Bust Fake News with ML & AWS

Ananth Iyer

Principal Machine Learning Engineer

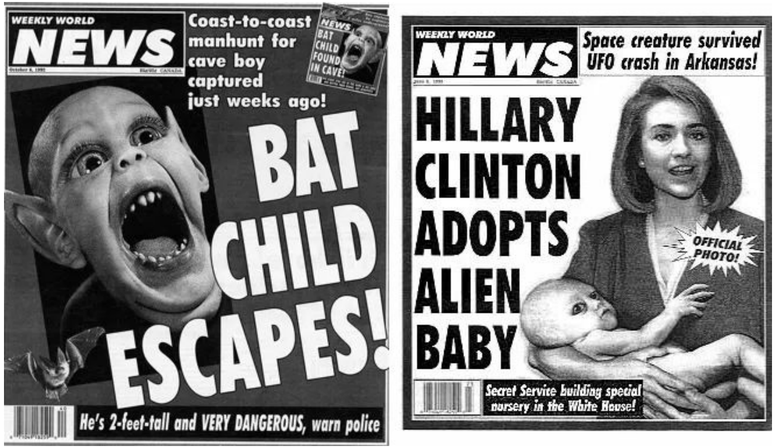
LIFEOMIC

what are we doing today?

explore pre-trained model

classifier training/inference

package and deploy



WEEKLY WORLD

NEWS

Coast-to-coast manhunt for cave boy captured just weeks ago!

NEWS

BAT CHILD FOUND IN CAVE!

BAT CHILD ESCAPES!

He's 2-feet-tall and VERY DANGEROUS, warn police

WEEKLY WORLD

NEWS

Space creature survived UFO crash in Arkansas!

HILLARY CLINTON ADOPTS ALIEN BABY

OFFICIAL PHOTO!

Secret Service building special nursery in the White House!

what is needed

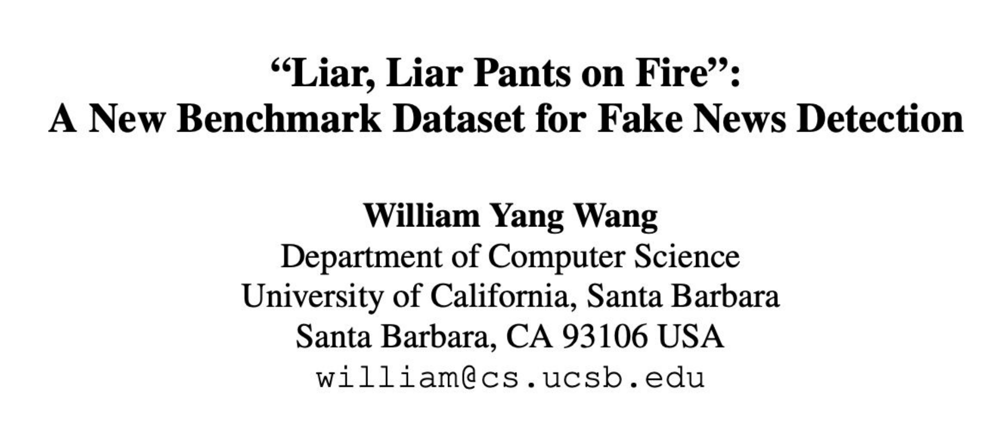
- dataset
- pretrained model
- compute for training
- hosted inference

