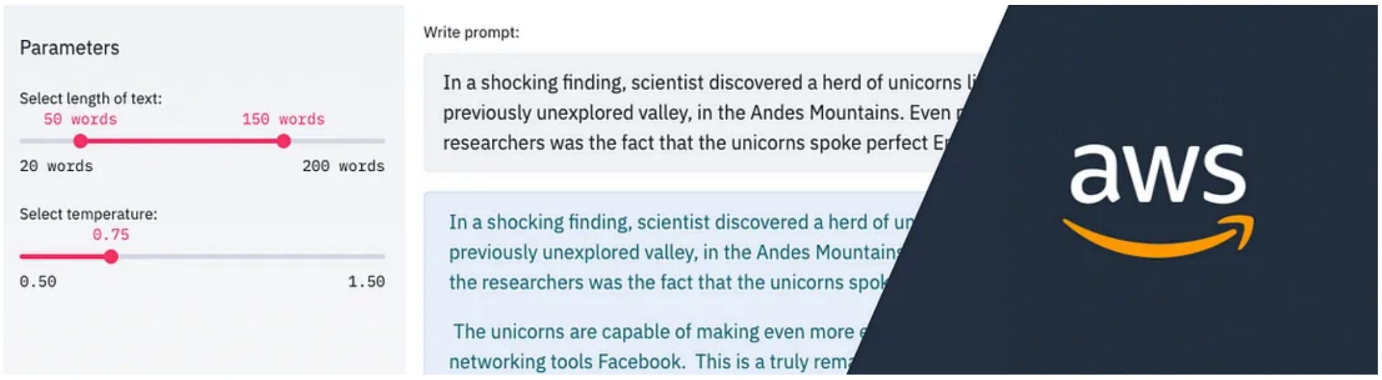
Streamlit on AWS: a fully-featured solution for Streamlit deployments.

Sample AWS CloudFormation template & code included.

Thom Lane · Follow
6 min read · Jun 29, 2020



You've trained a useful machine learning model, and you've created an intuitive Streamlit application to match. Only issue is… they're currently stuck on your development machine. So what should you do if you need to share your Streamlit application with the rest of the world or within your company? We'll walk through a comprehensive answer to that question in this post, dive into important details that might otherwise be overlooked and share all the code that's required to deploy your own dashboards.

🐙 : Click here to see code on GitHub

🚀 : Click here to launch AWS CloudFormation stack

Minimal Approach

Using Amazon EC2

One of the simplest sounding approaches to Streamlit deployment on AWS would be to use Amazon EC2. You can deploy your Streamlit application on an Amazon EC2 instance (i.e. a virtual server in the AWS Cloud) and let application users connect to that instance. You can use an AWS Deep Learning AMI on a GPU instance if you need to use a deep learning model for your Streamlit application. Although this is architecturally simple, there are a number of factors that can make things more complex.

Who can access the application and how can access be limited to certain individuals? Are sensitive communications being encrypted with HTTPS? What happens if the server crashes? Who's going to install bug fixes and security updates on the instance? What happens if the number of users goes up and down significantly over time? Will the instance handle peak traffic? What about updates to the model and application? And this list goes on.

So although this approach might work in some cases, there are a number of other AWS services that can help us for a fully-featured deployment.

Comprehensive Approach

We have 3 central AWS services in this approach: Amazon SageMaker, Amazon ECS and Amazon Cognito. Amazon SageMaker is designed for seamless model training and deployment, and it works great for Streamlit development too. Amazon ECR and ECS are a perfect compliment for containerised deployments. And Amazon Cognito specialises in simple and secure authentication. Combining these AWS services, we end up with the following architecture in Figure 1. We'll then dive straight into the details.

