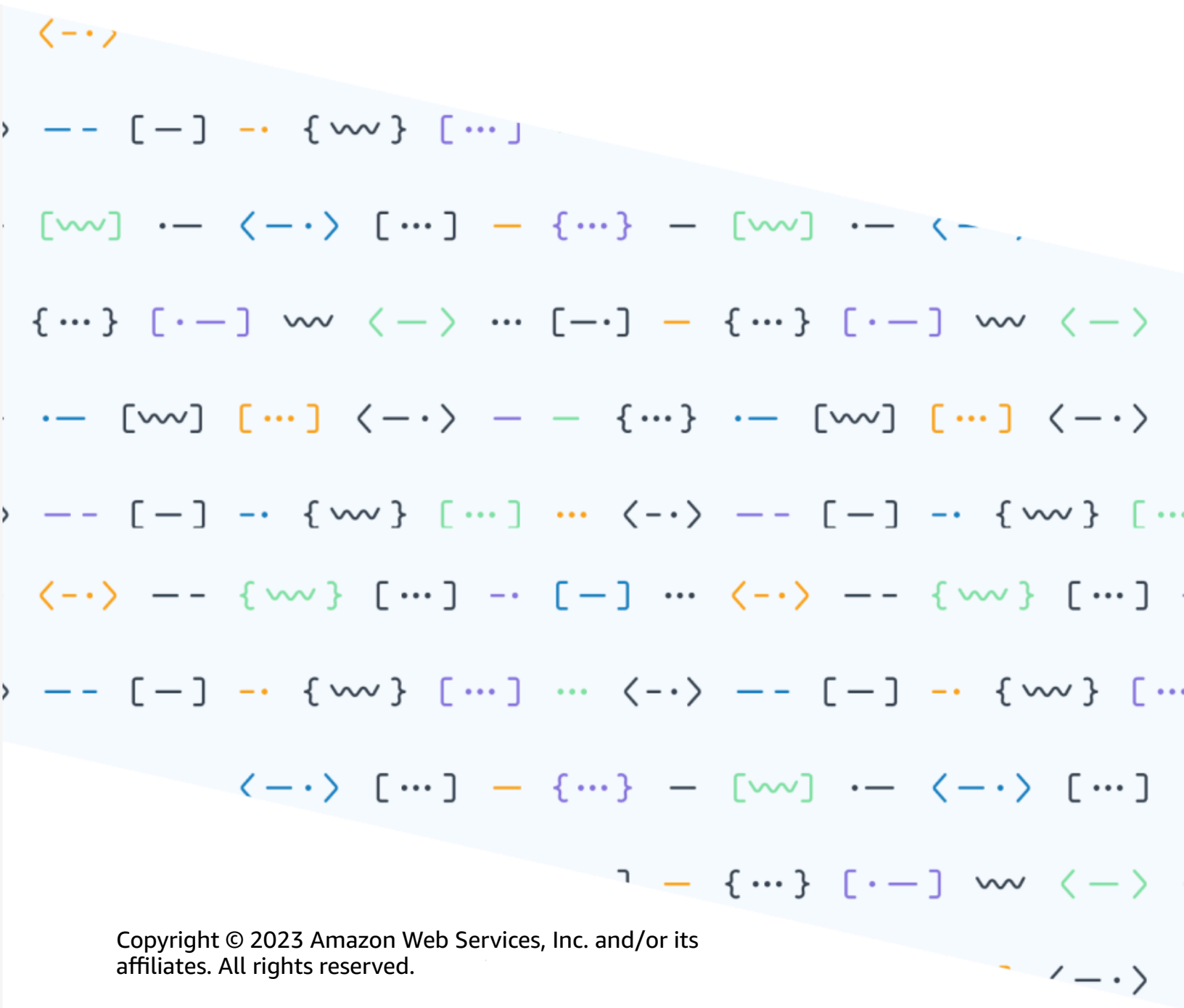
AWS Whitepaper

Hybrid Machine Learning



Hybrid Machine Learning: AWS Whitepaper

Table of Contents

Hybrid Machine Learning

Publication date: **August 15, 2021** (*Document history*)

The purpose of this whitepaper is to outline known considerations, design patterns, and solutions that customers can leverage today when considering hybrid dimensions of the Amazon Web Services (AWS) artificial intelligence/machine learning (AI/ML) stack across the entire ML lifecycle.

