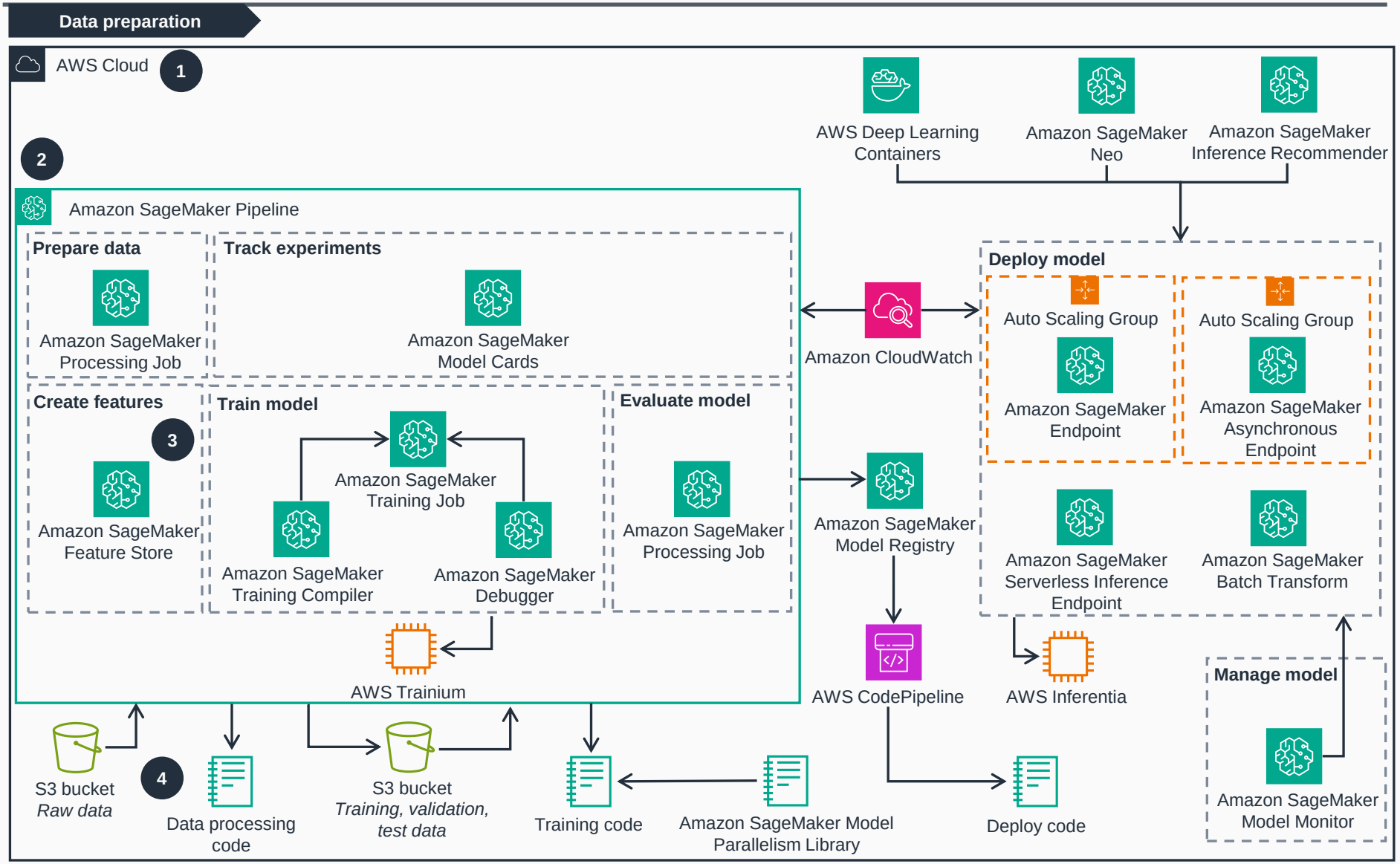
Guidance for Optimizing MLOps for Sustainability on AWS

Data preparation

This architecture diagram helps you align to MLOps sustainability goals. This slide focuses on data preparation.



1. Choose a Region based on both business requirements and sustainability goals. When regulations and legal aspects allow, use one of the **AWS Regions where the electricity consumed is attributable to 100% renewable energy** or Regions where the grid has a published carbon intensity that is lower than other locations (or Regions). When selecting a Region, aim to minimize data movement across networks—store your data close to your producers and train your models close to your data.