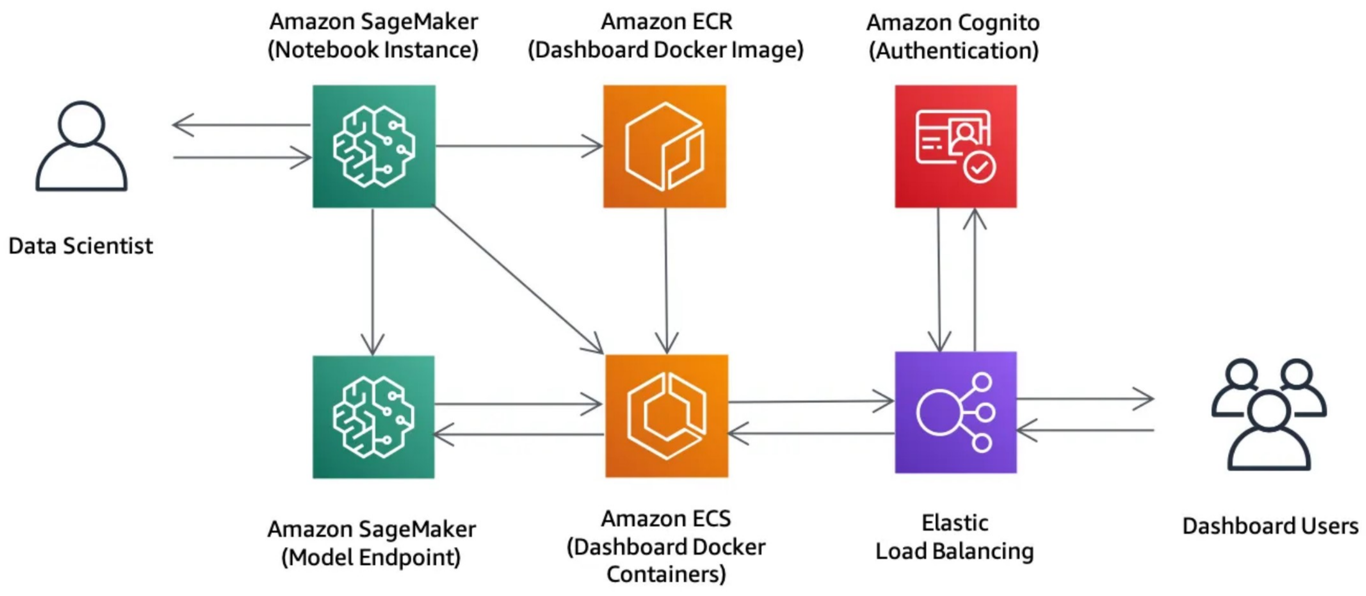

Amazon SageMaker (Notebook Instance)
Amazon ECR (Dashboard Docker Image)
Amazon Cognito (Authentication)
Data Scientist
Amazon SageMaker (Model Endpoint)
Amazon ECS (Dashboard Docker Containers)
Elastic Load Balancing
Dashboard Users

Figure 1: Architecture of AWS components used. Some of which are optional.

> *An AWS CloudFormation template is shared so you can create all of the required resources inside your own AWS account with just a few clicks. You can choose to deploy one of two example Streamlit applications: 'Uber Pickups in New York City' (a self-contained example) and 'DistilGPT-2 Text Generation' (an example that interacts with a machine learning model). All code is customisable.*