I picked

- PolitiFact
- BERT
- p2.xlarge
- fargate

PolitiFact

https://www.politifact.com/



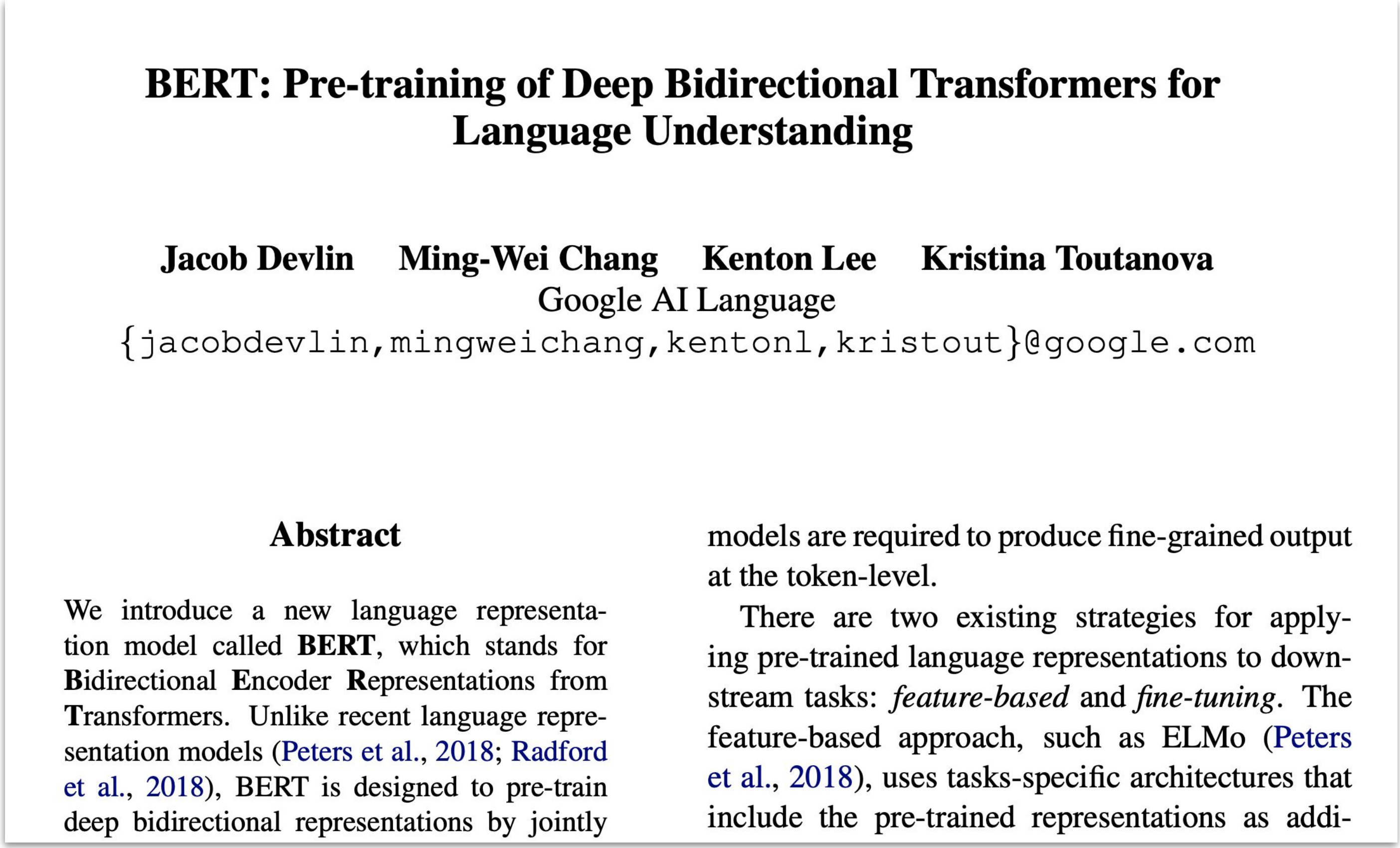
“Liar, Liar Pants on Fire”: A New Benchmark Dataset for Fake News Detection

William Yang Wang
Department of Computer Science
University of California, Santa Barbara
Santa Barbara, CA 93106 USA
`william@cs.ucsb.edu`

https://arxiv.org/abs/1705.00648

BERT

https://www.ripleys.com/weird-news/what-if-bert-was-real/



BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Jacob Devlin Ming-Wei Chang Kenton Lee Kristina Toutanova
Google AI Language
{jacobdevlin,mingweichang,kentonl,kristout}@google.com

