Lab3 - CI/CD with Sagemaker

In the third lab you are going to create a full CI/CD pipeline for your machine learning model so you can develop, test and deploy machine learning models in an efficient, safe and repeatable manner.

The AWS infrastructure you will be deploying will be modelled using the principle of Infrastructure as Code. Codifying your infrastructure allows you to treat your infrastructure as just code. You can author it with any code editor, check it into a version control system, and review the files with team members before deploying into production.

The lab consist of the following steps:

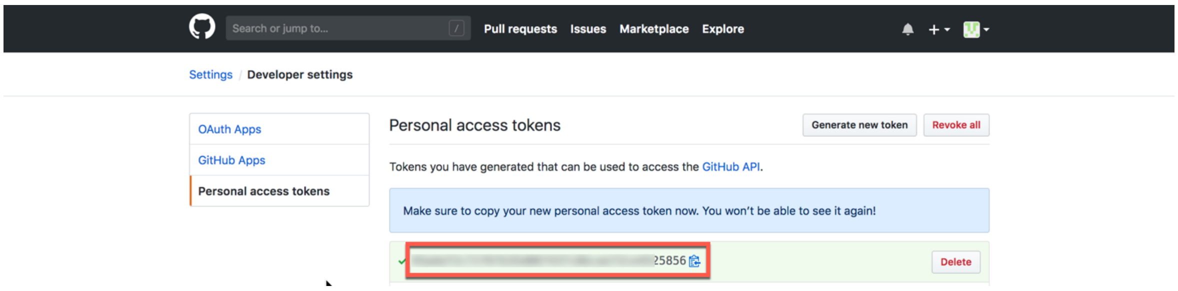
1. Create a GitHub repo and create an OAuth token
2. Create a CI/CD pipeline using AWS CodePipeline
3. Bonus Exercise: Add Sagemaker Hyperparameter tuning to your Sagemaker training

Create a GitHub repo and create an OAuth token

1. To integrate your continous delivery pipeline with GitHub, you will use OAuth tokens. Go to GitHub's Token Settings to generate your token and ensure you enable the following two scopes:

- **repo**, which is used to read and pull artifacts from public and private repositories into a pipeline
- **admin:repo_hook**, which is used to detect when you have committed and pushed changes to the repository

  Make sure to copy your new personal access token now as you will need it later.



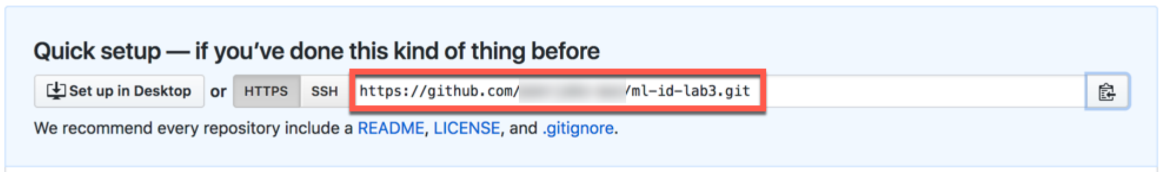
Personal access tokens

Tokens you have generated that can be used to access the GitHub API.

Make sure to copy your new personal access token now. You won't be able to see it again!

[redacted]25856

2. You will need a GitHub repo for your code. In GitHub create a new repository called "ml-id-lab3" and make sure it is public. Do not initialize with a README file and do not add any .ignore or license file. Copy your GitHub repo URL as you will need it later.



Quick setup — if you've done this kind of thing before

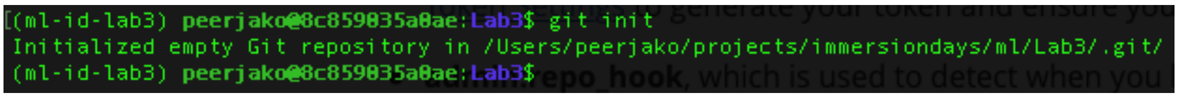
https://github.com/[redacted]/ml-id-lab3.git

We recommend every repository include a README, LICENSE, and .gitignore.

3. Open up a terminal (or cmd on Windows) and change directory into the directory where you extracted the Immersion Day material earlier. Now initialize your git repository by running the following command lines:

```
cd Lab3
git init
```

The output will look something like this:



```
[(ml-id-lab3) peerjako@8c859035a8ae:Lab3$ git init
Initialized empty Git repository in /Users/peerjako/projects/immersiondays/ml/Lab3/.git/
(ml-id-lab3) peerjako@8c859035a8ae:Lab3$
```

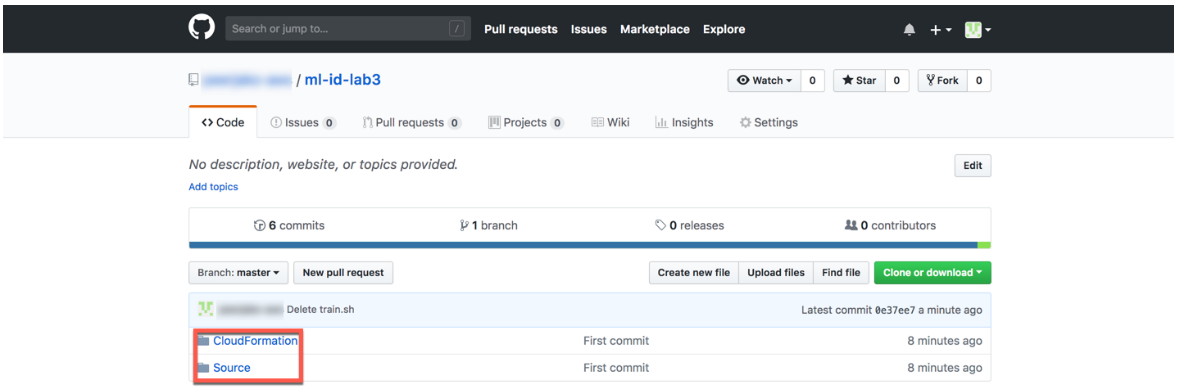
4. Stage your files running the following command lines:

```
git add .
git commit -am "First commit"
```

5. Set the remote origin and push the files by running the following command lines (use the GitHub repo URL from step 2):

```
git remote add origin https://github.com/<GITHUBUSER>/ml-id-lab3.git
git push -u origin master
```

Now validate that you have a CloudFormation directory and Source directory in your GitHub repo.