Due to the scalability, flexibility, and pricing models enabled by the cloud, we at AWS continue to believe that the majority of ML workloads are better suited to run in the cloud in the long haul. However, given that less than 5% of overall IT spend is allocated for the cloud, the actual amount of IT spend on-premises is north of 95%. This tells us that there is a sizeable underserved market.

Change is hard - particularly for enterprises. The complexity, magnitude, and length of migrations can be a perceived barrier to getting started. For these customers, we propose hybrid ML patterns *as an intermediate step in their cloud and ML journey.* Hybrid ML patterns are those that involve a minimum of two compute environments, typically local compute resources such as personal laptops or corporate data centers, and the cloud. The following section outlines a full introduction to the concept of hybrid ML.

We think customers win when they deploy a workload that touches the cloud to get some value, and we at AWS are committed to supporting any customer's success, even if only a few percentage points of that workload hit the cloud today.

This document is intended for individuals who already have a baseline understanding of machine learning, in addition to Amazon SageMaker. We will not dive into best practices for SageMaker per se, nor into best practices for storage or edge services. Instead, we will focus explicitly on hybrid workloads, and refer readers to resources elsewhere as necessary.

Introduction

Developing successful technology that applies machine learning is a non-trivial task. Datasets vary from bytes to petabytes, objects to file systems, text to vision, tables to logs. Software frameworks supporting machine learning models evolve rapidly, undergoing potentially major changes multiple times a year, if not quarter or month. Data science projects revolve around the skill levels of your team, interest from your business stakeholders, quality and availability of your datasets and models, and customer adoption.

Companies who adopt a cloud-native approach realize its value when they marry compute capacity with the needs of their business, unleashing their highly valued technical resources to focus on building features that differentiate their business, rather than taking on the burden of managing and maintaining their own underlying infrastructure.

But for those companies born before the cloud, in many cases even before generalized computers, how can they take concrete steps to realize the potential hidden within their own datasets and talent? Even for newer companies founded more recently, potentially those that made an informed decision to build on-premises, how can they realize the value of newly launched cloud services when the early requirements that were once infeasible on the cloud are now within reach?

For customers who want to integrate the cloud with existing on-premises ML infrastructure, we propose a series of tenets to guide our discussion of a world-class hybrid machine learning experience:

1. **Seamless management experience.** Customers need end-to-end ML management across multiple compute environments, limiting the burden of administrative overhead while providing sufficient depth and modularity to successfully operate complex tasks.
2. **Tunable latency.** Technology consumers enjoy using applications that respond well within the timelines of their moment-to-moment expectations, and designers of these applications understand the criticality of time-bound SLA's. Engineers want to work with ML models that can respond to an app's request within single digit milliseconds round trip, regardless of where they are hosted, and they will prioritize solutions that do so. While not every customer requires response times at ultra-low latency levels, we note that generally speaking, faster is better.
3. **Fast time-to-value.** Builders do not want to waste engineering effort learning new frameworks and interfaces. They expect solutions to be easy to use, with simple interfaces, and not requiring significant amounts of platform-specific engineering to execute a task.
4. **Flexible.** Customers need compute paradigms that provide the flexibility their business demands. Models can come from open-source frameworks such as TensorFlow, PyTorch, R, and MXNet, or third party SaaS-provided models such as those from DataRobot, H2O, or Databricks. ML applications may need to serve real-time responses to billions of users worldwide, or batch responses to 10s of users. Service providers should anticipate and provide world-class experiences for deploying all of these.
5. **Low cost**. Not only do customers want transparency in their cost structure, they need to see a clear economic advantage to computing in the cloud relative to developing locally. If they are able to run compute locally for a lower upfront dollar-value, they will. Service providers need to anticipate this and compete on cost with respect to local compute options.

6. **End in the cloud.** If there was any doubt that cloud computing is the way of the future, the global pandemic of 2020 put that doubt to rest. The question now is not "if," it is "when, where, and how?" Towards that end, across all of our architectural patterns in this document we provide a "when to move" consideration. We call out the key indicators of a current state hybrid design and discuss when using this hybrid solution is in fact more pain than it's worth. We also call out the final state of that design, helping customers understand which cloud technologies to leverage in the long run.