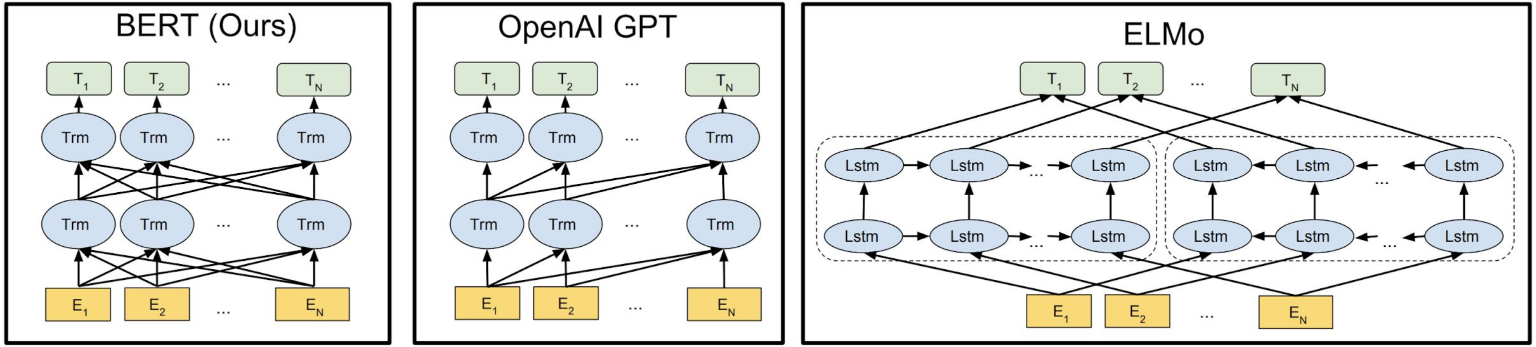
Abstract

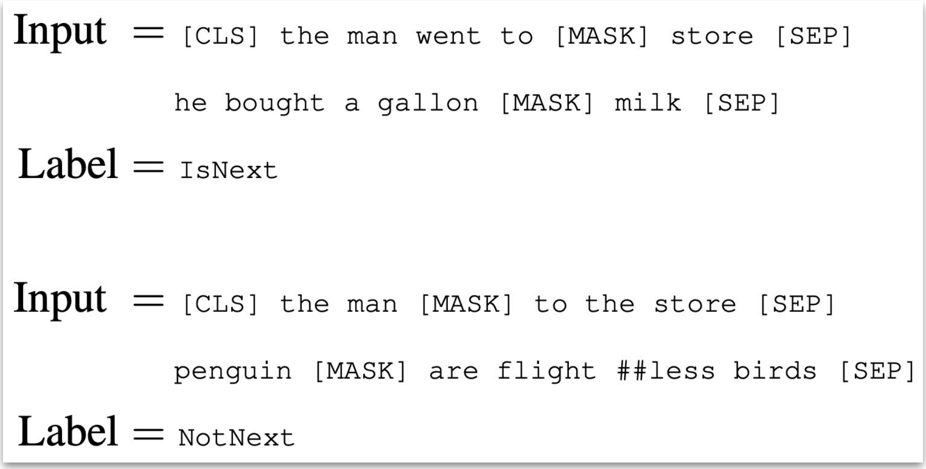
We introduce a new language representation model called **BERT**, which stands for **B**idirectional **E**ncoder **R**epresentations from **T**ransformers. Unlike recent language representation models (Peters et al., 2018; Radford et al., 2018), BERT is designed to pre-train deep bidirectional representations by jointly

models are required to produce fine-grained output at the token-level.

There are two existing strategies for applying pre-trained language representations to downstream tasks: *feature-based* and *fine-tuning*. The feature-based approach, such as ELMo (Peters et al., 2018), uses tasks-specific architectures that include the pre-trained representations as addi-

