aws SUMMIT

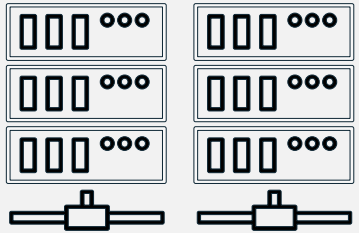
Serverless Predictions at Scale

Thomas Reske

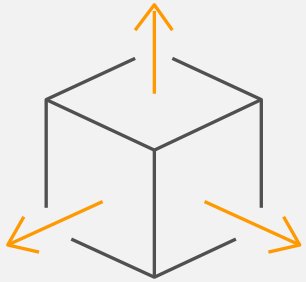
Global Solutions Architect, Amazon Web Services

Serverless computing
allows you to
build and run applications and services
without thinking about servers

What are the benefits of serverless computing?



No server management

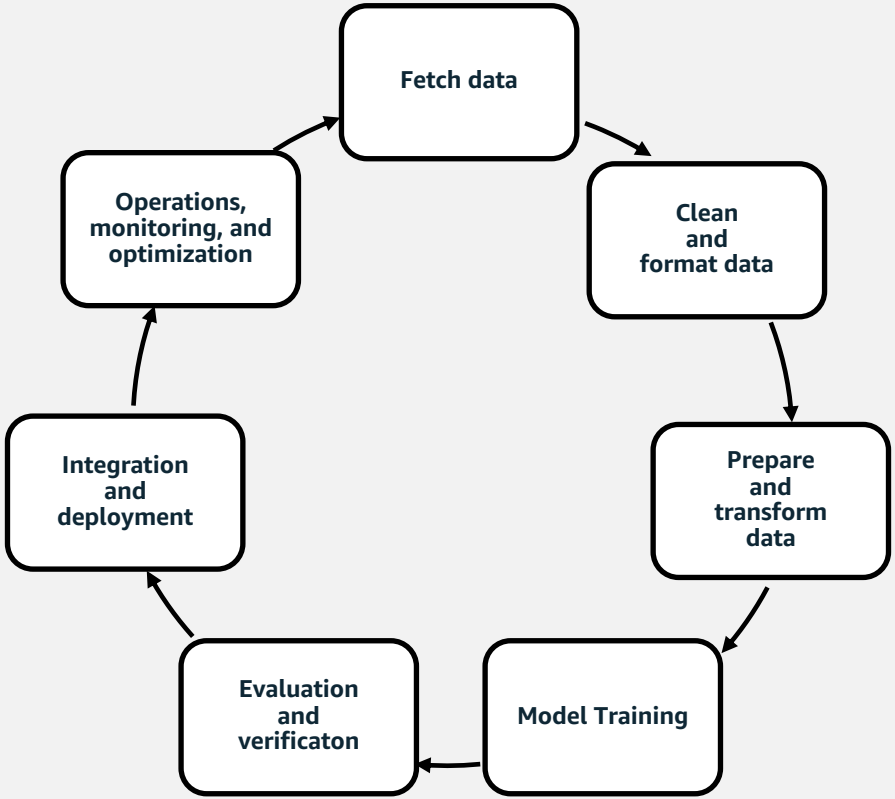


Flexible scaling

Automated high availability

No excess capacity

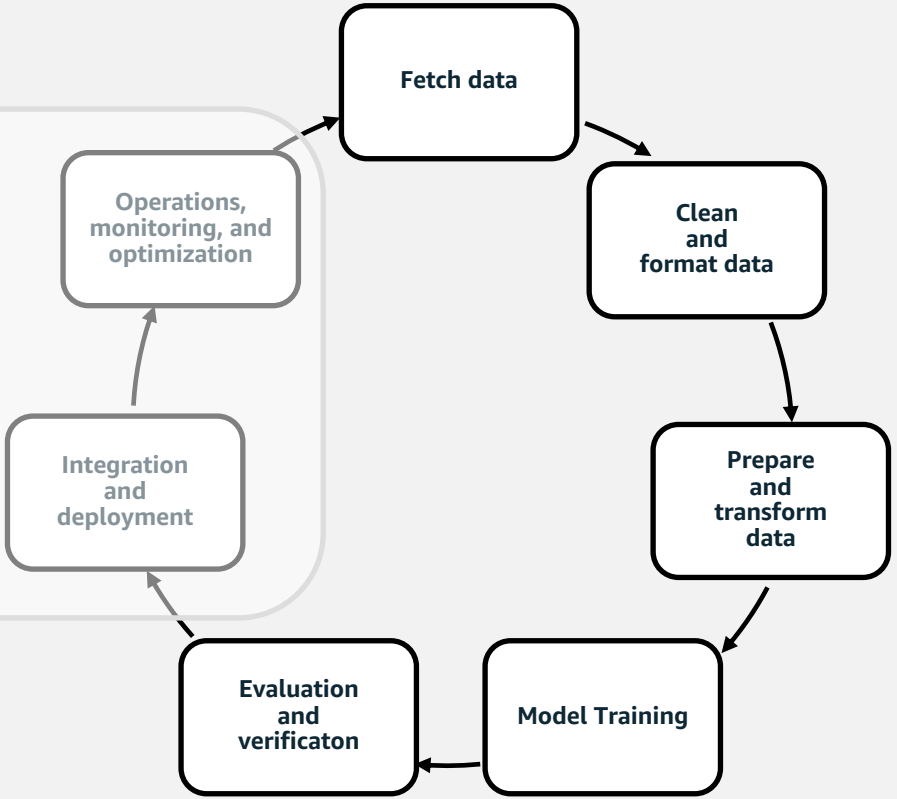
Machine Learning Process



Machine Learning Process

How can I use pre-trained models for serving predictions?