2. Adopt a serverless architecture for your pipeline so it only provisions resources when work needs to be done. Use **Amazon SageMaker Pipeline** to avoid maintaining compute infrastructure at all times. You can extend a template provided by **Amazon SageMaker Projects**, such as **MLOps template for model building, training, deployment, and Amazon SageMaker Model Monitor**.

3. Reduce duplication and re-run of feature engineering code across teams and projects by using **Amazon SageMaker Feature Store**.

4. Reduce the volume of data to be stored and adopt sustainable storage options to limit the carbon impact of your workload. Use energy-efficient, archival-class storage for infrequently accessed data, such as your raw data. If you can easily re-create an infrequently accessed dataset, like training, validation and test data, use the **Amazon Simple Storage Service (Amazon S3) One Zone-Infrequent Access** class to minimize the total data stored. Manage the lifecycle of all your data and automatically enforce deletion timelines to minimize the total storage requirements of your workload using **Amazon S3 Lifecycle policies**. **Amazon S3 Intelligent-Tiering** will automatically move your data to the most energy-efficient access tier when access patterns change. Define data retention periods that support your sustainability goals while meeting your business requirements, not exceeding them.

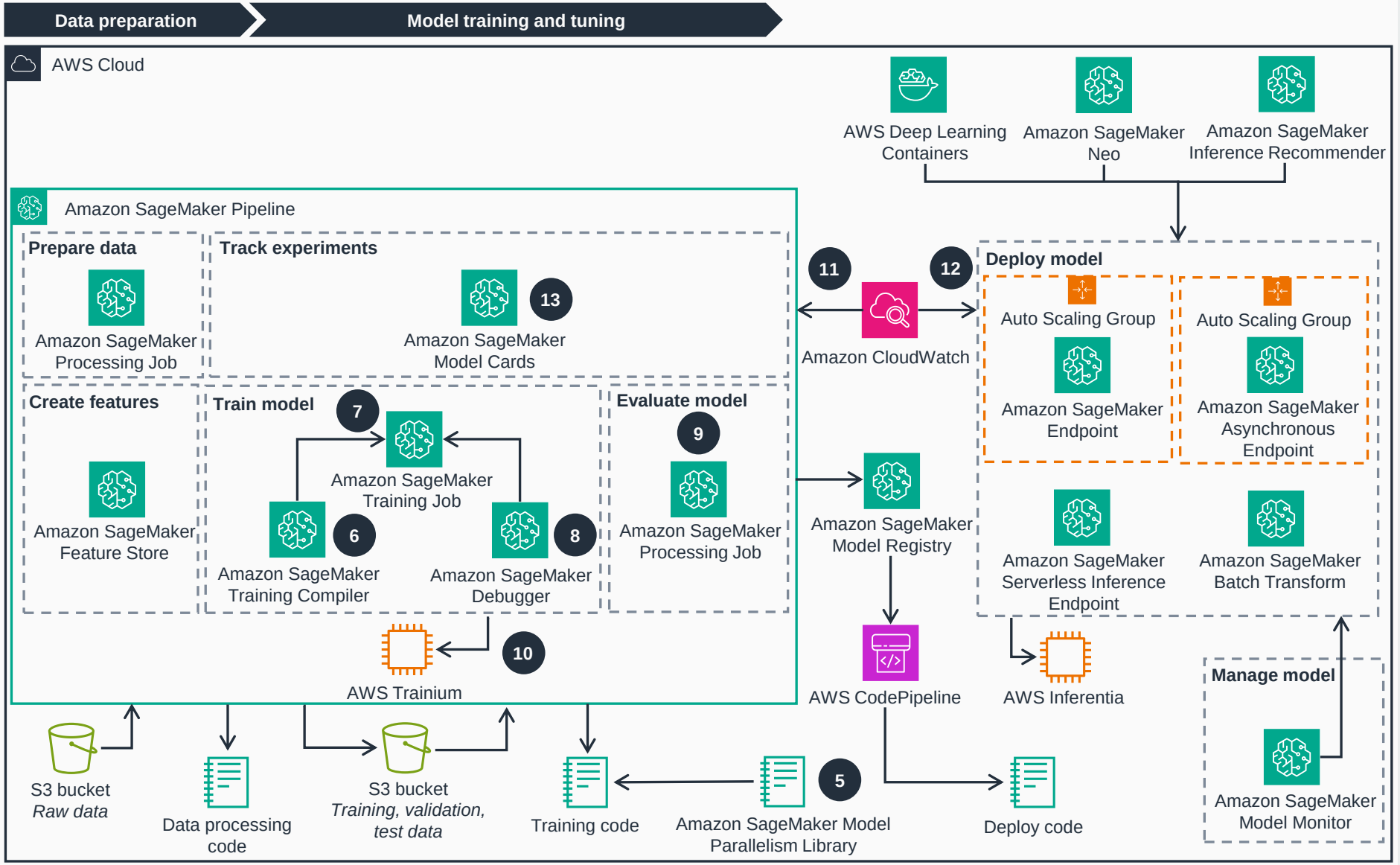
Reviewed for technical accuracy November 21, 2024

