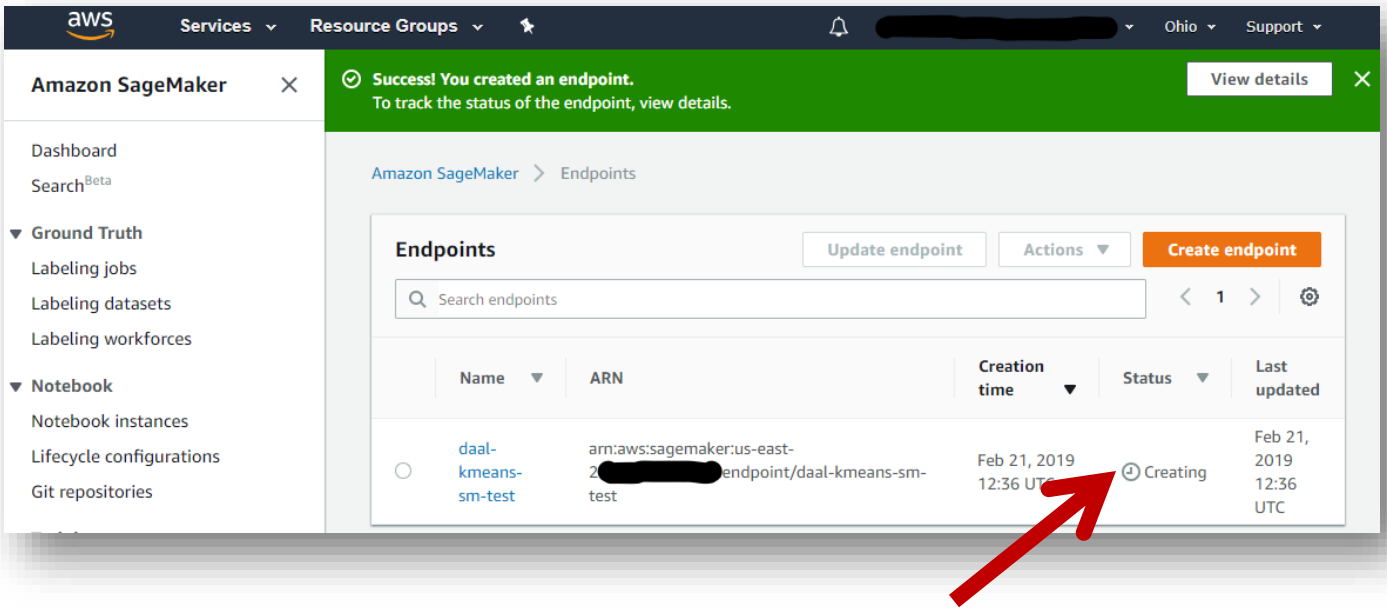


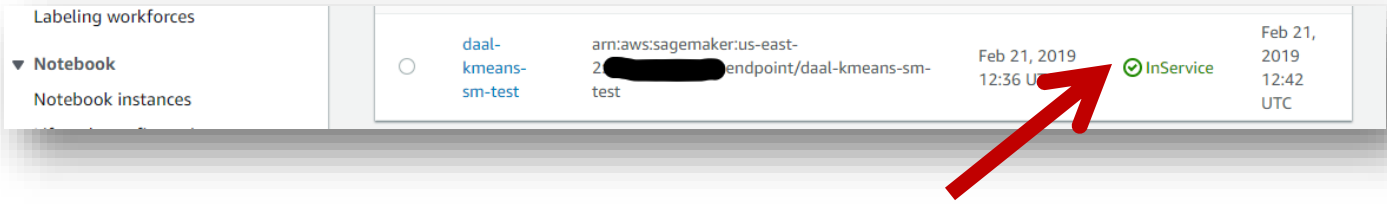
17. Type endpoint name, edit and create endpoint configuration and click “Submit”





18. Wait until endpoint is ready





19. Use AWS CLI to get real-time prediction

Type command:
aws sagemaker-runtime invoke-endpoint --endpoint-name <endpoint-name> --body "$(cat <prediction_data_file_name>)" --content-type text/csv --accept text/csv <output_data_file_name>

```
(base) ubuntu@ip-172-31-22-46:~$ aws sagemaker-runtime invoke-endpoint --endpoint-name daal-knn-sm-test
--body "$(cat probe_data.csv)" --content-type text/csv --accept text/csv output.txt
```