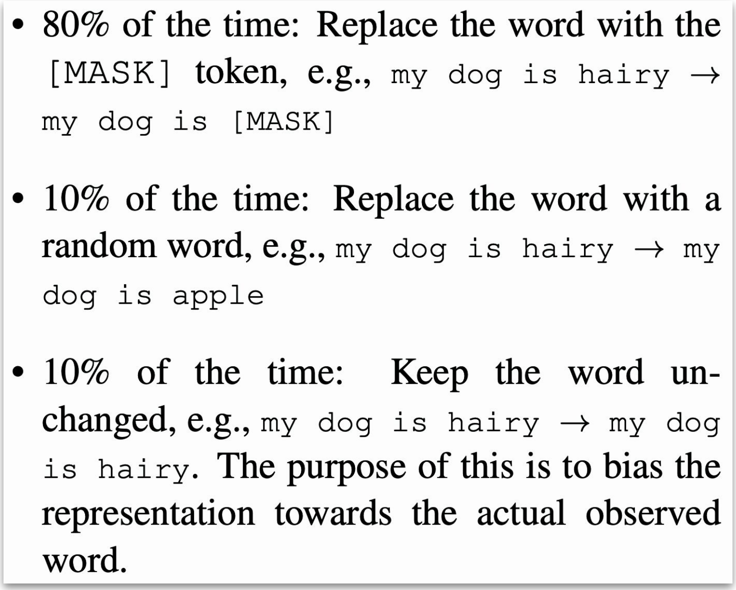
BERT: method to model language



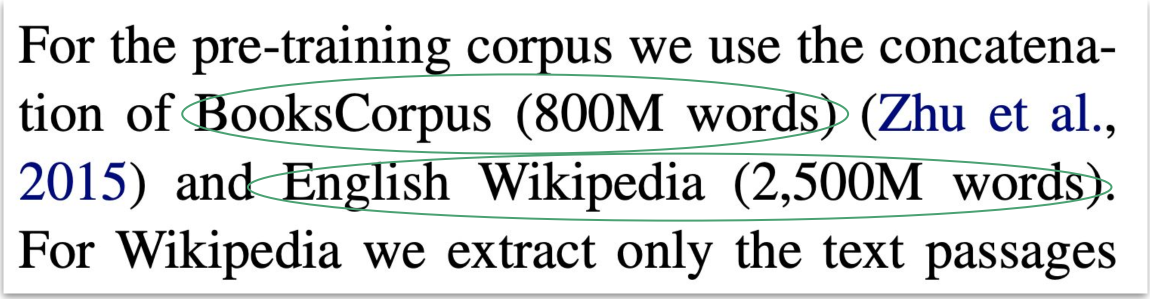


Input = [CLS] the man went to [MASK] store [SEP]
he bought a gallon [MASK] milk [SEP]
Label = IsNext

Input = [CLS] the man [MASK] to the store [SEP]
penguin [MASK] are flight ##less birds [SEP]
Label = NotNext

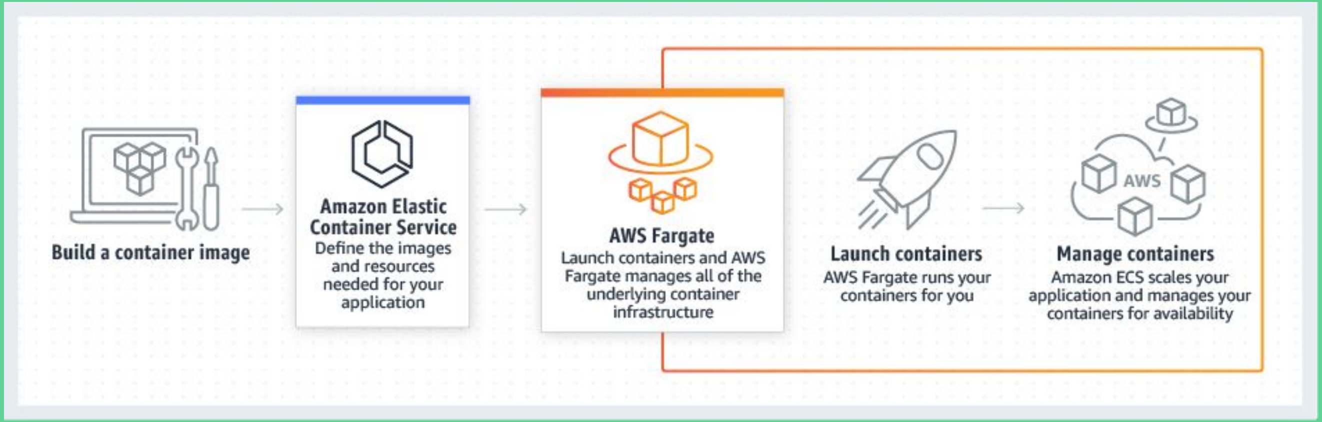


- 80% of the time: Replace the word with the [MASK] token, e.g., `my dog is hairy` → `my dog is [MASK]`
- 10% of the time: Replace the word with a random word, e.g., `my dog is hairy` → `my dog is apple`
- 10% of the time: Keep the word unchanged, e.g., `my dog is hairy` → `my dog is hairy`. The purpose of this is to bias the representation towards the actual observed word.