AWS Reference Architecture

Guidance for Optimizing MLOps for Sustainability on AWS

Model training and tuning

This architecture diagram helps you align to MLOps sustainability goals. This slide focuses on model training and tuning.



5 For distributed training of large deep learning models, use **Amazon SageMaker Model Parallelism Library** in your training code to maximize usage of graphics processing units (GPUs).

6 Use **Amazon SageMaker Training Compiler** to compile your deep learning models from their high-level language representation to hardware-optimized instructions to reduce training time. This can speed up deep learning model training by up to 50%.

7 Use **Bayesian optimization search** rather than random or grid search. Bayesian search typically requires 10 times fewer jobs than random search to find the best hyperparameters.

8 Use **Amazon SageMaker Debugger** to detect **under-utilization of system resources** and identify training problems. **SageMaker Debugger built-in rules** can monitor your training jobs and automatically stop them upon bug detection.

9 Define acceptable performance criteria: evaluate the accuracy of your models using **Amazon SageMaker Processing Jobs** and make trade-offs between your model's accuracy and its carbon footprint. Establish performance criteria that support your sustainability goals while meeting your business requirements, not exceeding them.

10 Use **AWS Trainium** to train deep learning models at up to 52% less energy than comparable **Amazon Elastic Compute Cloud (Amazon EC2)** instances. Consider **Managed Spot Training**, which takes advantage of unused **Amazon EC2** capacity, to improve your overall resource efficiency and reduce idle capacity of cloud resources.

11 Right-size your training jobs with **Amazon CloudWatch** metrics.

12 Reduce the volume of **CloudWatch** logs you keep. By **setting limited retention time** for your notebooks and training logs, you'll avoid unnecessary log storage.