[redacted] / ml-id-lab3

No description, website, or topics provided.

Add topics

6 commits | 1 branch | 0 releases | 0 contributors

Branch: master

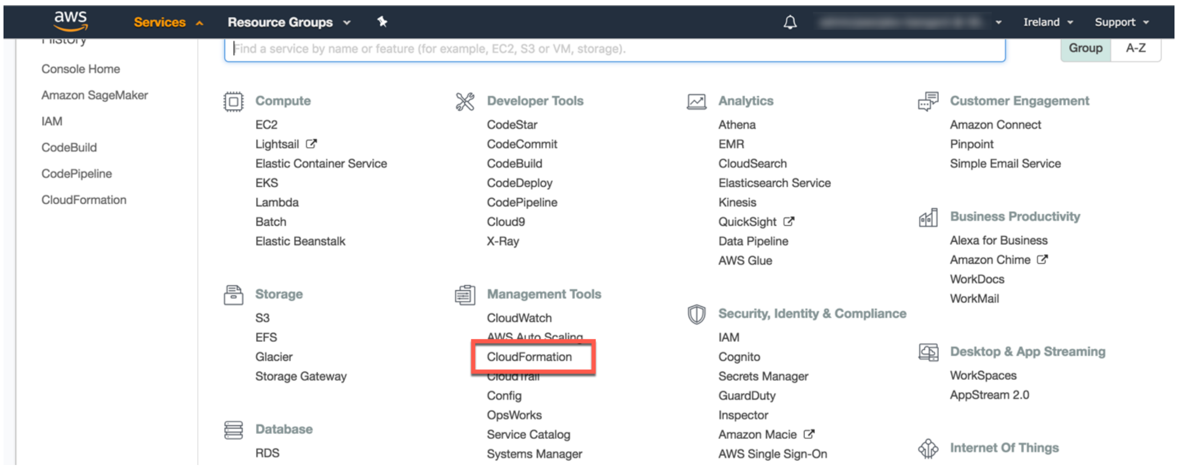
[redacted] Delete train.sh — Latest commit 0e37ee7 a minute ago

CloudFormation	First commit	8 minutes ago
Source	First commit	8 minutes ago

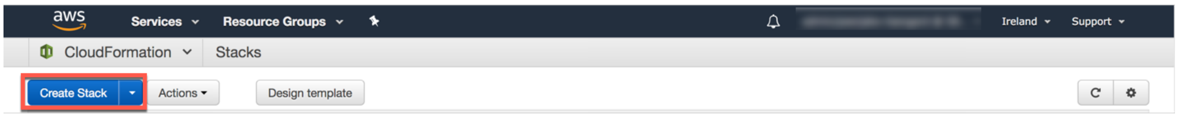
Create a CI/CD pipeline using AWS CodePipeline

Now that you have a GitHub repo with your AWS infrastructure code and your machine learning python code, you are ready to create a CI/CD pipeline.

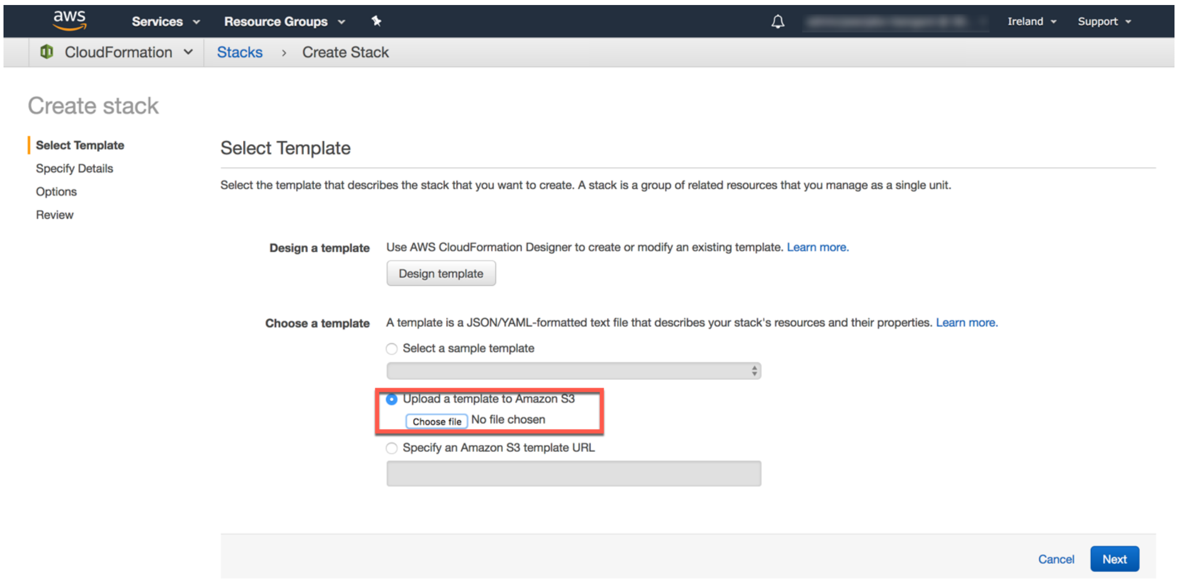
1. In case you have been logged out, sign into the AWS Management Console https://console.aws.amazon.com/, and confirm that you are working in the correct region.
2. Locate and click on **CloudFormation** from the list of all services. This will bring you to the Amazon Sagemaker dashboard. https://console.aws.amazon.com/cloudformation/home#/stacks?filter=active



3. Click on the **Create Stack** button.



4. You will create the stack using an existing CloudFormation template from your local disk located in the Lab3/CloudFormation directory of the Immersion Day material. Click on the **Choose file** button.