For the pre-training corpus we use the concatenation of BooksCorpus (800M words) (Zhu et al., 2015) and English Wikipedia (2,500M words). For Wikipedia we extract only the text passages

fargate



Build a container image

Amazon Elastic Container Service
Define the images and resources needed for your application

AWS Fargate
Launch containers and AWS Fargate manages all of the underlying container infrastructure

Launch containers
AWS Fargate runs your containers for you

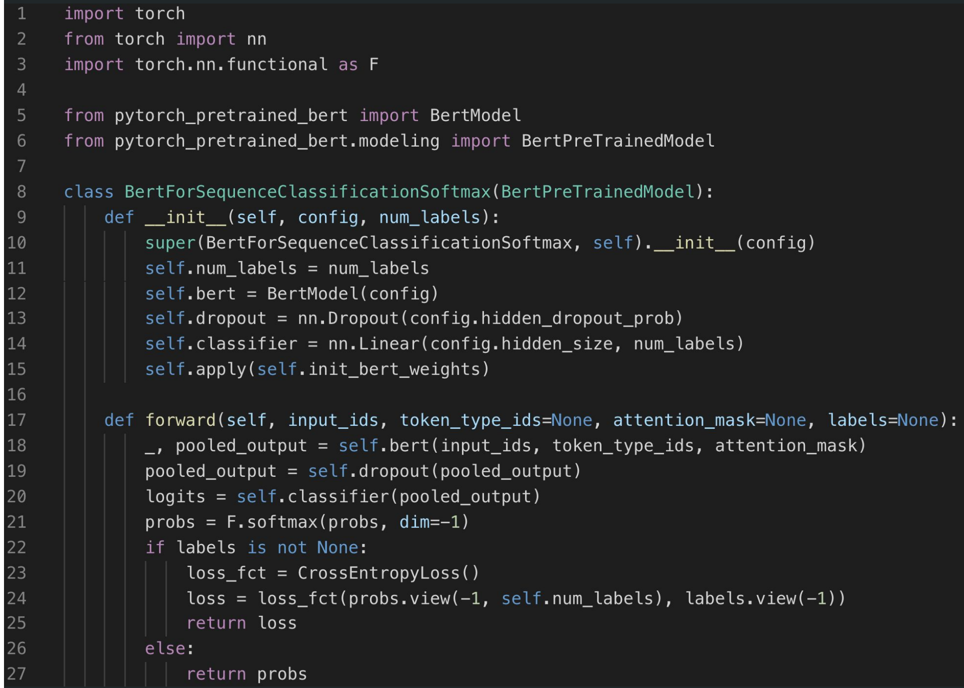
Manage containers
Amazon ECS scales your application and manages your containers for availability