How to scale effectively?



What exactly are we deploying?

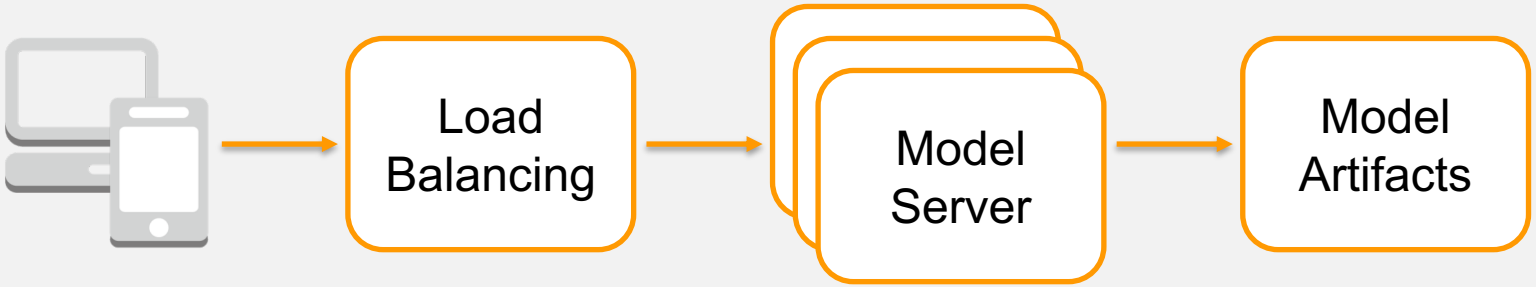
A Pre-trained Model…

- … is simply a model which has been trained previously
- Model artifacts (structure, helper files, parameters, weights, …) are files, e.g. in JSON or binary format
- Can be serialized (saved) and de-serialized (loaded) by common deep learning frameworks
- Allows for fine-tuning of models („head-start“ or „freezing“), but also to apply them to similar problems and different data („transfer learning“)
- Models can be made publicly available, e.g. in the MXNet Model Zoo

What's missing?

Blueprints?

Generic Architecture



Amazon API Gateway and AWS Lambda Integration

Trade-Offs?

Model Server for Apache MXNet

- Flexible and easy to use open-source project (Apache License 2.0) for serving deep learning models
- Supports MXNet and Open Neural Network Exchange (ONNX)
- CLI to package model artifacts and pre-configured Docker images to start a service that sets up HTTP endpoints to handle model inference requests
- More Details at https://github.com/awslabs/mxnet-model-server and https://onnx.ai/

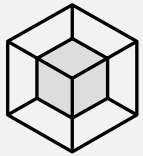
Quick Start - CLI

```
# create directory
mkdir -p ~/mxnet-model-server-quickstart && cd $_

# create and activate virtual environment
python3 -m venv ~/mxnet-model-server-quickstart/venv
source ~/mxnet-model-server-quickstart/venv/bin/activate

# install packages
pip install --upgrade pip
pip install mxnet-model-server

# run model server for Apache MXNet
mxnet-model-server --models \
  squeezenet=https://s3.amazonaws.com/model-server/models/squeezenet_v1.1/squeezenet_v1.1.model

# classify an image
curl -s -O https://s3.amazonaws.com/model-server/inputs/kitten.jpg
curl -s http://127.0.0.1:8080/squeezenet/predict -F "data=@kitten.jpg" | jq -r
```

Quick Start - Docker

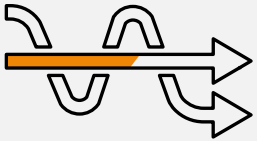
```
# create directory
mkdir -p ~/mxnet-model-server-quickstart-docker && cd $_

# run model server for Apache MXNet
docker run -itd --name mms -p 8080:8080 awsdeeplearningteam/mms_cpu \
  mxnet-model-server start --mms-config /mxnet_model_server/mms_app_cpu.conf

# classify an image
curl -s -O https://s3.amazonaws.com/model-server/inputs/kitten.jpg
curl -s http://127.0.0.1:8080/squeezenet/predict -F "data=@kitten.jpg" | jq -r
```