Select Template

Select the template that describes the stack that you want to create. A stack is a group of related resources that you manage as a single unit.

Design a template: Use AWS CloudFormation Designer to create or modify an existing template. Learn more.

Design template

Choose a template: A template is a JSON/YAML-formatted text file that describes your stack's resources and their properties. Learn more.

Select a sample template

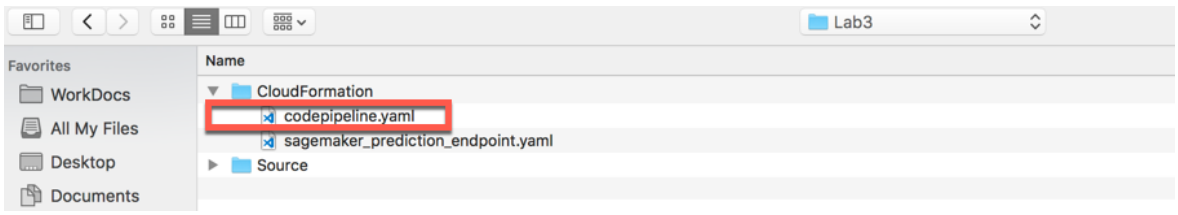
[✓ Upload a template to Amazon S3]

Choose file No file chosen

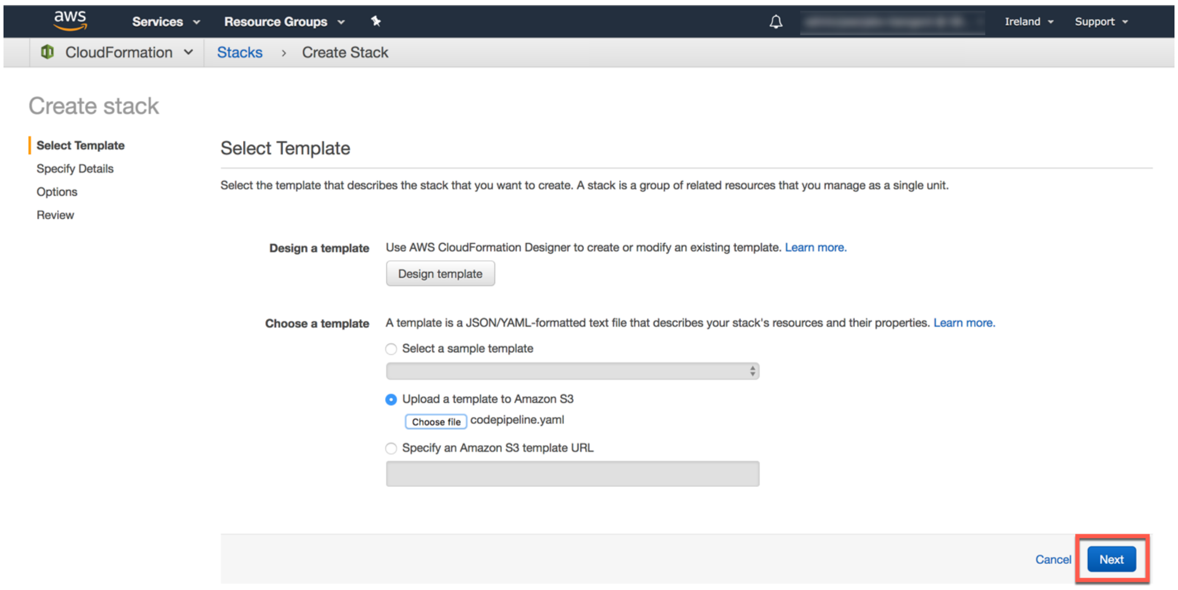
Specify an Amazon S3 template URL

Cancel Next

5. Navigate to the Lab3/CloudFormation directory and double click the **codepipeline.yaml** file.



6. Now click the **Next** button.



Create stack

Select Template

Select the template that describes the stack that you want to create. A stack is a group of related resources that you manage as a single unit.

Design a template Use AWS CloudFormation Designer to create or modify an existing template. Learn more.

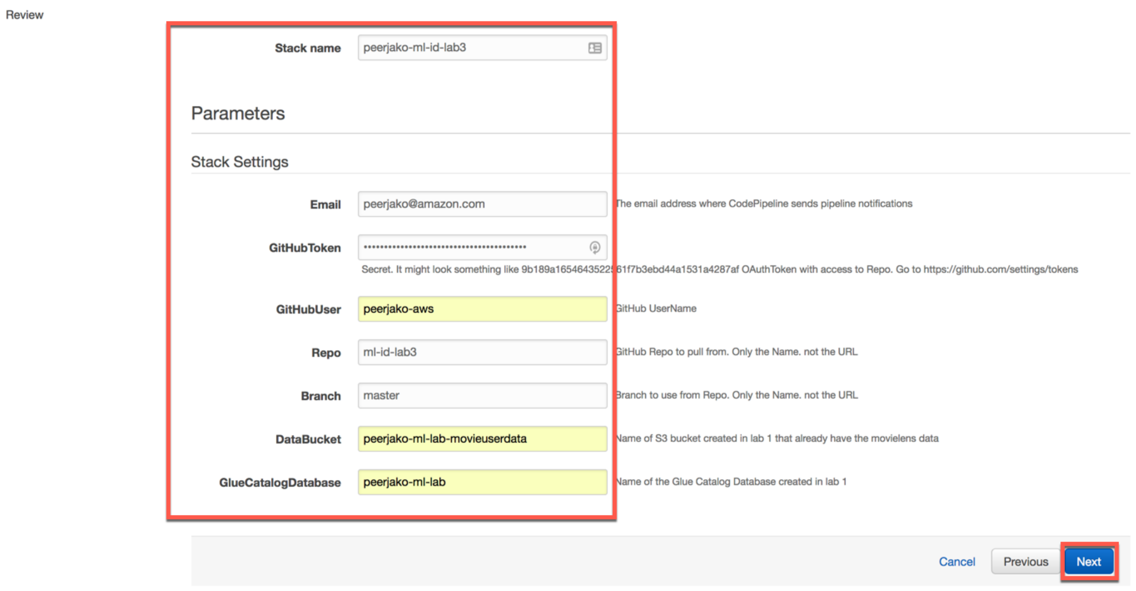
Choose a template A template is a JSON/YAML-formatted text file that describes your stack's resources and their properties. Learn more.