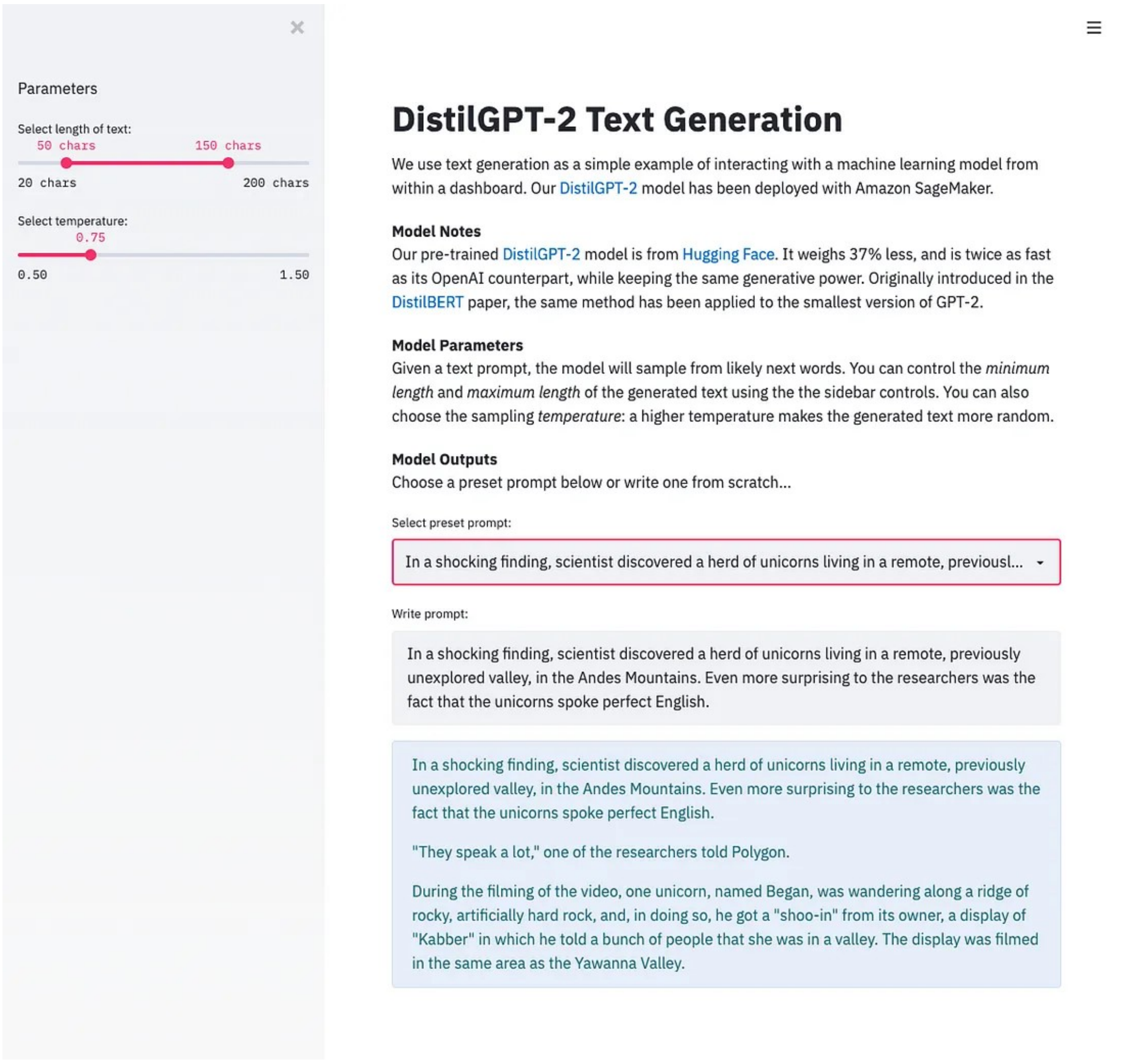
Using Amazon SageMaker

When it comes to developing and deploying machine learning models, Amazon SageMaker simplifies the manual experience of Amazon EC2. Within minutes you can jump into a Jupyter notebook and start deploying models on dedicated and fully managed infrastructure. Out of the box, you have access to a number of pre-build Conda environments and Docker containers. In the 'DistilGPT-2 Text Generation' example, the pre-built PyTorch Docker container is used to deploy the DistilGPT-2 model from transformers on a ml.c5.xlarge instance. Amazon SageMaker then provides a simple HTTPS endpoint to interact with the deployed model. Our example application uses the `invoke_endpoint` method (from `boto3`) to call the text generation model.

```
import boto3
import json

data = {
    'text': 'In a shocking finding',
    'parameters': {
        'min_length': 100,
        'max_length': 200
    }
}

sagemaker_client = boto3.client('sagemaker-runtime')

response = sagemaker_client.invoke_endpoint(
    EndpointName='text-generation',
    ContentType="application/json",
    Accept="application/json",
    Body=json.dumps(data)
)

body_str = response['Body'].read().decode("utf-8")
body = json.loads(body_str)

print(body['text'])
# In a shocking finding, scientist discovers a herd of unicorns...
```

Amazon SageMaker can be used for Streamlit development too. After deploying the model, the Streamlit application can be built and tested directly on the notebook instance. A containerized approach is taken for simplified application deployment, but you still get all the benefits of live-reload when editing files (thanks to a local volume mount). When running the Streamlit container on the Amazon SageMaker Notebook Instance, you can access it via the jupyter-server-proxy at the following authenticated URL:

```
https://{NOTEBOOK_URL}/proxy/8501/
```


Parameters

Select length of text:
50 chars 150 chars
20 chars 200 chars

Select temperature:
0.75
0.50 1.50

DistilGPT-2 Text Generation

We use text generation as a simple example of interacting with a machine learning model from within a dashboard. Our DistilGPT-2 model has been deployed with Amazon SageMaker.