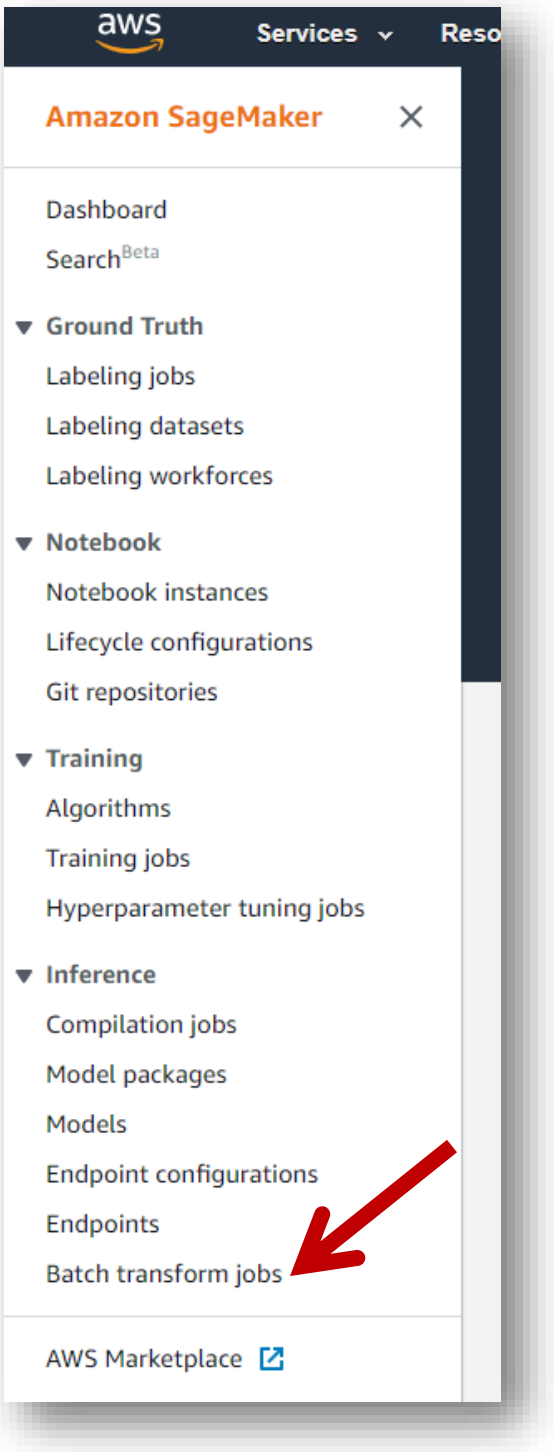
Then, see content of output file:

```
(base) ubuntu@ip-172-31-22-46:~$ cat output.txt
1
4
0
2
3
4
1
1
0
2
4
```

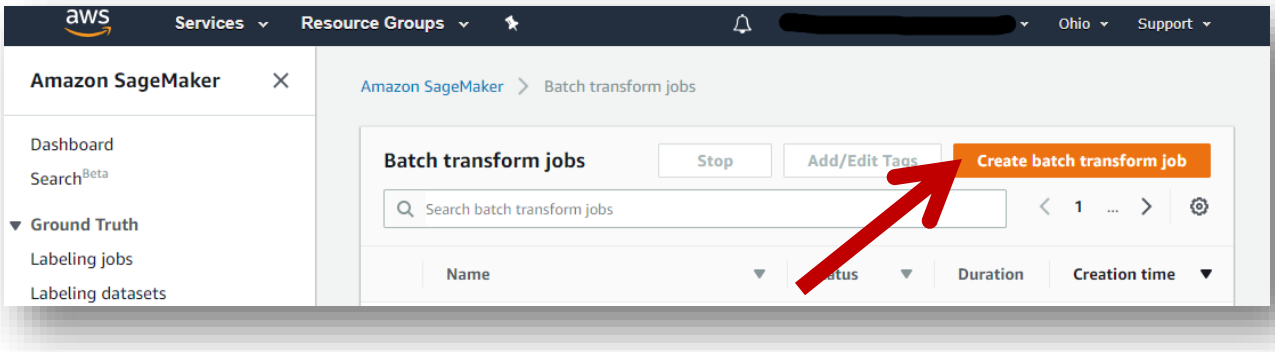
Batch transform job as alternative to endpoint

You can use batch transform job if you need compute predictions once.