13 Document your model's environmental impact using **Amazon SageMaker Model Cards**.

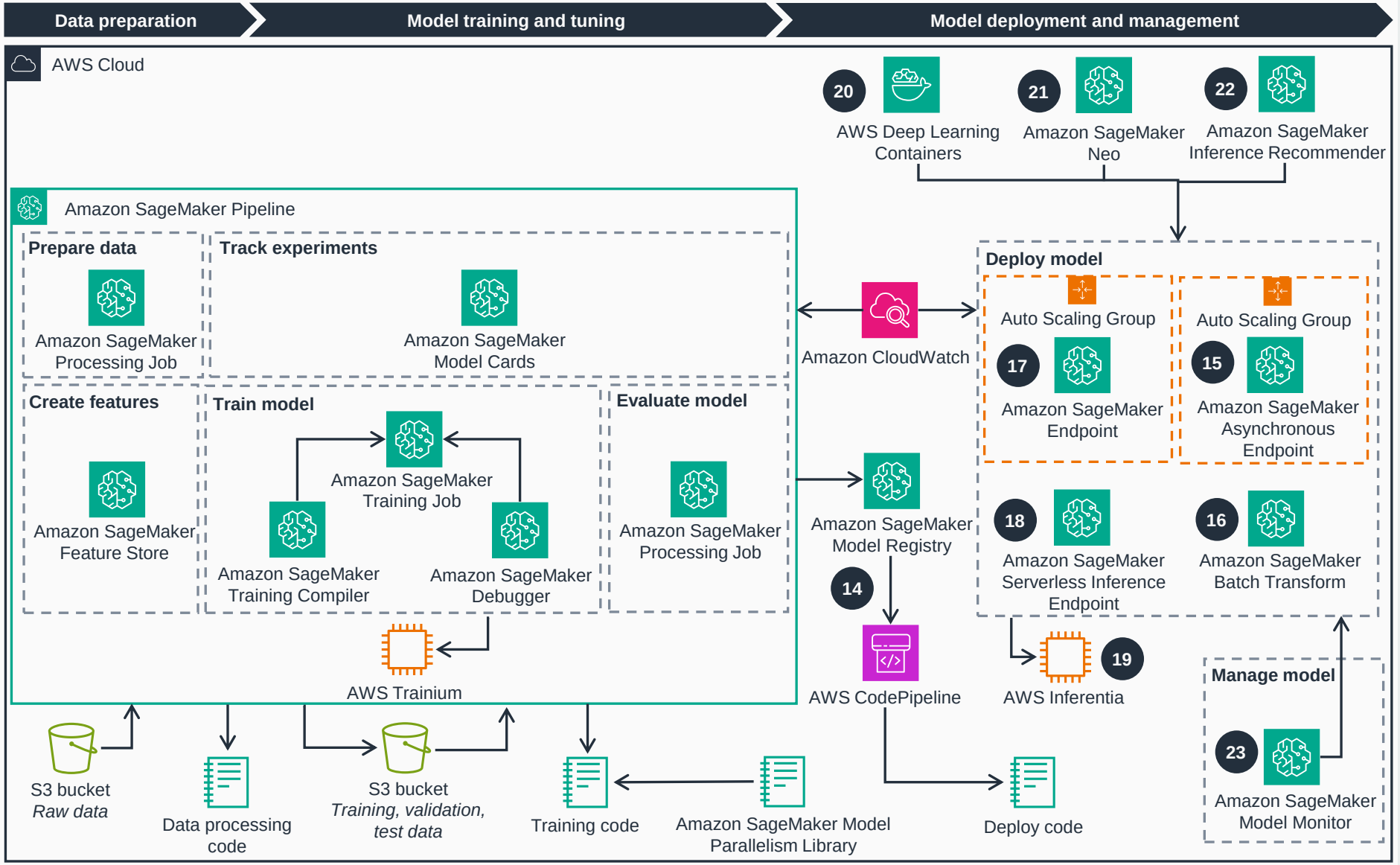
Reviewed for technical accuracy November 21, 2024

AWS Reference Architecture

Guidance for Optimizing MLOps for Sustainability on AWS

Model deployment and management

This architecture diagram helps you align to MLOps sustainability goals. This slide focuses on model deployment and management.



14 Automate the deployment of your models. Use **Amazon SageMaker Model Registry** and **AWS CodePipeline** to run your deployment code.

15 If your users can tolerate latency, deploy your model on **Amazon SageMaker Asynchronous Endpoints** with auto scaling groups to reduce idle resources between tasks and minimize the impact of load spikes.

16 When you don't need real-time inference, use **Amazon SageMaker Batch Transform**. Unlike persistent endpoints, clusters are decommissioned when batch transform jobs finish.

17 Deploy **multiple models behind a single Amazon SageMaker endpoint** with **auto scaling inference endpoints**, which is more sustainable than deploying a single model behind one endpoint.

18 If your workload has intermittent or unpredictable traffic, use **Amazon SageMaker Serverless Inference Endpoints**, which automatically launch compute resources and scale depending on traffic.

19 Use **AWS Inferentia** to deploy your deep learning models, which provides up to 50% better performance per watt over comparable **EC2** instances.

20 For Large Model Inference (LMI), use tensor parallelization available in the **Deep learning containers for LMI** to reduce latency.

21 Improve efficiency of your models by compiling them into optimized forms with **Amazon SageMaker Neo**.

22 Right-size your endpoints by using metrics from **CloudWatch** or **Amazon SageMaker Inference Recommender**, which recommends the proper instance type to host your model.

23 Monitor your ML model in production using **SageMaker Model Monitor**, automate model drift detection, and only retrain when predictive performance has fallen below defined key performance indicators (KPIs).

Reviewed for technical accuracy November 21, 2024

AWS Reference Architecture