Why developers love containers



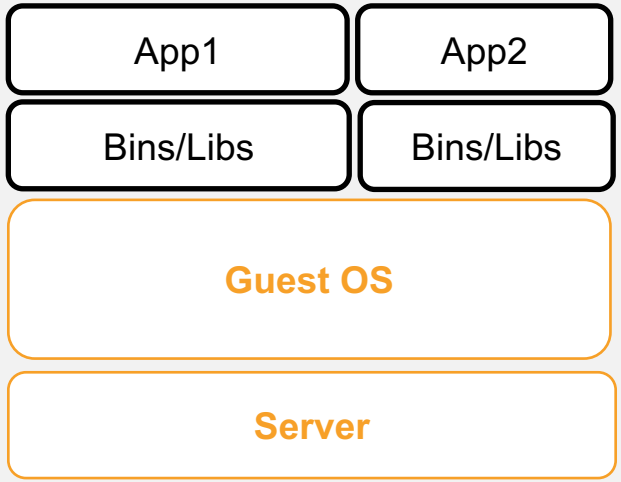
Software Packaging

Distribution

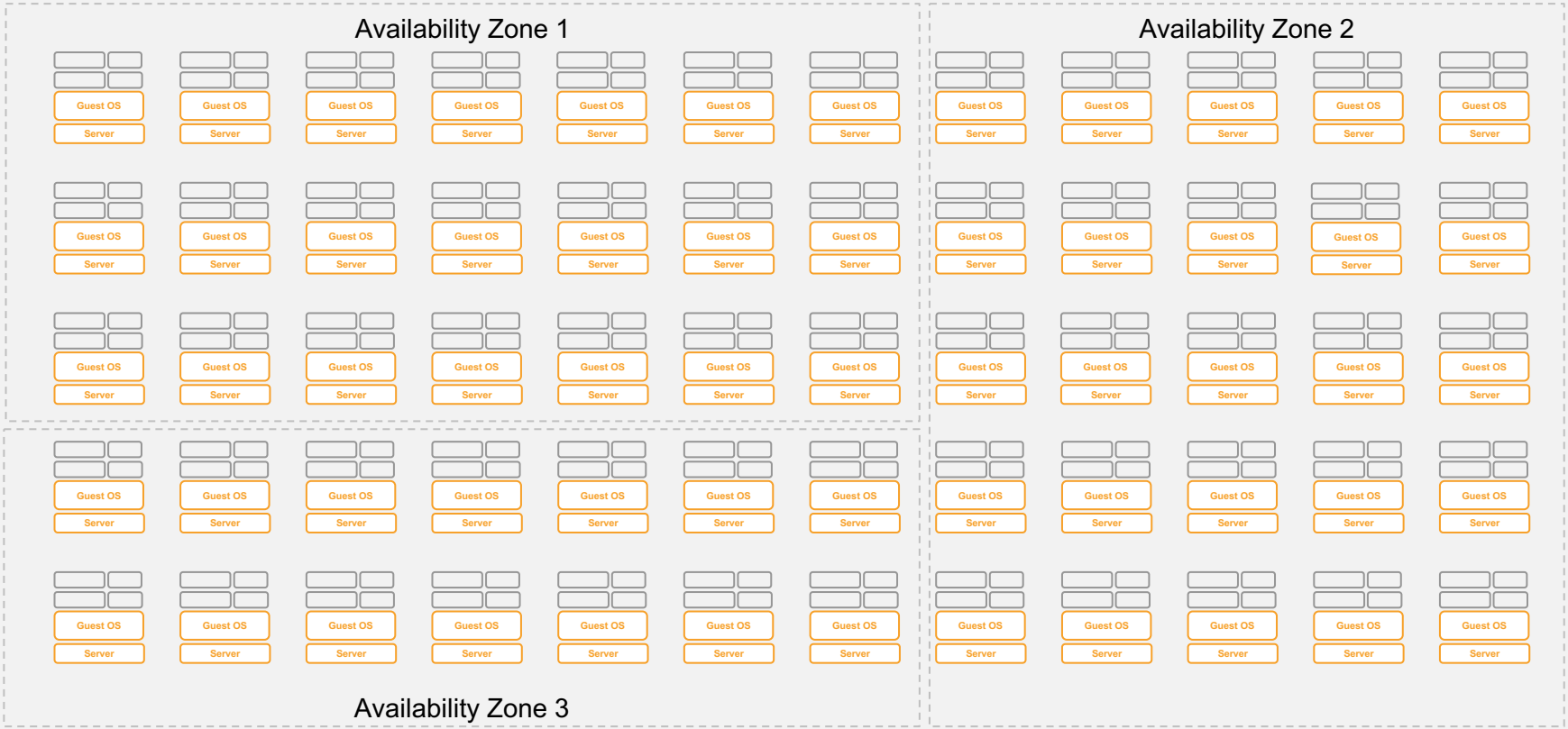


Immutable Infrastructure

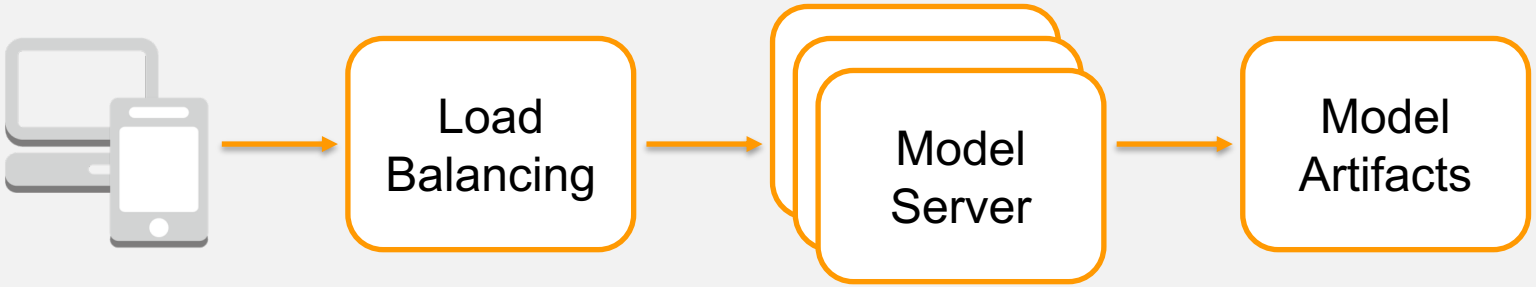
Managing One Host Is Straightforward



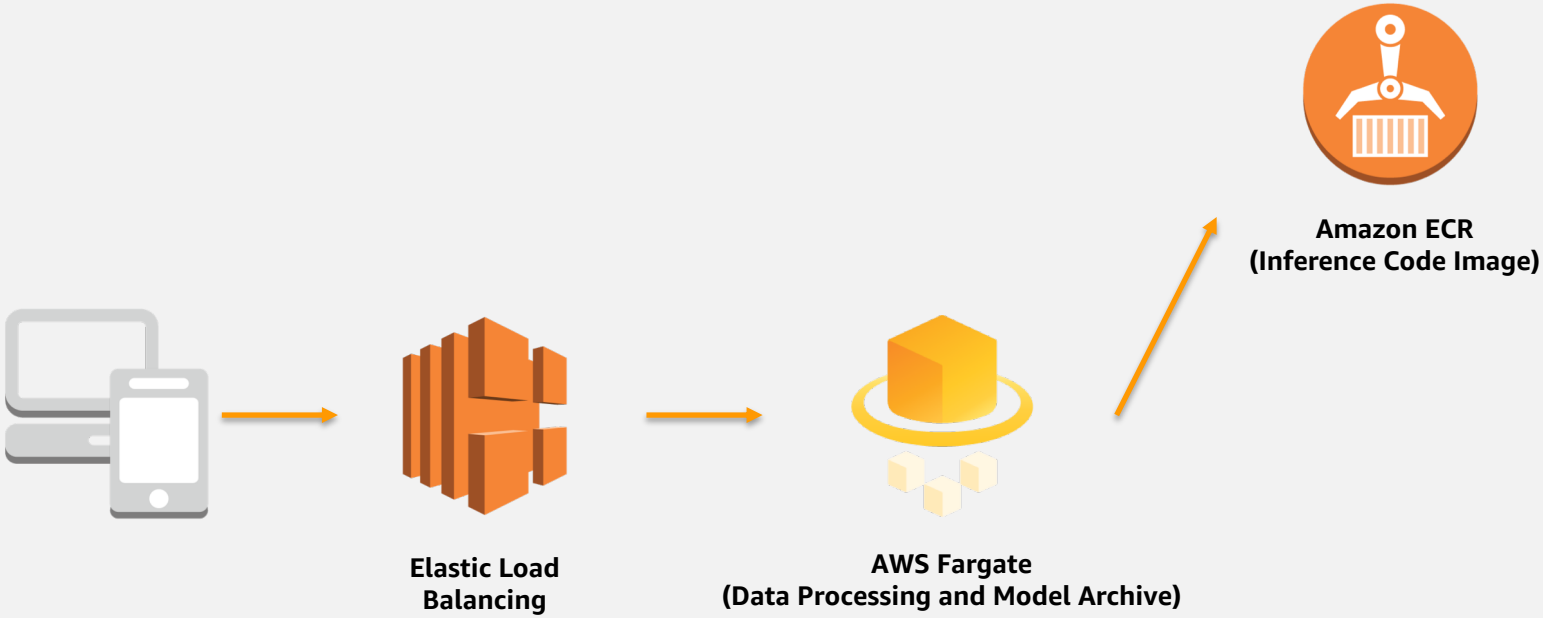
Managing a Fleet Is Hard



Generic Architecture



AWS Fargate and Apache MXNet Model Server



Quick Start – AWS Fargate CLI – Basic Task Definition

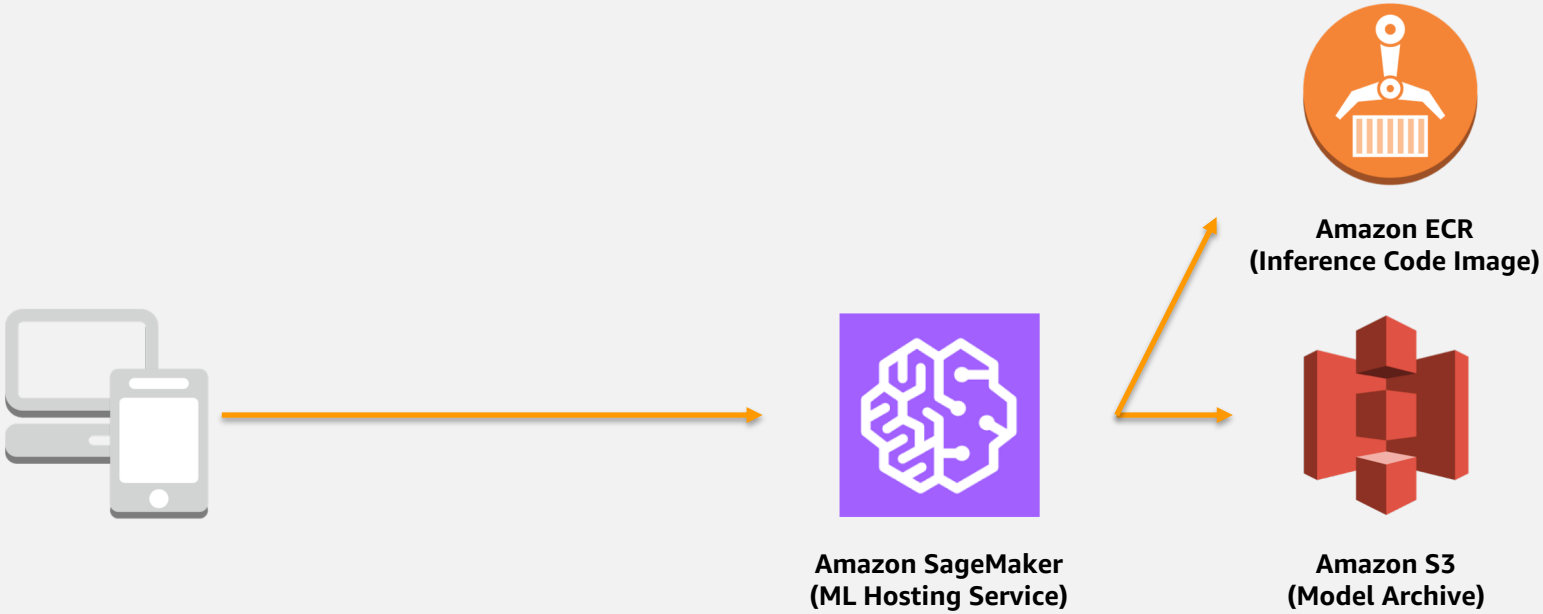
```
{
    "family": "mxnet-model-server-fargate",
    "networkMode": "awsvpc",
    "containerDefinitions": [
        {