We will apply these tenets across the entire machine learning lifecycle, introducing hybrid patterns for ML development, data preparation, training, model deployment, and ongoing management. For each pattern we provide a preliminary reference architecture in addition to both the positive benefits, or pros, and the negative detriments, or cons. We'll discuss services such as Amazon SageMaker, Amazon Elastic Container Service, AWS Lambda, AWS Outposts, AWS Direct Connect, AWS DataSync, and many others. Lastly, we'll explore common use cases for these patterns.

This document follows the machine learning lifecycle, from development to training and deployment. Each hybrid pattern touches a minimum of two compute environments, usually a customer's on-premises data center along with services used within the AWS Cloud. We identify a "when to move" criterion - for example, when the level of effort requirement to maintain and scale a given pattern has exceeded the value it provides.

Without a doubt, there is going to be some overlap between these sections. Patterns that center on development enable training, and patterns for training open the door to hosting. Hosting in turn requires training. You'll especially notice two very different approaches to hosting – one type of pattern trains in the cloud with the intention of hosting the model itself on-premises, while another hosts the model in the cloud to applications deployed on-premises.

Finally, a key pillar in applying these patterns is security. This paper doesn't dive in to the security aspects of these patterns, however, it's recommended that customers still consider these challenges as they build.

Are you Well-Architected?

The AWS Well-Architected Framework helps you understand the pros and cons of the decisions you make when building systems in the cloud. The six pillars of the Framework allow you to learn architectural best practices for designing and operating reliable, secure, efficient, cost-effective, and sustainable systems. Using the AWS Well-Architected Tool, available at no charge in the AWS

Management Console, you can review your workloads against these best practices by answering a set of questions for each pillar.

In the Machine Learning Lens, we focus on how to design, deploy, and architect your machine learning workloads in the AWS Cloud. This lens adds to the best practices described in the Well-Architected Framework.

In the Games Industry Lens, we focus on designing, architecting, and deploying your games workloads on AWS.

In the Streaming Media Lens, we focus on the best practices for architecting and improving your streaming media workloads on AWS.

For more expert guidance and best practices for your cloud architecture—reference architecture deployments, diagrams, and whitepapers—refer to the AWS Architecture Center.

What is hybrid?

At AWS, we look at *hybrid* capabilities as those that touch the cloud in some capacity, while also touching local compute resources. Those local compute resources can be laptops hosting Jupyter notebooks and Python scripts, HDFS clusters storing terabytes of data, web applications serving millions of users worldwide, AWS Outposts stored on-premises, or countless other applications.

Whether customers are building architectures to meet their current needs, or are designing for future growth, we want to equip them with the best solutions to make their hybrid experience as seamless as possible.

We look at hybrid architectures as having a minimum of two compute environments, what we will call “primary” and “secondary” environments. Generally speaking, we see the primary environment as where the workload begins, and the secondary environment is where the workload ends.

Depending on your use case, however, the importance of these environments and their designations, will vary. If you are packaging up a model locally to deploy to the cloud, you might call your local laptop “primary” and your cloud environment “secondary.”

Conversely, if you are training in the cloud with the intention of deploying locally, you might call your cloud environment “primary” and your local environment “secondary.”

Here we are simply highlighting that a hybrid workload is one that uses two compute environments. Whether you consider one or the other a primary or secondary environment is up to you.

What hybrid is not

While it is entirely possible that either your primary or secondary environment is, in fact, another cloud provider, we prefer not to explicitly discuss those patterns here. That’s because the technical details and capabilities around specific services, features, and global deployment options are not necessarily the same across all cloud providers, and we consider that type of analysis outside the scope for this document.

There are some container-specific tools that provide a “run anywhere” experience, such as Amazon EKS and Amazon ECS. In those contexts, we will lean into prescriptive guidance for building, training, and deploying machine learning models with these services.