○ Select a sample template

● Upload a template to Amazon S3 — codepipeline.yaml

○ Specify an Amazon S3 template URL

7. You will now configure your stack. Type **[Your-Initials]-ml-lab3** into the **Stack name** text box (do not use a name longer than 15 characters. If curious why, ask your lab instructor.)



Stack name	peerjako-ml-id-lab3	

Parameters

Stack Settings

Email	peerjako@amazon.com	The email address where CodePipeline sends pipeline notifications
GitHubToken	••	Secret. It might look something like 9b189a1654643522561f7b3ebd44a1531a4287af OAuthToken with access to Repo. Go to https://github.com/settings/tokens
GitHubUser	peerjako-aws	GitHub UserName
Repo	ml-id-lab3	GitHub Repo to pull from. Only the Name. not the URL
Branch	master	Branch to use from Repo. Only the Name. not the URL
DataBucket	peerjako-ml-lab-movieuserdata	Name of S3 bucket created in lab 1 that already have the movielens data
GlueCatalogDatabase	peerjako-ml-lab	Name of the Glue Catalog Database created in lab 1

For the stack settings parameters configure them like this

Email

```
Enter your email address here. This is so you can receive CI/CD
notifications.
```

GitHubToken

```
This is the GitHub OAuth token you created in the beginning of this
lab. It might look something like this:
9b189a1654643522561f7b3ebd44a1531a4287af
```

GitHubUser

```
This is the user name of the GitHub account where you just create the
repo.
```

Repo

```
This is the name of the GitHub repo you just created. It is probably:
ml-id-lab3
```

Branch

```
Leave the git branch as: master
```

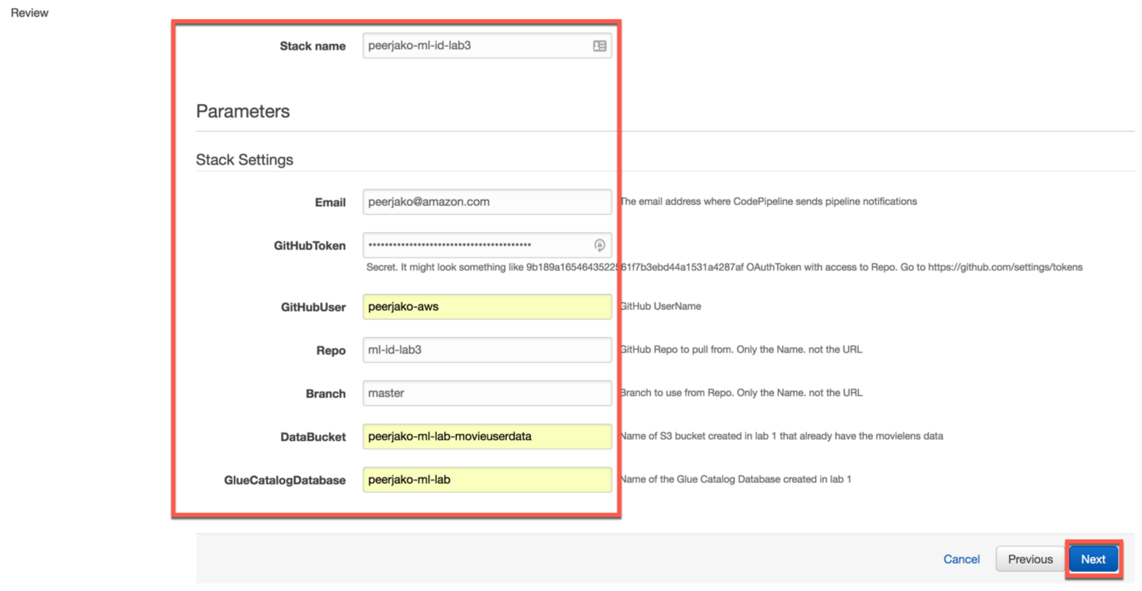
DataBucket

```
This is the name of the bucket you created in lab 1 and which you used
in lab 2. You probably named it something like this: [Your-Initials]-
ml-lab-movieuserdata
You can verify the bucket name here:
https://s3.console.aws.amazon.com/s3/home
```

GlueCatalogDatabase

```
This is the name of the Glue Catalog Database you created in lab1. You
probably named it something like this: [Your-Initials]-ml-lab
You can verify the database name here:
https://console.aws.amazon.com/glue/home#catalog:tab=databases
```