Model Notes
Our pre-trained DistilGPT-2 model is from Hugging Face. It weighs 37% less, and is twice as fast as its OpenAI counterpart, while keeping the same generative power. Originally introduced in the DistilBERT paper, the same method has been applied to the smallest version of GPT-2.

Model Parameters
Given a text prompt, the model will sample from likely next words. You can control the *minimum length* and *maximum length* of the generated text using the the sidebar controls. You can also choose the sampling *temperature*: a higher temperature makes the generated text more random.

Model Outputs
Choose a preset prompt below or write one from scratch...

Select preset prompt:
In a shocking finding, scientist discovered a herd of unicorns living in a remote, previousl...

Write prompt:
In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.

In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.

"They speak a lot," one of the researchers told Polygon.

During the filming of the video, one unicorn, named Began, was wandering along a ridge of rocky, artificially hard rock, and, in doing so, he got a "shoo-in" from its owner, a display of "Kabber" in which he told a bunch of people that she was in a valley. The display was filmed in the same area as the Yawanna Valley.

Figure 2: an example Streamlit application deployed with this solution.

When you're finished with application development on Amazon SageMaker, you can push your container to Amazon Elastic Container Registry (ECR). Similar to Docker Hub, it provides a repository for your Docker images, but it keeps the images within your AWS account for extra security and reliability.

```
docker tag {IMAGE_NAME} {STREAMLIT_ECR_REPOSITORY_URL}:latest
docker push {STREAMLIT_ECR_REPOSITORY_URL}:latest
```

Using Amazon ECS

Your Streamlit Docker image is now on Amazon ECR, but the application isn't actually running yet. Amazon Elastic Container Service (ECS) is a fully-managed service for running Docker containers. You don't need to provision or manage servers, you just define the task that needs to be run and specify the resources the task needs. Our example task definition states that the Streamlit Docker container should be run with a single vCPU and 2GB of memory. Our example service runs and maintains a specified number of instances of the task definition simultaneously. So for increased availability, you can set the desired task count of your service to 2 using the AWS CLI:

```
aws ecs update-service \
  --cluster {STREAMLIT_ECS_CLUSTER} \
  --service {STREAMLIT_ECR_SERVICE} \
  --desired-count 2
```

One of the main advantages to using an Amazon ECS service is that it constantly monitors the health of the tasks and replaces tasks that have failed or stopped for any reason. Amazon ECS services can also auto-scale the number of tasks (i.e. automatically increase or decrease) to deal with high demand at peak times and reduce cost during periods of low utilization. Our example solution also includes an Application Load Balancer which distributes traffic across tasks, integrates with Amazon Certificate Manager (for HTTPS) and authenticates traffic with Amazon Cognito.

Using Amazon Cognito

When the contents of your Streamlit application are private, Amazon Cognito can be used to restrict access to a certain set of users. Although this component is optional, it is enabled by default on the solution. You can integrate with social and enterprise identify providers (such as Google and Microsoft Active Directory) but the solution creates its own user pool with application specific accounts. You just need to provide an email address during stack creation to receive the temporary login credentials.