Hybrid patterns for development

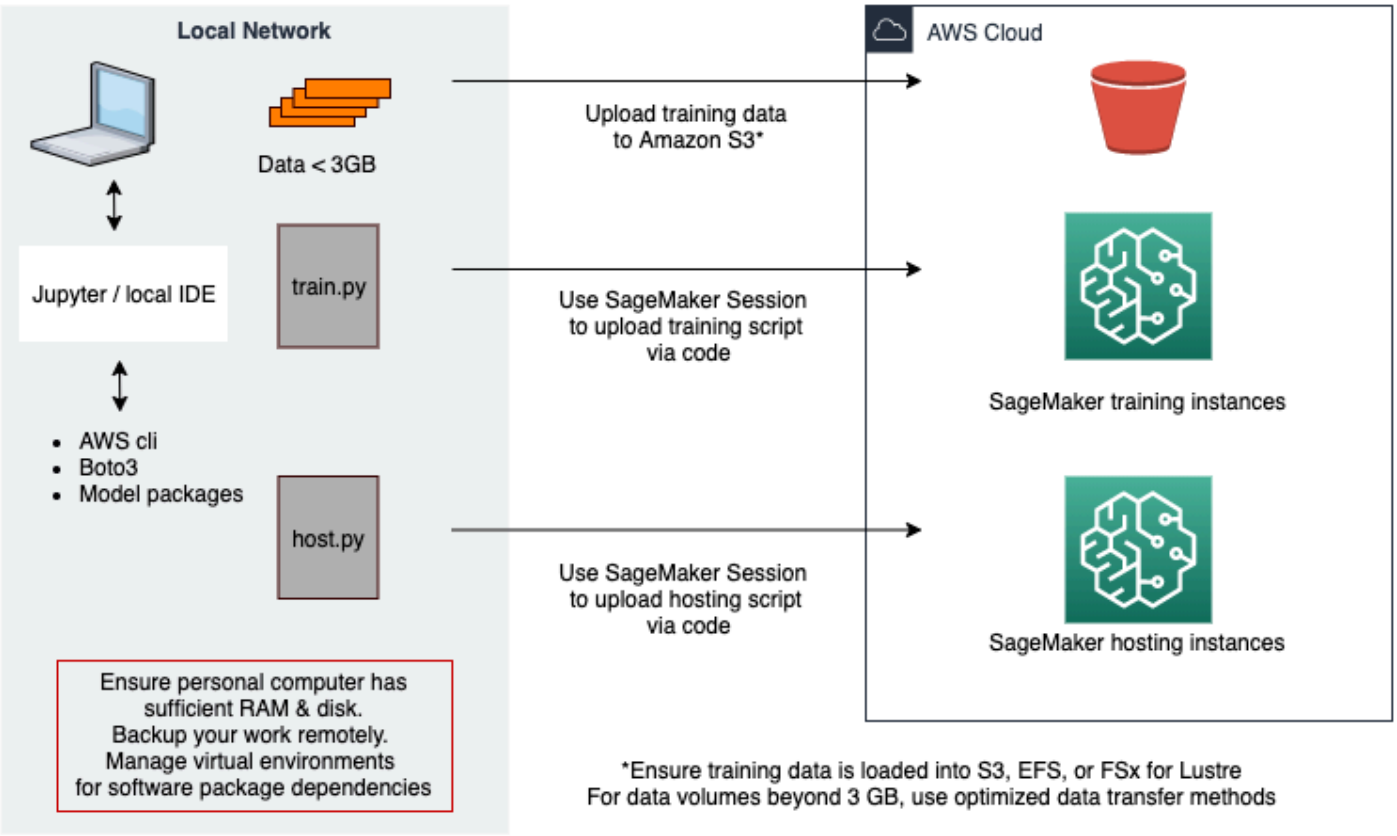
Development refers to the phase in machine learning when customers are iteratively building models. This may or may not include exploratory data analysis, deep learning model development and compilation, software package installation and management, Jupyter kernels, visualization, Docker image building, and Python-driven data manipulation.

For patterns around larger-scale data transformation or coding-free data manipulation, refer to the following section Hybrid patterns for data labelling and preparation.

Generally speaking, there are two major options for hybrid development: (1) laptop and desktop personal computers, or (2) self-managed local servers utilizing specialized GPUs, colocations, self-managed racks, or corporate data centers. Customers can develop in one or both of these compute environments, and below we'll describe hybrid patterns for model deployment using both of these.