8. Now that you have typed in all the parameters click the **Next** button.


Review

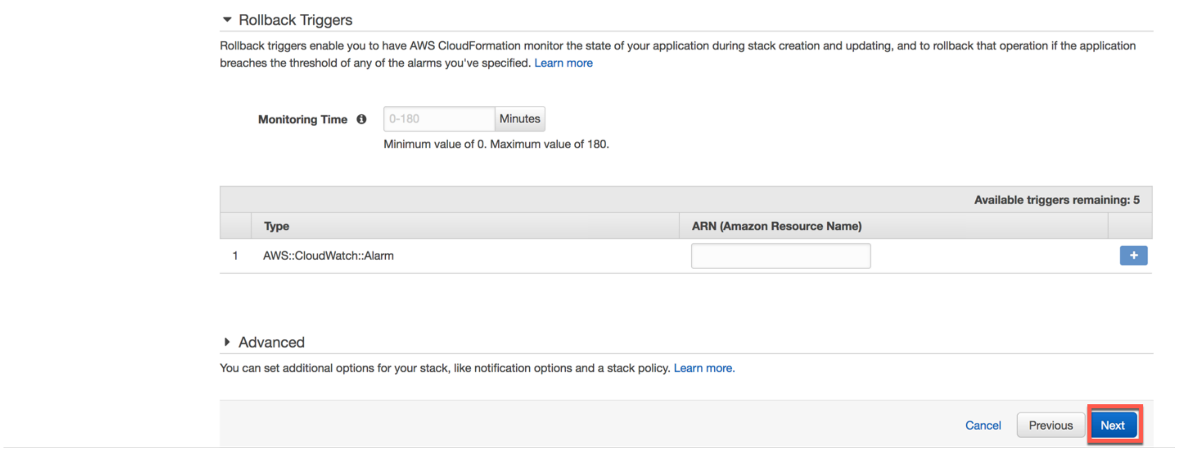
Stack name peerjako-ml-id-lab3

Parameters

Stack Settings

Email	peerjako@amazon.com	The email address where CodePipeline sends pipeline notifications
GitHubToken	••• Secret. It might look something like 9b189a1654643522[illegible]61f7b3ebd44a1531a4287af OAuthToken with access to Repo. Go to https://github.com/settings/tokens	
GitHubUser	peerjako-aws	GitHub UserName
Repo	ml-id-lab3	GitHub Repo to pull from. Only the Name. not the URL
Branch	master	Branch to use from Repo. Only the Name. not the URL
DataBucket	peerjako-ml-lab-movieuserdata	Name of S3 bucket created in lab 1 that already have the movielens data
GlueCatalogDatabase	peerjako-ml-lab	Name of the Glue Catalog Database created in lab 1

9. On this next page you leave all settings as default so scroll down to the bottom of the page and click the **Next** button.


Rollback Triggers

Rollback triggers enable you to have AWS CloudFormation monitor the state of your application during stack creation and updating, and to rollback that operation if the application breaches the threshold of any of the alarms you've specified. Learn more

Monitoring Time 0-180 Minutes

Minimum value of 0. Maximum value of 180.

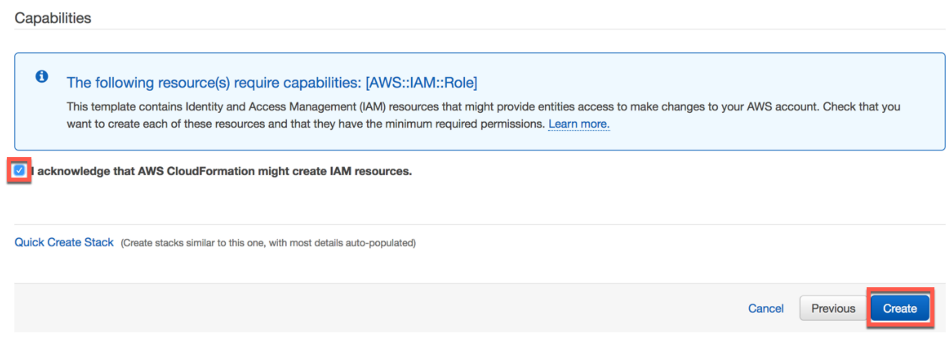
Available triggers remaining: 5

	Type	ARN (Amazon Resource Name)
1	AWS::CloudWatch::Alarm	

Advanced

You can set additional options for your stack, like notification options and a stack policy. Learn more.

10. On this next page scroll down to the bottom of the page and click the **I acknowledge that AWS CloudFormation might create IAM resources** check box. This give CloudFormation the right to create the IAM roles needed by the CodePipeline, CodeBuild and Sagemaker services. Now click the **Create** button.


Capabilities

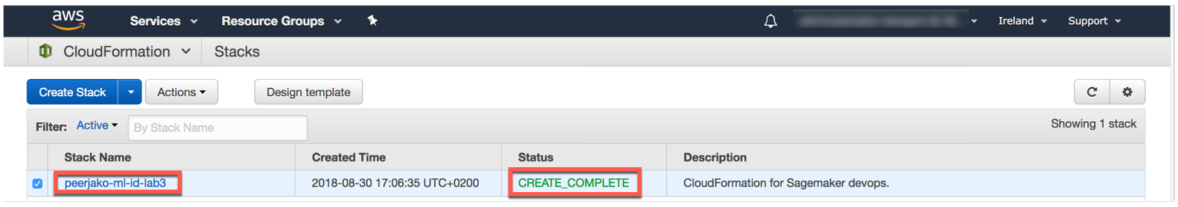
The following resource(s) require capabilities: [AWS::IAM::Role]

This template contains Identity and Access Management (IAM) resources that might provide entities access to make changes to your AWS account. Check that you want to create each of these resources and that they have the minimum required permissions. Learn more.

☑ I acknowledge that AWS CloudFormation might create IAM resources.

Quick Create Stack (Create stacks similar to this one, with most details auto-populated)

11. Now the CloudFormation Stack is being created. Wait for the **Status** to change from **CREATE_IN_PROGRESS** to **CREATE_COMPLETE**.

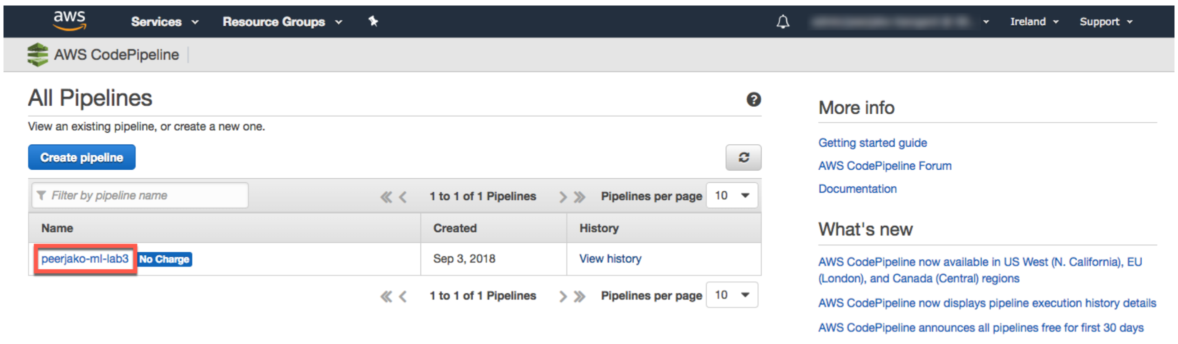


Filter: Active By Stack Name Showing 1 stack

	Stack Name	Created Time	Status	Description
☑	peerjako-ml-id-lab3	2018-08-30 17:06:35 UTC+0200	CREATE_COMPLETE	CloudFormation for Sagemaker devops.