            "name": "mxnet-model-server-fargate-app",
            "image": "awsdeeplearningteam/mms_cpu",
            "portMappings": [
                {
                    "containerPort": 8080,"hostPort": 8080,"protocol": "tcp"
                }
            ],
            "essential": true,
            "entryPoint": [
                "mxnet-model-server","start","--mms-config","/mxnet_model_server/mms_app_cpu.conf"
            ]
        }
    ],
    "requiresCompatibilities": [
        "FARGATE"
    ],
    "cpu": "256",
    "memory": "512"
}
```

Quick Start – AWS Fargate CLI

```
# register task definition
aws ecs register-task-definition --cli-input-json file://basic_taskdefinition.json

# create AWS Fargate cluster
aws ecs create-cluster --cluster-name mxnet-model-server

# create service
aws ecs create-service \
  --cluster mxnet-model-server \
  --service-name mxnet-model-server-fargate-service \
  --task-definition mxnet-model-server-fargate:1 \
  --desired-count 2 \
  --launch-type "FARGATE" \
  --network-configuration "awsvpcConfiguration={subnets=[$MMS_SUBNETS],securityGroups=[$MMS_SG_ID]}" \
  --load-balancers "targetGroupArn=$MMS_TG_ARN,containerName=$MMS_CONTAINER_NAME,containerPort=8080"
```

Trade-Offs?