Develop on personal computers, to train and host in the cloud

Customers can use local development environments, such as PyCharm or Jupyter installations on their laptops or personal computer, and then connect to the cloud via AWS Identity and Access Management (IAM) permissions and interface with AWS service API's through the AWS CLI or an AWS SDK (ex boto3). Having connected to the cloud, customers can execute training jobs and/or deploy resources.

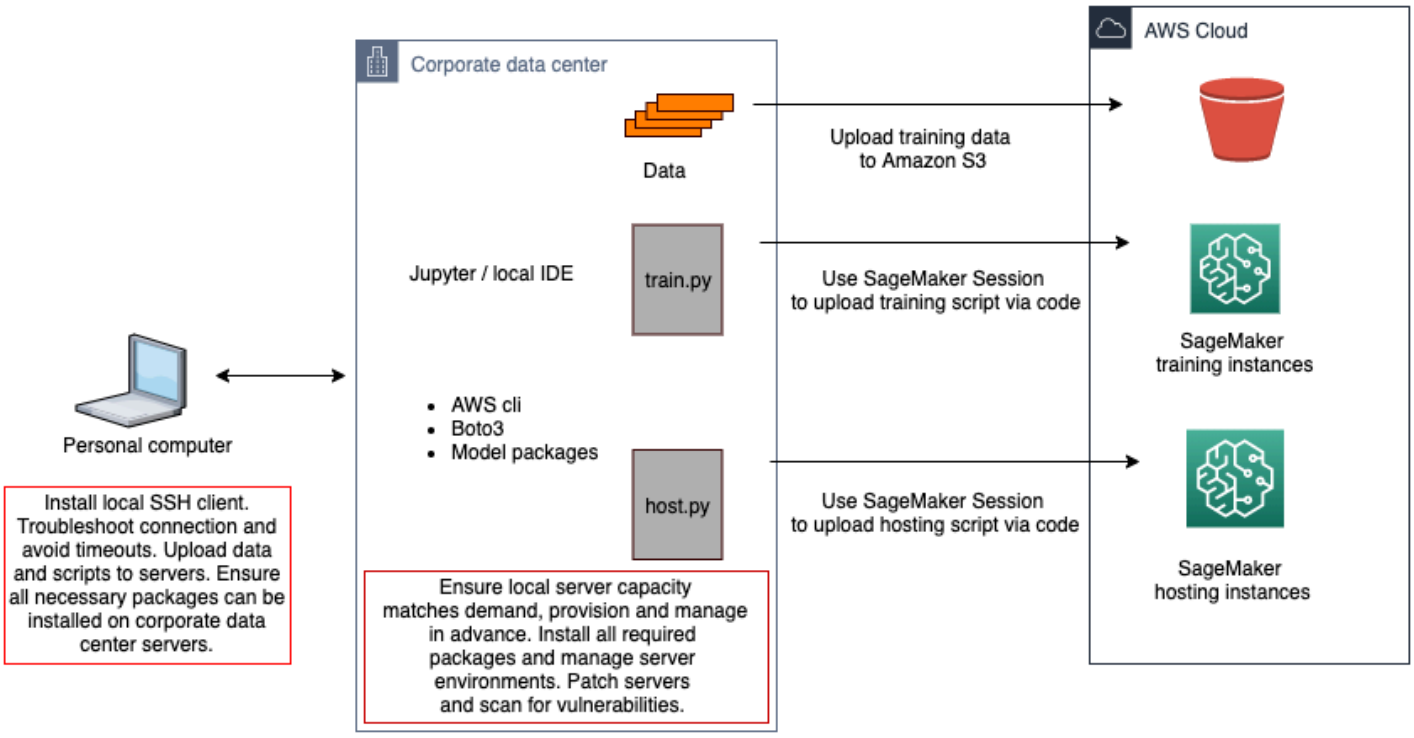


Develop on personal computers, to train and host in the cloud

Advantages	You have full control of your IDE in this scenario. You just have to open up your computer to get started. You can easily manage what's sitting in your S3 bucket, vs what's running on your local laptop. You iteratively write a few lines of code in your complex models, you check them locally, and you only land in the cloud to scale / track / deploy. This is ideal for frugal super users who thrive on managing virtual environments and software installation versions (more like a