12. The CloudFormation stack will create a CodePipeline that executes your Sagemaker CI/CD pipeline. Go to the CodePipeline console: https://console.aws.amazon.com/codepipeline/home#/dashboard

    Click on the name of the pipeline called something like this: **[Your-Initials]-ml-lab3**



All Pipelines

View an existing pipeline, or create a new one.

1 to 1 of 1 Pipelines Pipelines per page 10

Name	Created	History
peerjako-ml-lab3 No Charge	Sep 3, 2018	View history

1 to 1 of 1 Pipelines Pipelines per page 10

More info

Getting started guide
AWS CodePipeline Forum
Documentation

What's new

AWS CodePipeline now available in US West (N. California), EU (London), and Canada (Central) regions
AWS CodePipeline now displays pipeline execution history details
AWS CodePipeline announces all pipelines free for first 30 days

13. You should see a pipeline that consists of 4 different stages:

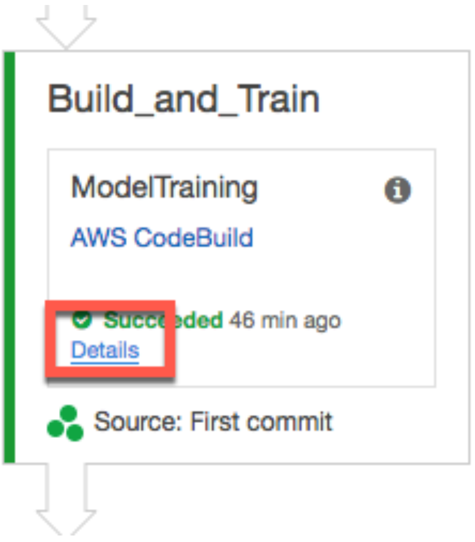
14. **Source** - this stage pulls the input source files from the GitHub repo you created earlier. The source files are zipped and uploaded to an S3 bucket. This source zip-file will then serve as input to the following pipeline stages. Every time you push new commits into your GitHub repo, the pipeline will execute again automatically pulling your latests commits. You can see the the S3 pipeline artifact bucket in the S3 management console: https://s3.console.aws.amazon.com/s3/home

    The bucket is called something like **[Your-Initials]-ml-lab3-[Your-Initials]-pipeline-artifact-store**

2. **Build_and_Train** - This stage runs a CodeBuild build using a python 3.4.5 build container. It runs the **prepdata.py** and **train.py** python scripts that you can find in the **Source** directory of your GitHub repo.
    - prepdata.py - This python script is very similar to the first part of the python notebook you ran manually in lab 2. It ends up creating a training data file and test data file that can be used by the Sagemaker training. It uses Athena to dynamically get some of the parameters that were hardcoded in the lab 2 notebook.
    - train.py - This python script is similar to the training part of the python notebook you ran manually in lab 2. It creates a training job in Sagemaker which ends up producing an ML model that you can you can use for ML inference through Sagemaker endpoints.

    After the two scripts have executed, CodeBuild packages and uploads the build artifacts to an S3 bucket. In this case the artifacts include the test data file and some config files used in the next pipeline stages.

    If you click the **Details** link of the **Build_and_Train** stage then you will go to the CodeBuild details page. In the bottom of CodeBuild details page you can see the build logs. You will find these logs very similar to the logs you saw when running the python notebook in lab 2.

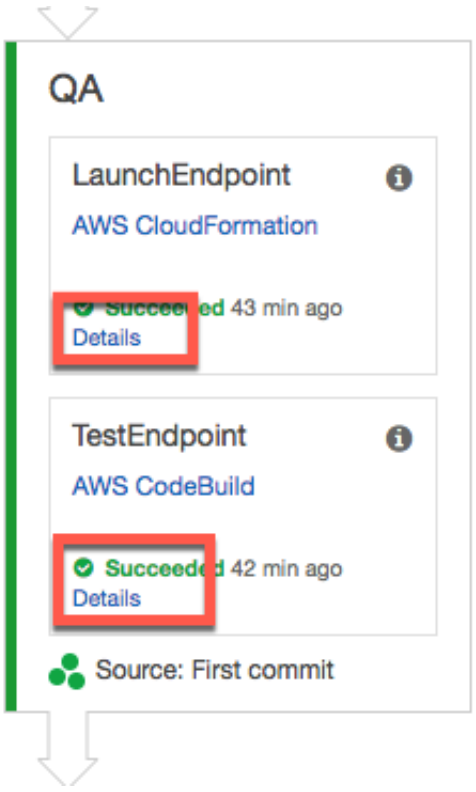


3. **QA** - this stage uses the CloudFormation template file **sagemaker_prediction_endpoint.yaml** found in the **CloudFormation** directory. CloudFormation is used to launch a Sagemaker QA Endpoint based on the ML model created in the Build_and_Train stage.