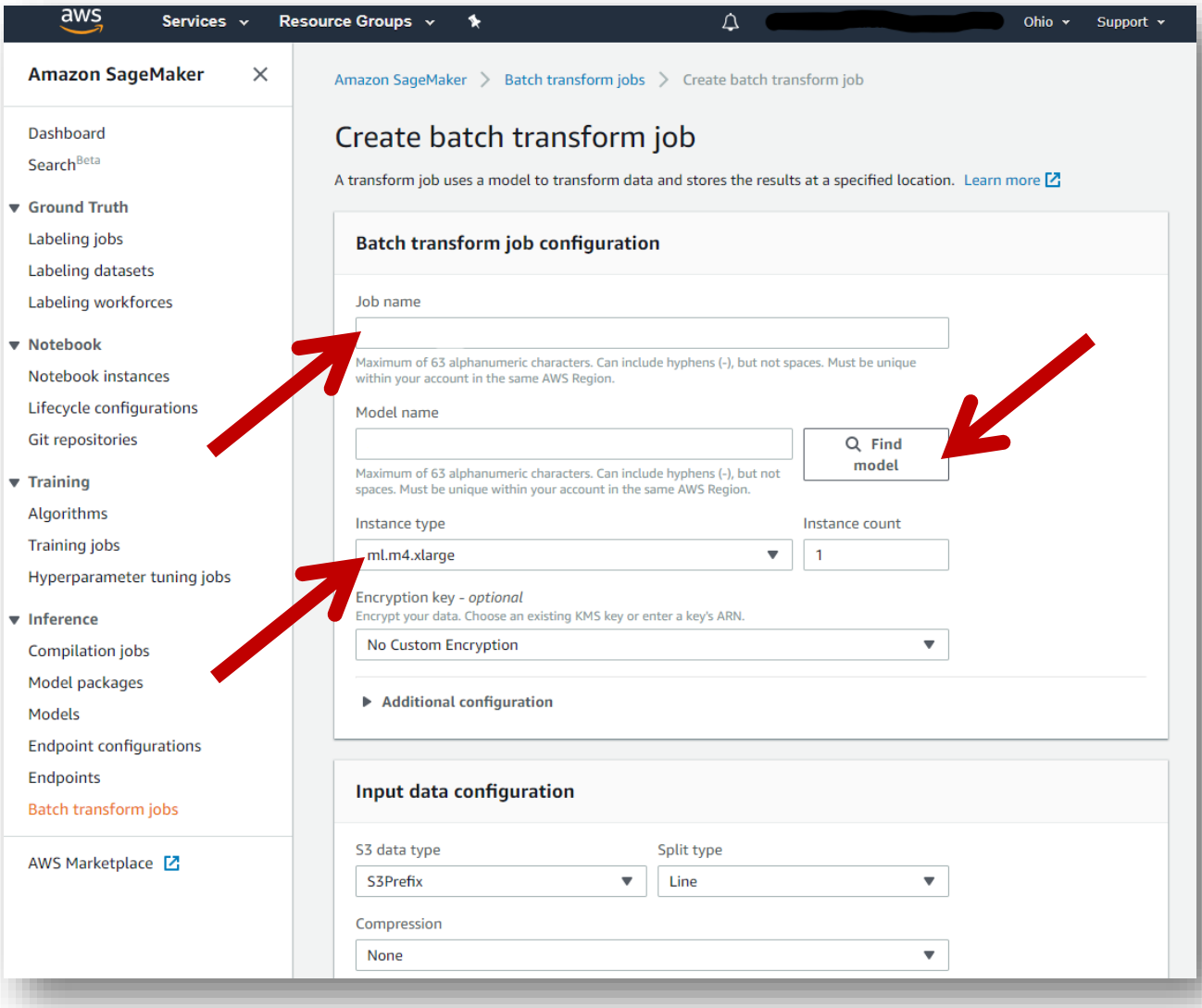
1. Go to “Batch transform job” page



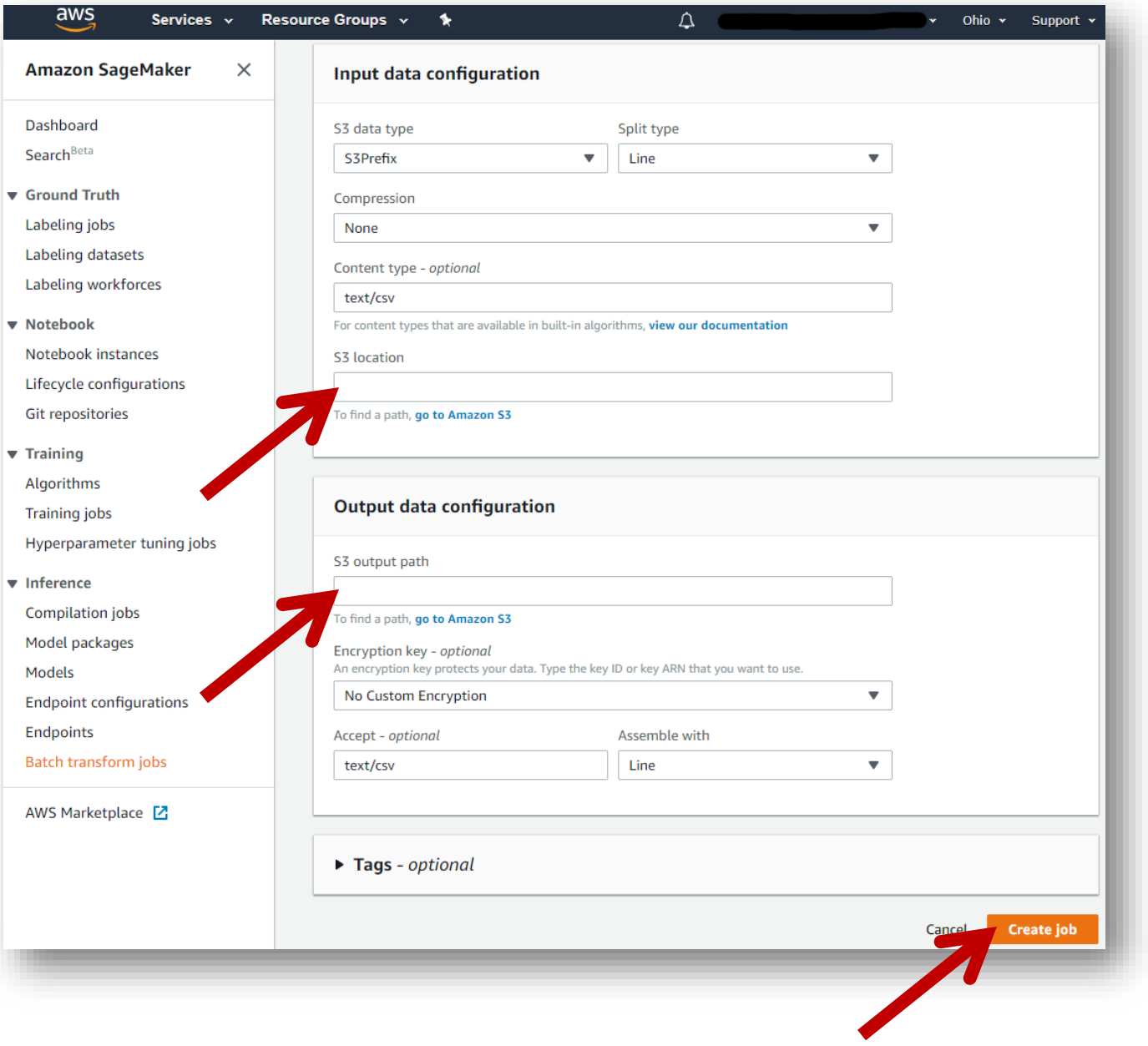
2. Click “Create batch transform job”



3. Enter job name, select previously created model, instance and set instance count to 1



4. Specify S3 location of data for prediction, S3 output path (predictions will be stored here) and click “Create job”



5. Wait until job is completed and find predictions in previously specified S3 output path