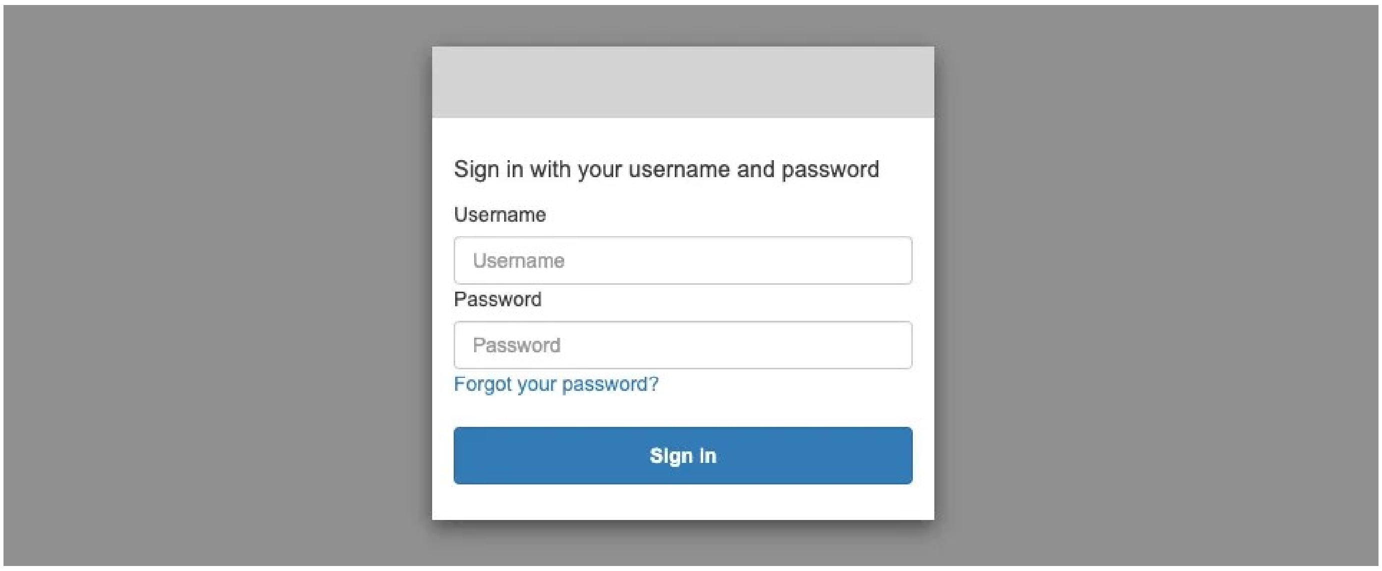


Figure 3: Amazon Cognito Sign-in

When Amazon Cognito authentication is enabled, you will see the managed sign-in page when trying to access the application for the first time. Use the temporary login credentials, and then enter a new password for the account. When successfully logged in, you'll be able to see your Streamlit application.

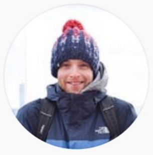
Summary

Sometimes Streamlit applications can be showcased directly from your development machine. Other times they can be manually deployed on Amazon EC2 instances. But for the times when you need to share you Streamlit applications with the rest of the world or within your company, a more robust and secure approach might be required. Using the AWS CloudFormation template included in this solution, you can deploy your Streamlit application on AWS in a scalable and secure way within minutes.

> 🐙 : Click here to see code on GitHub

> 🚀 : Click here to launch AWS CloudFormation stack

Streamlit · Dashboard · AWS · Machine Learning · Data Science



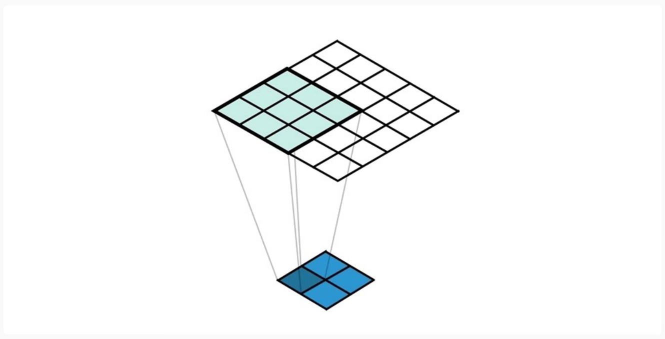
Written by Thom Lane

Follow

536 Followers

Machine Learning Scientist for AWS AI

More from Thom Lane



Thom Lane in Apache MXNet

Transposed Convolutions explained with… MS Excel!