https://aws.amazon.com/fargate/

training

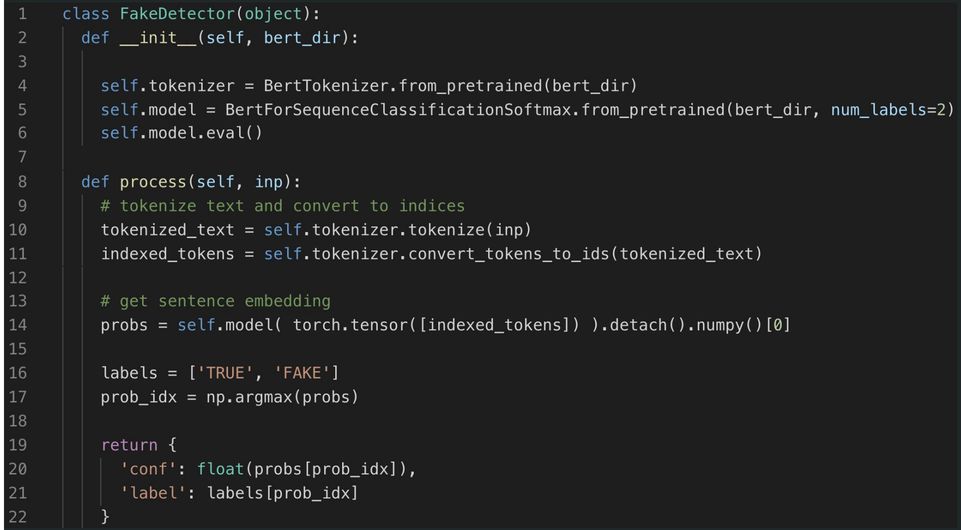
PyTorch

https://github.com/huggingface/pytorch-pretrained-BERT



```python
import torch
from torch import nn
import torch.nn.functional as F

from pytorch_pretrained_bert import BertModel
from pytorch_pretrained_bert.modeling import BertPreTrainedModel

class BertForSequenceClassificationSoftmax(BertPreTrainedModel):
    def __init__(self, config, num_labels):
        super(BertForSequenceClassificationSoftmax, self).__init__(config)
        self.num_labels = num_labels
        self.bert = BertModel(config)
        self.dropout = nn.Dropout(config.hidden_dropout_prob)
        self.classifier = nn.Linear(config.hidden_size, num_labels)
        self.apply(self.init_bert_weights)

    def forward(self, input_ids, token_type_ids=None, attention_mask=None, labels=None):
        _, pooled_output = self.bert(input_ids, token_type_ids, attention_mask)
        pooled_output = self.dropout(pooled_output)
        logits = self.classifier(pooled_output)
        probs = F.softmax(probs, dim=-1)
        if labels is not None:
            loss_fct = CrossEntropyLoss()
            loss = loss_fct(probs.view(-1, self.num_labels), labels.view(-1))
            return loss
        else:
            return probs
```

inference

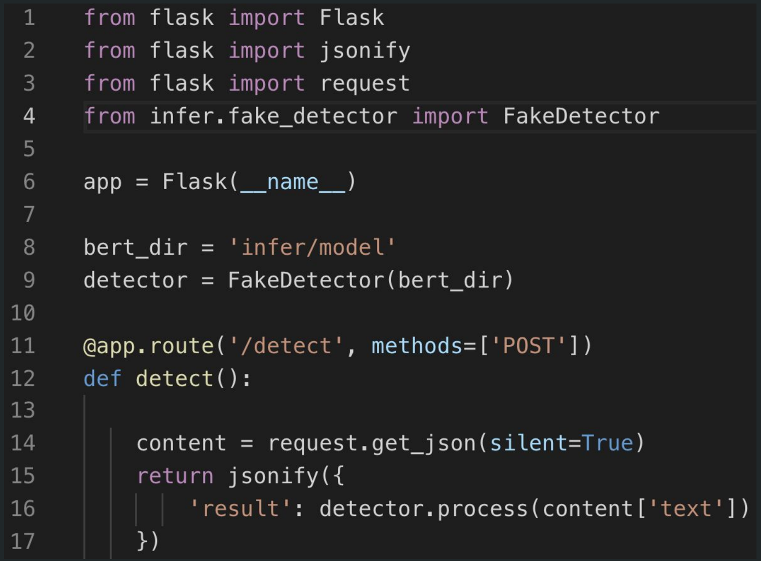


```python
class FakeDetector(object):
  def __init__(self, bert_dir):

    self.tokenizer = BertTokenizer.from_pretrained(bert_dir)
    self.model = BertForSequenceClassificationSoftmax.from_pretrained(bert_dir, num_labels=2)
    self.model.eval()

  def process(self, inp):
    # tokenize text and convert to indices
    tokenized_text = self.tokenizer.tokenize(inp)
    indexed_tokens = self.tokenizer.convert_tokens_to_ids(tokenized_text)

    # get sentence embedding
    probs = self.model( torch.tensor([indexed_tokens]) ).detach().numpy()[0]

    labels = ['TRUE', 'FAKE']
    prob_idx = np.argmax(probs)

    return {
      'conf': float(probs[prob_idx]),
      'label': labels[prob_idx]
    }
```