	Linux systems admin than the average data scientist).
Disadvantages	Inability to scale beyond the compute resources of your laptop. That refers to dataset size, software versions and packages, model size, and number of experiments you are running. Lack of access to GUI-centr ic features like Autopilot, Data Wrangler, Pipelines. If your laptop dies and you didn't back up externally, your work is gone! Difficult y in onboarding non-super user employees can increase over time as software, OS, and hardware versions change. This onboarding difficulty gets more and more painful as time goes by, in some worst-case scenarios, it leads to highly valued employees not getting access to Python or Pandas for multiple months!
When to move	While enticing upfront, local development is actually more challenging (and expensive) at scale. This can be scale in terms of data sets, in terms of breadth / depth of experimen tation, and volume of team members. **If you find yourself spending a significant portion of your time managing local compute environments, it's time to move to the cloud.** This movement will free up your team's cycles and resources to focus on your business, not on your infrastructure. You may find your teams have the bandwidth to take on more projects as a result, can deliver your projects faster, or dive deeper into the analysis of their datasets.

Develop on local servers, to train and host in the cloud



Develop on local servers, to train and host in the cloud

Advantages

- Ability to capitalize on previously investmen ts in local compute
- Simplifies infrastructure setup required to meet some regulatory requirements, such as those for specialized industries (health, gaming, financial).
- Lower price point per compute on some compute environments, including some advanced GPU's
- Ideal for non-bursting, stable compute with high precision in the 6+ month forecast. That is to say, if you have extremely stable compute needs which you can anticipate with a high degree of certainty inside of a 6-

	month window, then on-premises compute is a good way to go.
Disadvantages	Fundamental challenge to dynamically provision compute resources with the needs of your business, leading teams to frequentl y be stuck in either over-utilization of local compute resources or under-utilization. Expensive GPU's can take months to ship, which leads to a larger total cost of ownership . New ideas and features can take longer to launch, because of extra effort to develop.
When to move	When you spend more time managing your local development than you do working on new data science projects. Or when the multiple months it takes to procure, wait for, and provision additional compute resources leaves your development teams sitting idle.

Hybrid patterns for training

Usually a hybrid pattern for training comes down to one of two paths. Either you are training locally, and you want to deploy in the cloud, or you have all of your data sitting on local resources, and you want to select from that to move in to the cloud to train. We'll cover both of those here.