Amazon SageMaker



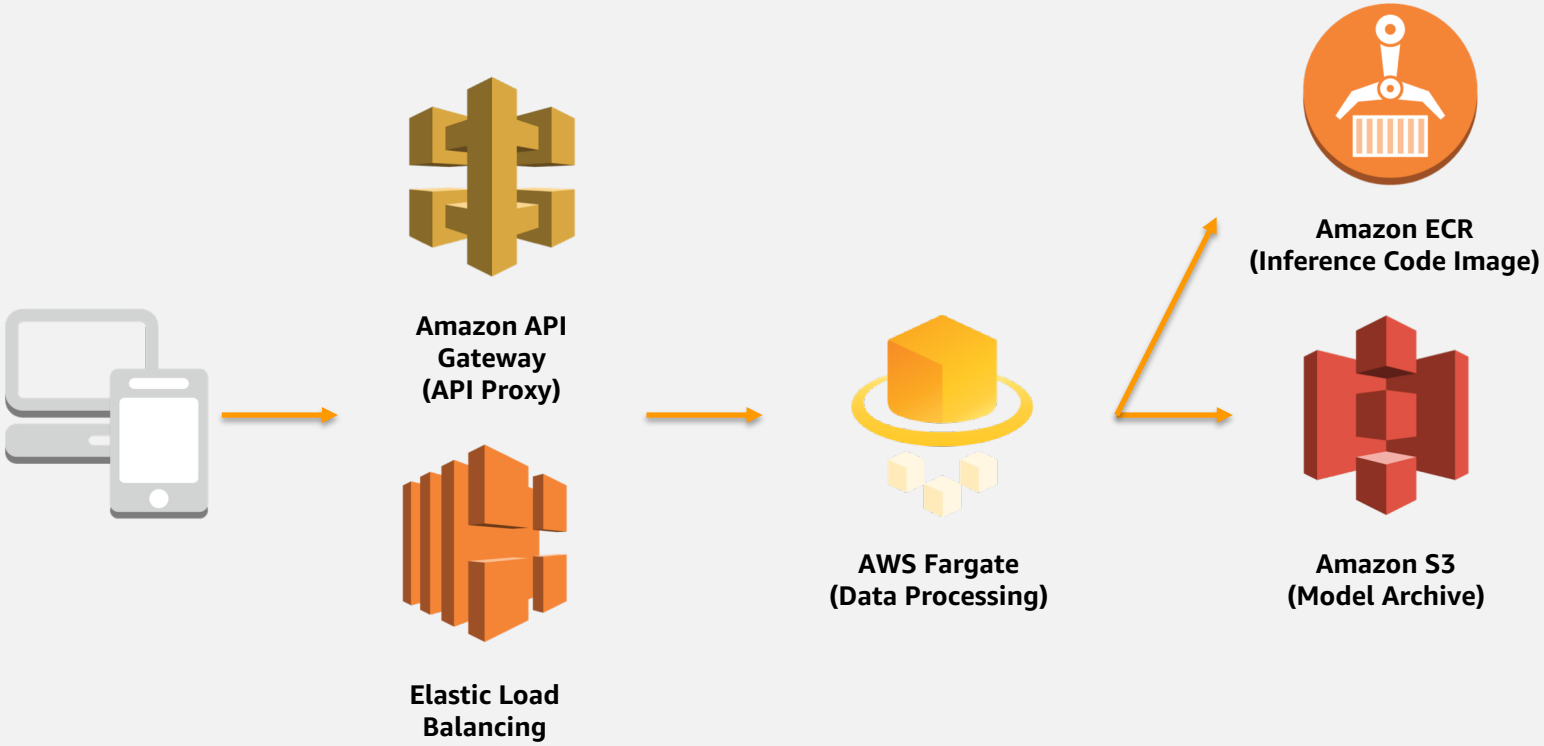
Quick Start – Amazon SageMaker CLI

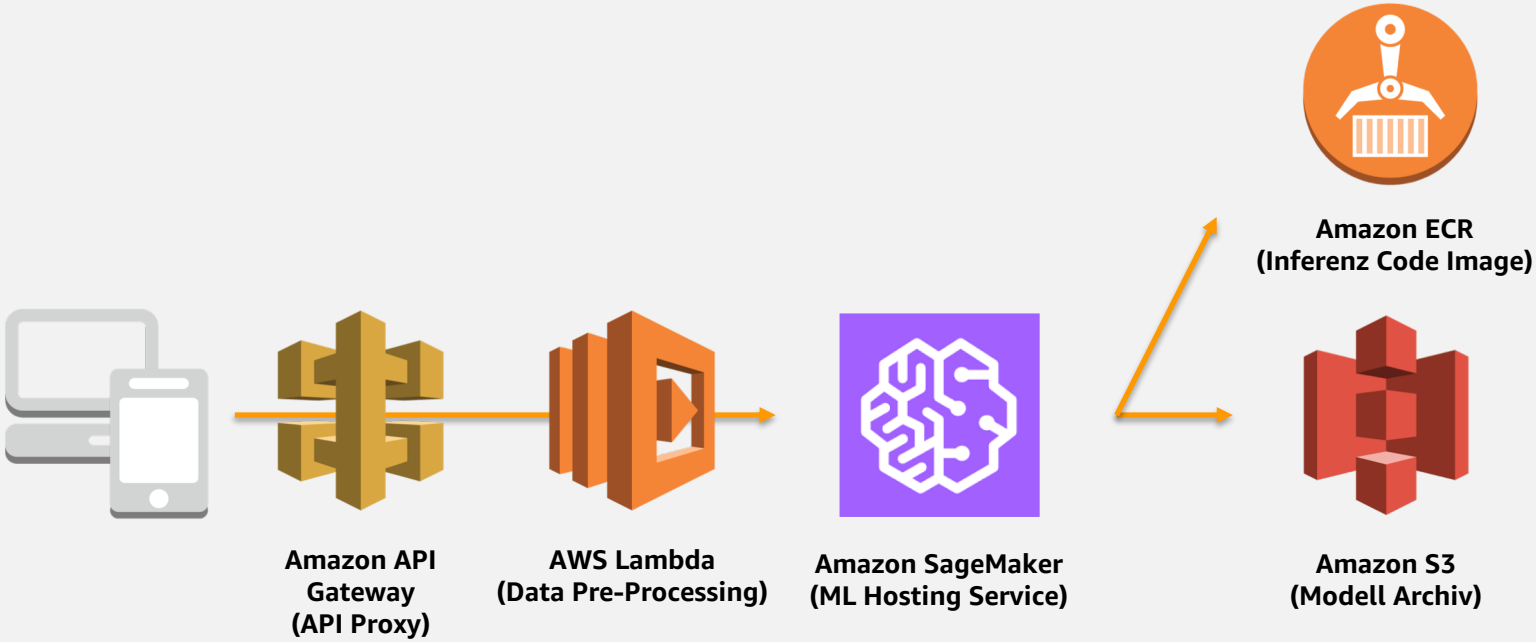
```
# register model
aws sagemaker create-model \
  --model-name image-classification-model \
  --primary-container Image=$INFERENCE_IMG,ModelDataUrl=$MODEL_DATA_URL \
  --execution-role-arn $EXECUTION_ROLE_ARN

# register next-generation model
aws sagemaker create-model \
  --model-name image-classification-model-nextgen \
  --primary-container Image=$INFERENCE_IMG_NG,ModelDataUrl=$MODEL_DATA_URL_NG \
  --execution-role-arn $EXECUTION_ROLE_ARN
```

Quick Start – Amazon SageMaker CLI

```
# create endpoint configuration
aws sagemaker create-endpoint-config \
  --endpoint-config-name "image-classification-config" \
  --production-variants '[