package

Flask

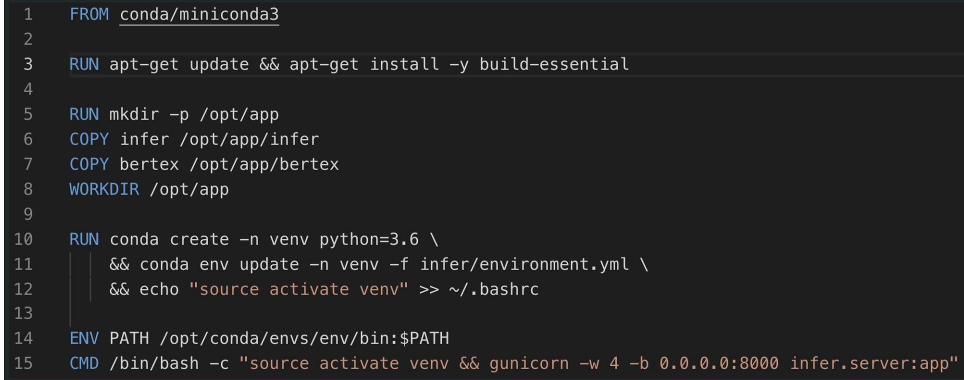
web development,
one drop at a time



```python
from flask import Flask
from flask import jsonify
from flask import request
from infer.fake_detector import FakeDetector

app = Flask(__name__)

bert_dir = 'infer/model'
detector = FakeDetector(bert_dir)

@app.route('/detect', methods=['POST'])
def detect():

    content = request.get_json(silent=True)
    return jsonify({
        'result': detector.process(content['text'])
    })
```

package

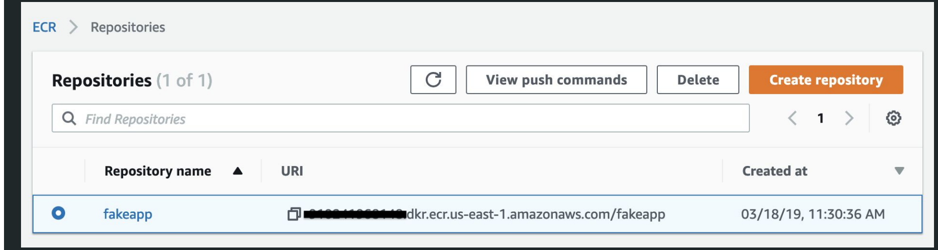
docker



```dockerfile
FROM conda/miniconda3

RUN apt-get update && apt-get install -y build-essential

RUN mkdir -p /opt/app
COPY infer /opt/app/infer
COPY bertex /opt/app/bertex
WORKDIR /opt/app

RUN conda create -n venv python=3.6 \
    && conda env update -n venv -f infer/environment.yml \
    && echo "source activate venv" >> ~/.bashrc

ENV PATH /opt/conda/envs/env/bin:$PATH
CMD /bin/bash -c "source activate venv && gunicorn -w 4 -b 0.0.0.0:8000 infer.server:app"
```