Training locally, to deploy in the cloud

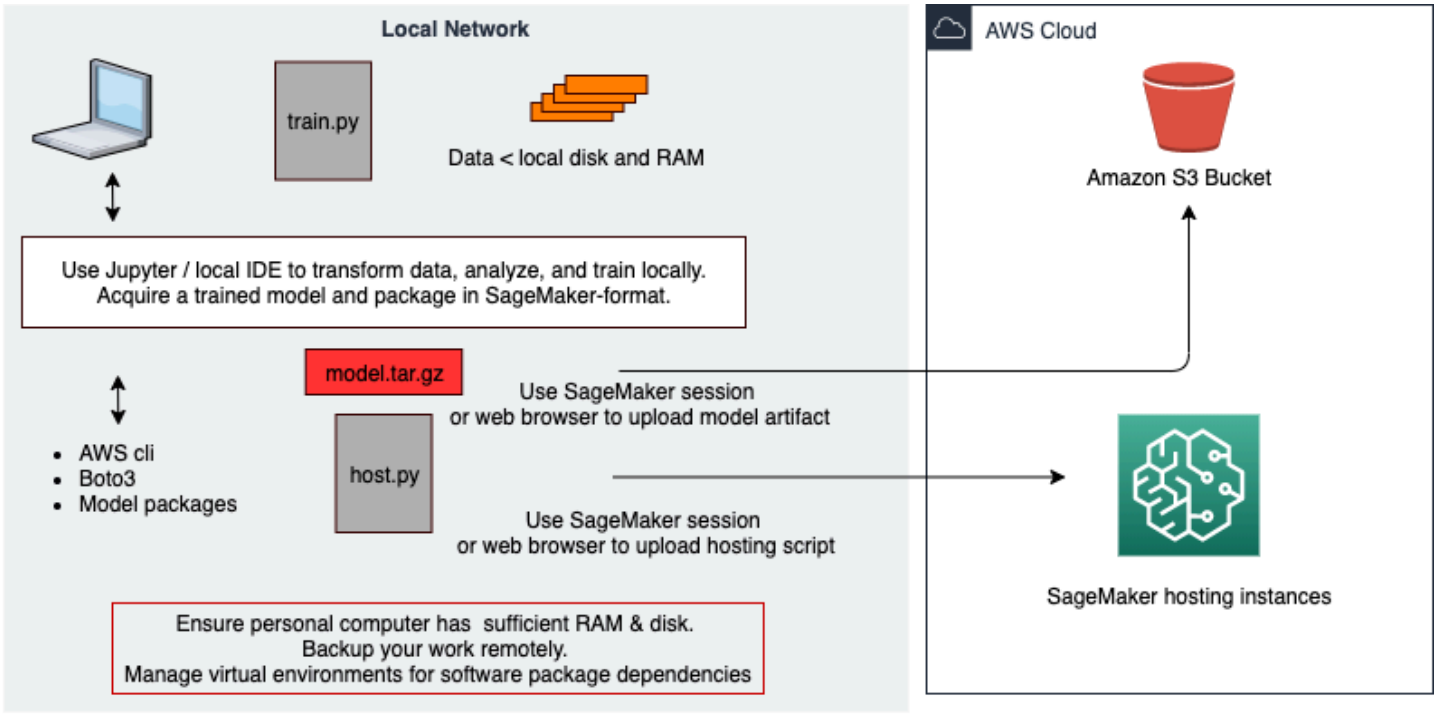
Customers may adopt a pattern to train locally when they already have significant investments on-premises, both in terms of the actual servers and the custom-built software to operationalize datasets at scale. During enterprise migrations, training locally may be advantageous as a first step to develop a model.

Generally speaking, you have two key actions here. First, if you are training locally then you will need to acquire the compute capacity to train a model. Think about the size of your datasets, and the size of the models you want to use. Ensure the hardware you are provisioning locally has enough capacity to support your datasets, both on disk and in memory. Consider the size of your data science team and their target goals. Ask yourself, do you need to support one experiment at a time? What about multiple experiments? What if you get a new customer, or a new feature idea, would you need extra local compute to support those? When you are training on-premises, you need to plan for that well in advance and acquire the compute resources ahead of time.

After your model is trained, there are two common approaches for packaging and hosting it in the cloud. One simple path forward is Docker – using a Docker file you can build your own custom image that hosts your inference script, model artifact, and packages. Register this image in the Amazon Elastic Container Registry (Amazon ECR), and point to it from your SageMaker estimator.

Another option is using the pre-built containers within the SageMaker Python SDK, also known as the deep learning containers (or DL AMIs). Bring your inference script and custom packages, upload your model artifact to Amazon S3, and import an estimator for your framework of choice. Define the version of the framework you need in the estimator, or install it directly with a *requirements.txt* file or a custom bash script.

In the following diagram, we outline how to do this from your laptop. The pattern is similar for doing the same from an enterprise data center with servers, as outlined above.



Text

Using pre-built containers

How to monitor your model in the cloud

After deploying a SageMaker resource for hosting, make sure to take advantage of pre-built features! A key feature for hosting is **model monitor**, or the ability to detect *data, bias, feature, and model quality drift.*

Generally speaking, this refers to the ability to capture data hitting your real-time endpoint, and programmatically compare this to your training data. If the inference data is outside of your pre-set thresholds, trigger a re-training pipeline.

Enabling model monitor is easy in SageMaker. Upload your training data to an Amazon S3 bucket, and use our pre-built image to learn the upper and lower bounds on your training data. This job uses Amazon Deequ to perform “unit testing for data,” and you will receive a JSON file with the upper and lower statistically-recommended bounds for each feature. You can modify these thresholds.

After confirming your thresholds, schedule monitoring jobs in your production environment. These jobs run automatically, comparing your captured inference requests in Amazon S3 with your thresholds.

You will receive CloudWatch alerts when your inference data is outside of your pre-determined thresholds, and you can use those alerts to trigger a re-training pipeline.