      {
        "VariantName" : "A", "ModelName" : "image-classification-model",
        "InitialInstanceCount" : 1, "InstanceType" : "ml.t2.medium",
        "InitialVariantWeight" : 8
      },
      {
        "VariantName" : "B", "ModelName" : "image-classification-model-nextgen",
        "InitialInstanceCount" : 1, "InstanceType" : "ml.t2.medium",
        "InitialVariantWeight" : 2
      }
    ]'

# create endpoint
aws sagemaker create-endpoint \
  --endpoint-name image-classification  \
  --endpoint-config-name image-classification-endpoint-config \
&& aws sagemaker wait endpoint-in-service --endpoint-name image-classification
```

Other Options?



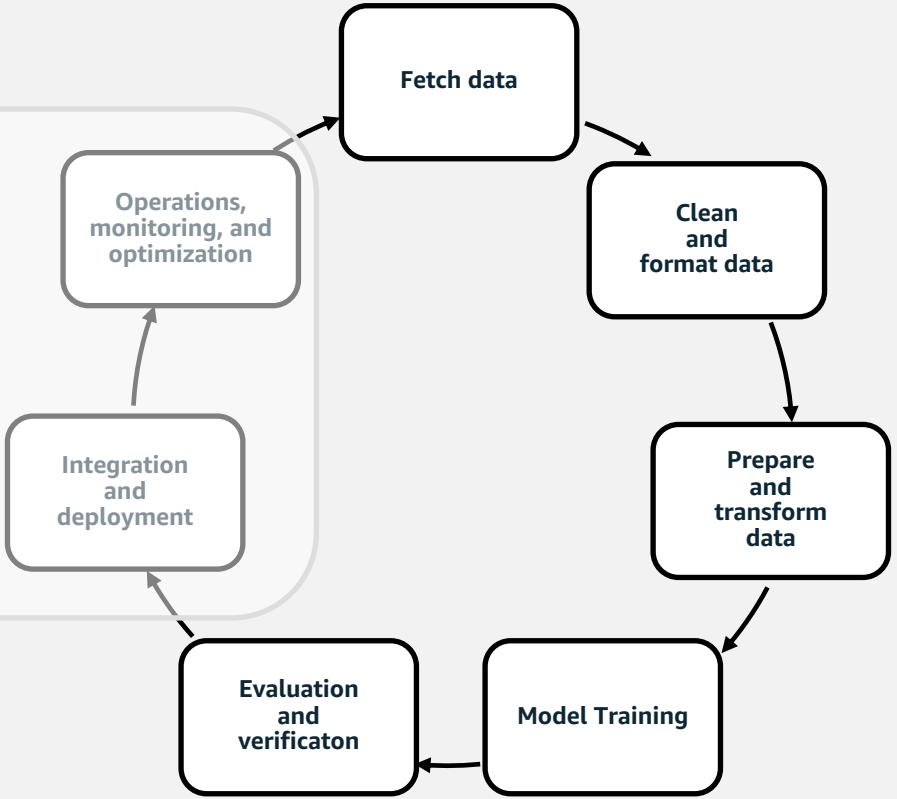


https://medium.com/@julsimon/using-chalice-to-serve-sagemaker-predictions-a2015c02b033