deploy

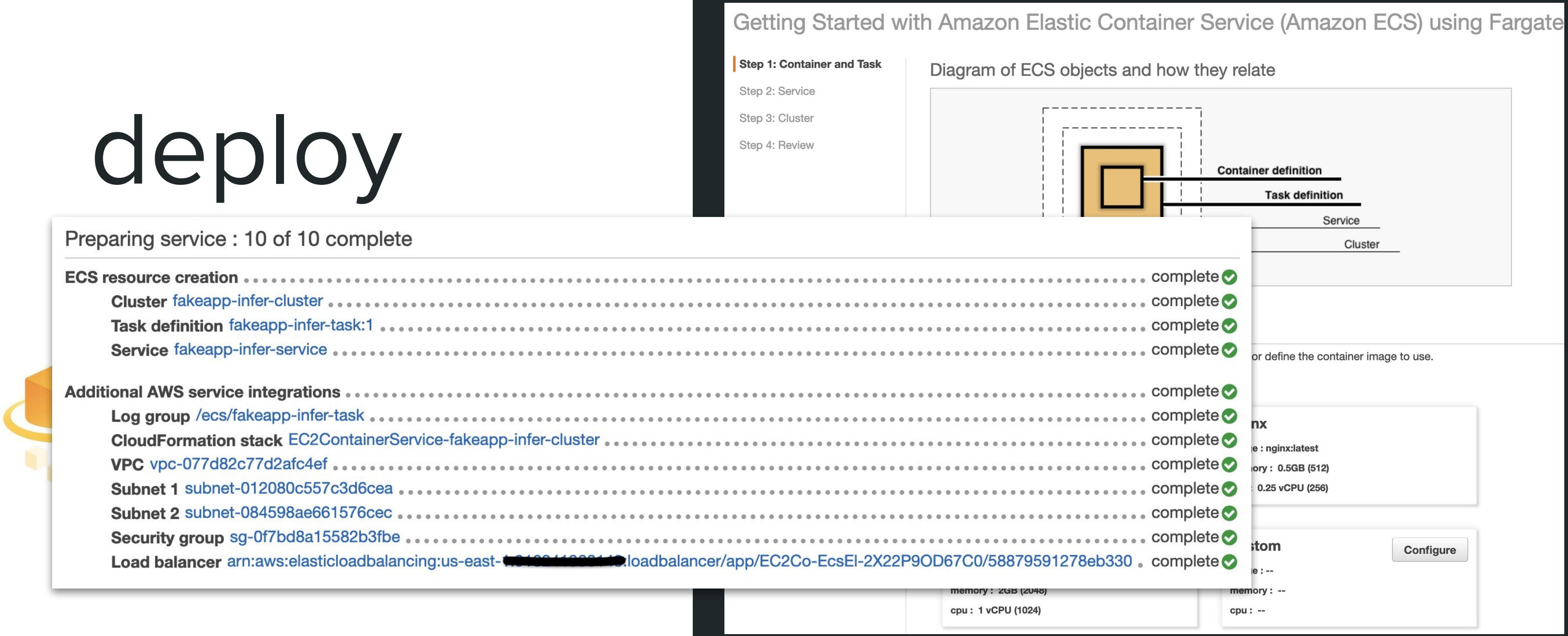
Amazon ECS



	Repository name ▲	URI	Created at
●	fakeapp	[redacted]dkr.ecr.us-east-1.amazonaws.com/fakeapp	03/18/19, 11:30:36 AM

```
> docker build

> docker tag

> docker push
```

deploy



Getting Started with Amazon Elastic Container Service (Amazon ECS) using Fargate

Step 1: Container and Task

Step 2: Service

Step 3: Cluster

Step 4: Review

Diagram of ECS objects and how they relate

or define the container image to use.

nx

e : nginx:latest

ory : 0.5GB (512)

: 0.25 vCPU (256)

stom

e : --

memory : 2GB (2048)

memory : --

cpu : 1 vCPU (1024)

cpu : --

Preparing service : 10 of 10 complete

ECS resource creation .... complete ✓

- **Cluster** fakeapp-infer-cluster .... complete ✓
- **Task definition** fakeapp-infer-task:1 .... complete ✓
- **Service** fakeapp-infer-service .... complete ✓

Additional AWS service integrations .... complete ✓

- **Log group** /ecs/fakeapp-infer-task .... complete ✓
- **CloudFormation stack** EC2ContainerService-fakeapp-infer-cluster .... complete ✓
- **VPC** vpc-077d82c77d2afc4ef .... complete ✓
- **Subnet 1** subnet-012080c557c3d6cea .... complete ✓
- **Subnet 2** subnet-084598ae661576cec .... complete ✓
- **Security group** sg-0f7bd8a15582b3fbe .... complete ✓
- **Load balancer** arn:aws:elasticloadbalancing:us-east-[redacted]:loadbalancer/app/EC2Co-EcsEl-2X22P9OD67C0/58879591278eb330 . complete ✓

let’s give it a try

discussion

aniyer7@gmail.com

/aniyer7

MITT LIV
Uppsala universitet 1852–1916
Det goda universitetet. Festskrift till Anders Hallberg
FÖLJ ERA CAPRICER
Studenten, Staden och Sanningen
UPPSALA UNIVERSITE