You've successfully navigated your way around 1D Convolutions, 2D Convolutions an…

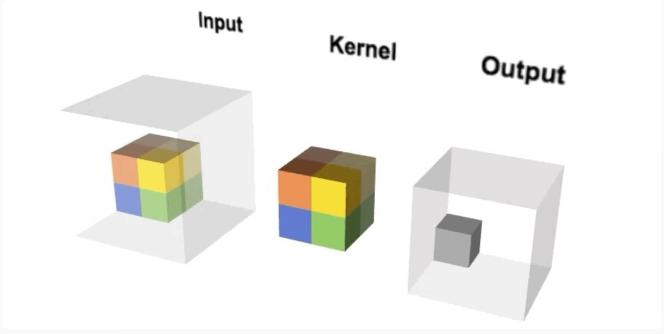
Nov 2, 2018 1.8K 13

Thom Lane in Apache MXNet

Multi-Channel Convolutions explained with… MS Excel!

We've looked at 1D Convolutions, 2D Convolutions and 3D Convolutions in previo…

Oct 18, 2018 918 4



Thom Lane in Apache MXNet

Convolutions explained with… MS Excel!

Convolutions are ubiquitous in deep learning. You can find them in the vast majority of…

Oct 16, 2018 585 3

Thom Lane in Apache MXNet

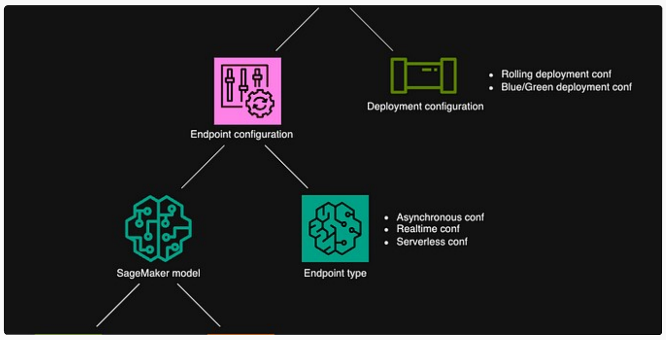
1D & 3D Convolutions explained with… MS Excel!

Welcome back to the blog post series on convolutions using MS Excel and Apache…

Oct 17, 2018 496 1

See all from Thom Lane

Recommended from Medium



Niklas Palm

SageMaker endpoints — custom Docker images

This is part of an opinionated series on how to master SageMaker for machine learning. For…

May 16 5

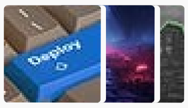
Manyi

Hosting Streamlit App on AWS EC2

If you have created a Streamlit app and like to deploy it on AWS, for your own use or for…

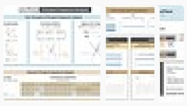
Aug 3 6

Lists



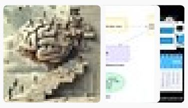
Predictive Modeling w/ Python

20 stories · 1621 saves



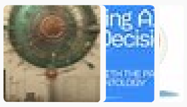
Practical Guides to Machine Learning

10 stories · 1978 saves



Natural Language Processing

1780 stories · 1384 saves



data science and AI

40 stories · 271 saves

Syntax Erreur

MLOps: Step-by-Step Tutorial to Model Deployment with Flask and…

Machine Learning Operations (MLOps) is an essential practice that combines Machine…

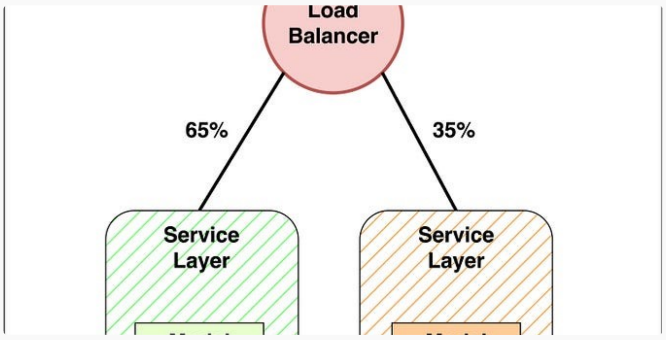
May 31 123

Shaw Talebi in Towards Data Science

How to Deploy ML Solutions with FastAPI, Docker, and GCP

A hands-on guide with Python example code

Jun 5 515 1



The MLOps Guy

Mastering Advanced ML Model Deployment Techniques

Are you grappling with which machine learning model deployment strategy to...

✦ Sep 20 30

Benjamin Franklin. S

Streamlit with Nginx: A Step-by-Step Guide to Setting up Your Dat...

Streamlit is an open-source app framework that is perfect for machine learning and data...

Jun 29 1

See more recommendations