Summary

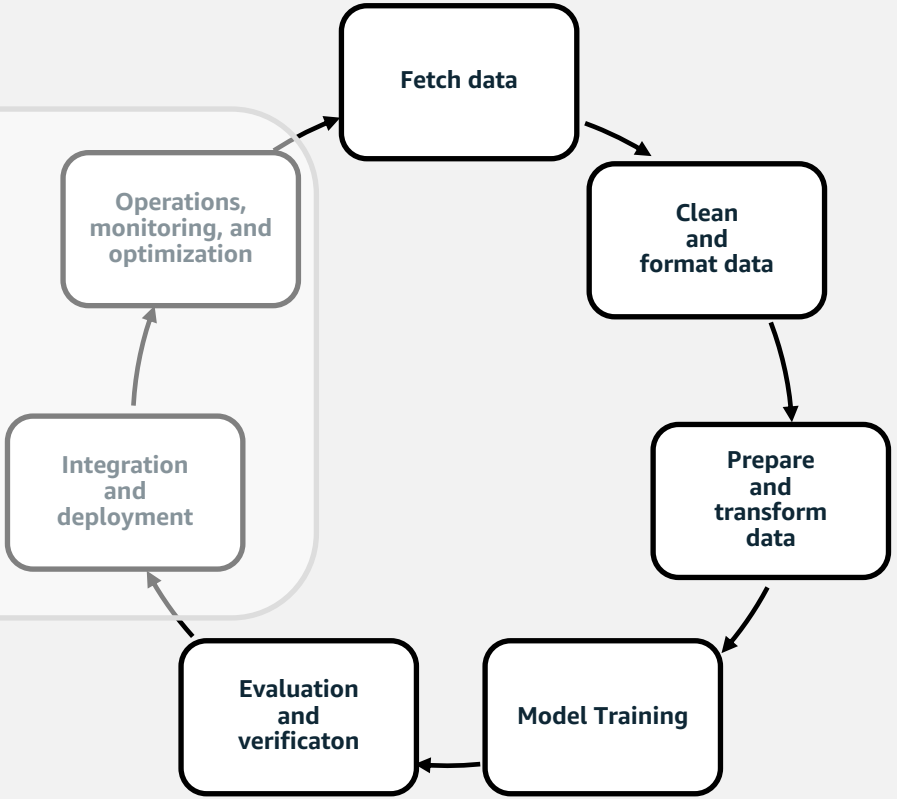
How can I use pre-trained models for serving predictions?

How to scale effectively?



Summary

1. API Gateway + Lambda + S3
2. ELB + AWS Fargate
3. Amazon SageMaker



Resources

- **Serverless Computing and Applications**
  https://aws.amazon.com/serverless/
- **Seamlessly Scale Predictions with AWS Lambda and MXNet**
  https://aws.amazon.com/blogs/compute/seamlessly-scale-predictions-with-aws-lambda-and-mxnet/
- **Model Server for Apache MXNet**
  https://github.com/awslabs/mxnet-model-server
- **Serverless Predictions at Scale**
  https://github.com/aws-samples/aws-ai-bootcamp-labs/blob/master/serverless_predictions.MD
- **Serverless Inference with MMS on AWS Fargate**
  https://github.com/awslabs/mxnet-model-server/blob/master/docs/mms_on_fargate.md
- **Introducing Model Server for Apache MXNet**
  https://aws.amazon.com/blogs/machine-learning/introducing-model-server-for-apache-mxnet/
- **Using Chalice to serve SageMaker predictions**
  https://medium.com/@julsimon/using-chalice-to-serve-sagemaker-predictions-a2015c02b033

Before you go...

Sessions (AI & ML)

Wednesday, June 6		Thursday, June 7	
13:00-15:00	**AWS ML & AI Deep Dive** Julien Simon, AWS	13:00-14:00	**Machine Learning & AI at Amazon** Ian Massingham, AWS
15:00-16:00	**Build Your Recommendation Engines on AWS Today** Yotam Yarden & Cyrus Vahid, AWS	14:00-15:00	**The Machine Learning Process: From Business Model to Machine Learning in Production** Constantin Gonzalez, AWS
16:00-17:00	**Serverless Predictions at Scale** Thomas Reske, AWS	15:00-16:00	**Using Machine Learning Algorithms to create Market Transparency** Dr. Markus Schmidberger & Dr. Markus Ludwig, Scout24 AG
		16:00-17:00	**Needle in the Live Video Cloud haystack – AI/ML for mass live acquisition** Matija Tonejc & David Querq, Make.TV

Please complete the session survey in the summit mobile app.

aws SUMMIT

Thank You!