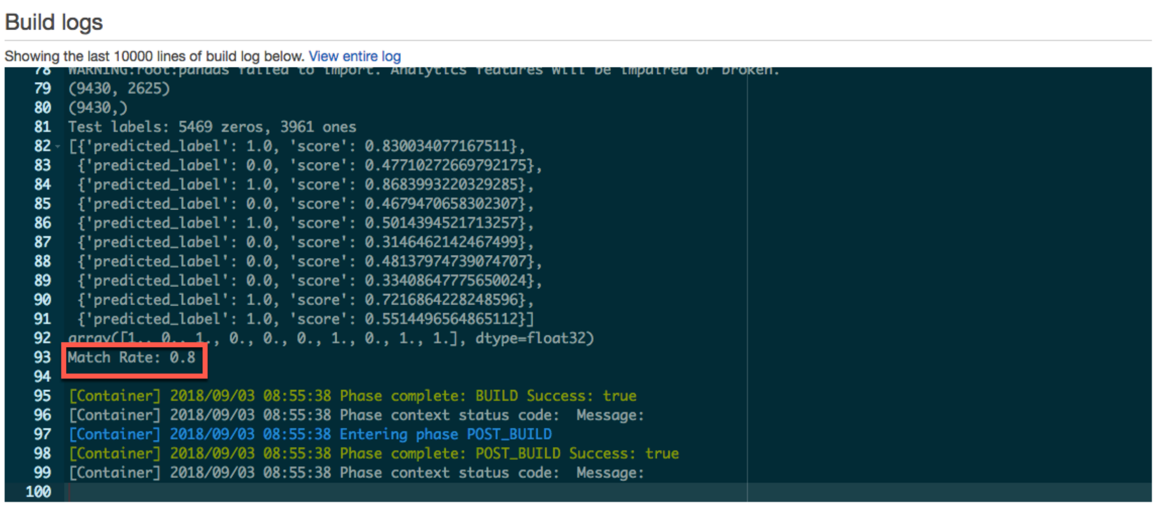
   When the Sagemaker QA Endpoint has launched (this can take up to 5 minutes) the Endpoint is tested by running a CodeBuild build that executes the **test.py** python script foind in the **Source** directory. This python script is very similar to the last part of the python notebook you ran manually in lab 2. It will call the Sagemaker endpoint and compare the inference results with the results from the test data file. If less than 80% of the inferences are not producing the same result as found in the test data file, then an exception is thrown and the pipeline stage will fail.

   If you click the **Details** link of the **LaunchEnpoint** step then you will go to details of the CloudFormation stack of the Sagemaker Endpoint.

   If you click the **Details** link of the TestEndpoint step then you will go to the CodeBuild build details.



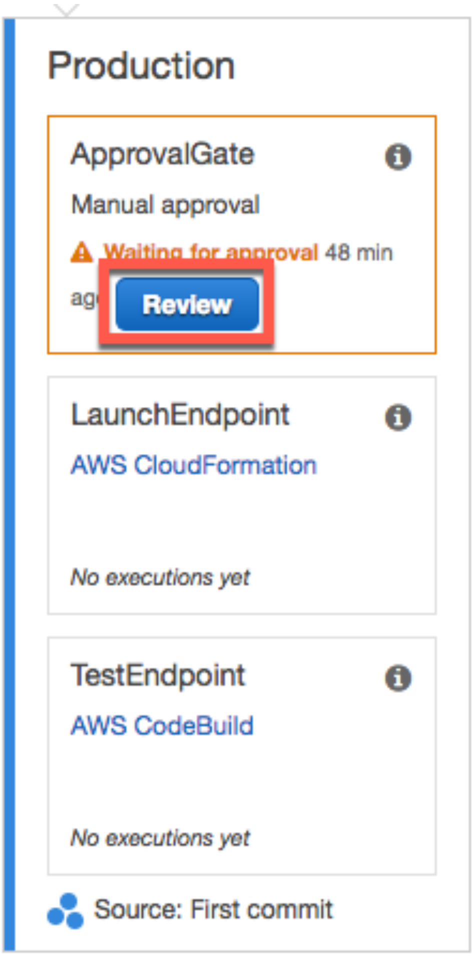
   Click the **Details** link of the TestEndpoint and scroll all the way down on that page to see the build logs. In the build logs you will see the result of the tests including how many tests was correctly predicted. With a **Match Rate** of 80% you can see that the test barely passes our 80% threshold. In the Bonus part of this lab we will improve this match rate.


Build logs
Showing the last 10000 lines of build log below. View entire log

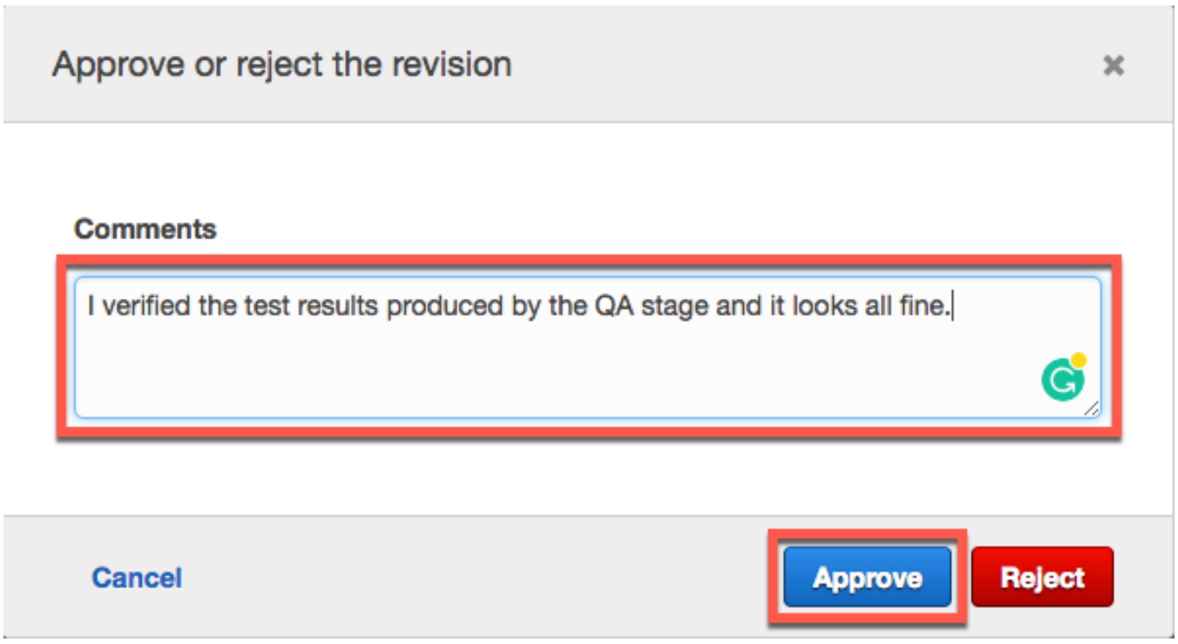
```
WARNING:root:pandas failed to import. Analytics features will be impaired or broken.
(9430, 2625)
(9430,)
Test labels: 5469 zeros, 3961 ones
[{'predicted_label': 1.0, 'score': 0.830034077167511},
 {'predicted_label': 0.0, 'score': 0.47710272669792175},
 {'predicted_label': 1.0, 'score': 0.8683993220329285},
 {'predicted_label': 0.0, 'score': 0.4679470658302307},
 {'predicted_label': 1.0, 'score': 0.5014394521713257},
 {'predicted_label': 0.0, 'score': 0.3146462142467499},
 {'predicted_label': 0.0, 'score': 0.48137974739074707},
 {'predicted_label': 0.0, 'score': 0.33408647775650024},
 {'predicted_label': 1.0, 'score': 0.7216864228248596},
 {'predicted_label': 1.0, 'score': 0.5514496564865112}]
array([1., 0., 1., 0., 0., 0., 1., 0., 1., 1.], dtype=float32)
Match Rate: 0.8

[Container] 2018/09/03 08:55:38 Phase complete: BUILD Success: true
[Container] 2018/09/03 08:55:38 Phase context status code:  Message:
[Container] 2018/09/03 08:55:38 Entering phase POST_BUILD
[Container] 2018/09/03 08:55:38 Phase complete: POST_BUILD Success: true
[Container] 2018/09/03 08:55:38 Phase context status code:  Message:
```

4. **Production** - this stage first starts with an approval gate. In order to continue the pipeline you have to manually approve. Click the **Review** button.



A pop-up is opened. Type in a comment in the **Comments** text box and click the **Approve** button.



I verified the test results produced by the QA stage and it looks all fine.

The rest of the production stage creates a Sagemaker production Endpoint and runs an Endpoint test similar to the QA stage. The creation of the production endpoint should take approximately 5 minutes.

Congratulations! You have now successfully created a fully functional CI/CD pipeline for your machine learning project.

If you update the pythons scripts and push the changes into your GitHub repo you will automatically trigger a redeployment of your ML endpoints based on a new version of the ML model. Using a CI/CD pipeline with a QA stage and a manual approval step, you can safely develop and incrementally improve the quality of your ML inference endpoints while using best practices from a DevOps perspective.

In the Sagemaker console you can view the training job, the ML model and the QA and Production endpoints created by the CI/CD pipeline:

- Sagemaker Jobs: https://console.aws.amazon.com/sagemaker/home#/jobs
- Sagemaker Models: https://console.aws.amazon.com/sagemaker/home#/models
- Sagemaker Endpoints: https://console.aws.amazon.com/sagemaker/home#/endpoints

You might already be using a CI/CD solution in your job such as Jenkins, CircleCI or TeamCity. Most of the parts of this lab 3 are reusable and do not depend on AWS CodePipeline. You can use the ideas behind the pipeline and the different parts as an inspiration to create a CI/CD pipeline in the CI/CD solution you are already using at your job.

Bonus Exercise: Add Sagemaker Hyperparameter tuning to your Sagemaker training

Automatic Model Tuning eliminates the undifferentiated heavy lifting required to search the hyperparameter space for more accurate models. With Amazon SageMaker Automatic Model Tuning you save significant time and effort in training and tuning your machine learning models.

A Hyperparameter Tuning job launches multiple training jobs, with different hyperparameter combinations, based on the results of completed training jobs. SageMaker trains a "meta" machine learning model, based on Bayesian Optimization, to infer hyperparameter combinations for our training jobs.

You will now change your existing python training code to use Hyperparameter Tuning.

1. Open the file Lab3/Source/train.py with your favorite text editor.
2. Go to line 136 and change the variabel **no_hyper_parameter_tuning** to **False** and save the changes

   ```
   no_hyper_parameter_tuning = False
   ```

3. Now you will commit and push your change to github. In your terminal run the following

   ```
   git commit -am "Do Hyperparameter Tuning"
   git push -u origin master
   ```

4. CodePipeline will automatically detect that a new commit has been pushed to your github repo and start a new pipeline execution.

   You can go and check this here:

   https://console.aws.amazon.com/codepipeline/home#/dashboard

   When the Build_and_Train stage of the pipeline has started running you can see a Hyperparameter tuning job here:

   https://console.aws.amazon.com/sagemaker/home#/hyper-tuning-jobs

   You can see how that job has created multiple training jobs here:

   https://console.aws.amazon.com/sagemaker/home#/jobs

5. When all the training jobs has finished the code will get that model from the best performing training job and the pipeline will use that model for QA and Production. In the QA test logs you can verify that the inference quality has improved from an 80% match rate to a 90% match rate.

The code for creating a Hyperparameter Tuner looks like this

```
    my_tuner = HyperparameterTuner(
        estimator=fm,
        objective_metric_name='test:binary_classification_accuracy',
        hyperparameter_ranges={
            'epochs': IntegerParameter(1, 200),
            'mini_batch_size': IntegerParameter(10, 10000),
            'factors_wd': ContinuousParameter(1e-8, 512)},
        max_jobs=4,
        max_parallel_jobs=4)
```

Here you can see that **epochs**, **mini_batch_size** and **factors_wd** are the parameters that are being tuned. In a real world scenario you would need more than just 4 jobs (**max_jobs**) to find the optimal model.

You can read a blog post that goes into the details of Hyperparameter Tuning here:

https://aws.amazon.com/blogs/aws/sagemaker-automatic-model-tuning/