How to handle retraining / retuning

Once you already have a model hosted in the cloud, it's compelling to consider the cloud for a retrain and retune pipeline. This is because you can easily run a retraining and retuning job in the cloud without the overhead of provision, scheduling, and managing your physical resources around this job.

SageMaker makes train and tuning jobs easy to manage, because all you need to bring is your training script and dataset. Follow best practices for training on SageMaker, ensuring your new dataset is loaded into an Amazon S3 bucket, or other supported data source.

Once you have defined a training estimator, it is trivial to extend this to support hyperparameter tuning. Extend your training job by explicitly accepting hyperparameters in the estimator, and ensure your training script emits an objective metric. Define your tuning job configuration using tuning best practices, and execute.

Having defined a tuning job, you can automate this in a variety of ways. While AWS Lambda may seem compelling upfront, in order to use the SageMaker Python SDK (and not boto3) with Lambda, sadly you need to create an executable layer to upload within your function.

A more compelling option may be to consider **SageMaker Pipelines**, an MLOps framework that uses your SageMaker Python SDK job constructs as argument and creates a step-driven framework to execute your entire pipeline.

On the plus side, you can also add bias detection to your SageMaker Pipeline, providing greater granularity around the fairness of your model and datasets, enabling more compliance in your workflow.

How to serve thousands of models in the cloud at low cost

Another key feature of hosting models in SageMaker is **multi-model endpoints.** Multi-model endpoints give you the ability to serve thousands of models from a single endpoint, invoking the name of the model when calling predict.

Define your inference script, ensuring the framework is supported by SageMaker multi-model endpoints or by building your own image.

Create the multi-model endpoint, pointing to Amazon S3, and load your model artifacts into the bucket. Invoke the endpoint from your client application, (such as with AWS Lambda), and dynamically select the right model in your application.

This is in addition to **multi-container endpoints**, allowing you to host up to 5 containers on a single SageMaker endpoint, invoking the endpoint with the name of the model you want to use.

Advantages	A perceived upside to this approach is the potential to have more control over your training environment. This is particularly appealing to those who work in highly regulated industries, which may have legal constraints around the physicality of data residence. However, in the long haul, we firmly believe that the cloud not only provides greater flexibility, but *can actually increase firm's security posture,* through freeing up resources from physical security, patching, and procurement.
Disadvantages	• Not taking advantage of cost savings on spot instances • Not using pre-built Docker images, but potentially wasting engineering effort developing these from scratch • Not using advanced distributed training toolkits • Not using custom hardware like upcoming Trainium • Not using prebuilt tuning packages, but need to build or buy your own tuning package

	• Not using debugger, profiler, feature store, and other training benefits
When to move	• When the cost of developing your own local training platform exceeds its use, particularly as the difference in features grows overtime between what your time can provide vs other solutions available on the market • Also when the time to provision additional compute resources is far outstripped by the demand for training by your data science resources or business needs, e.g. by retrainin g every day or by training thousands of models.

Storing data locally, to train and deploy in the cloud

The biggest questions here are when and how do I move my data to the cloud?

Schedule data transfer jobs with AWS DataSync

AWS DataSync is a data transfer service that simplifies, automates, and accelerates moving data between on-premises storage systems and AWS storage services, as well as between AWS storage services. Using AWS DataSync you can easily move petabytes of data from your local on-premises servers up to the AWS Cloud.

Using either the public internet or private connections, you deploy an agent onto your local resources and schedule data transfer tasks.

AWS DataSync connects to your local NFS resources, looks for any changes, and handles populating your cloud environment. You can deploy directly into Amazon S3 buckets or EFS volumes, both of which support training in SageMaker.

Migrating from Local HDFS

As customers explore migrating data stored in local HDFS clusters, typically they find themselves somewhere between two extremes. On the one hand, you might completely abandon your HDFS-centric workflow and opt for cloud-native solutions.

On the other, you might wholly embrace HDFS as your center and move towards hosting it within a managed service, Amazon Elastic Map Reduce (EMR). Most customers will be somewhere in between these two, using EMR for some Spark-centric data transformations at scale, while using Amazon S3 for longer term data storage.

If you are interested in learning how to migrate from local HDFS clusters to Amazon EMR, refer to the EMR Migration guide.

Best practices

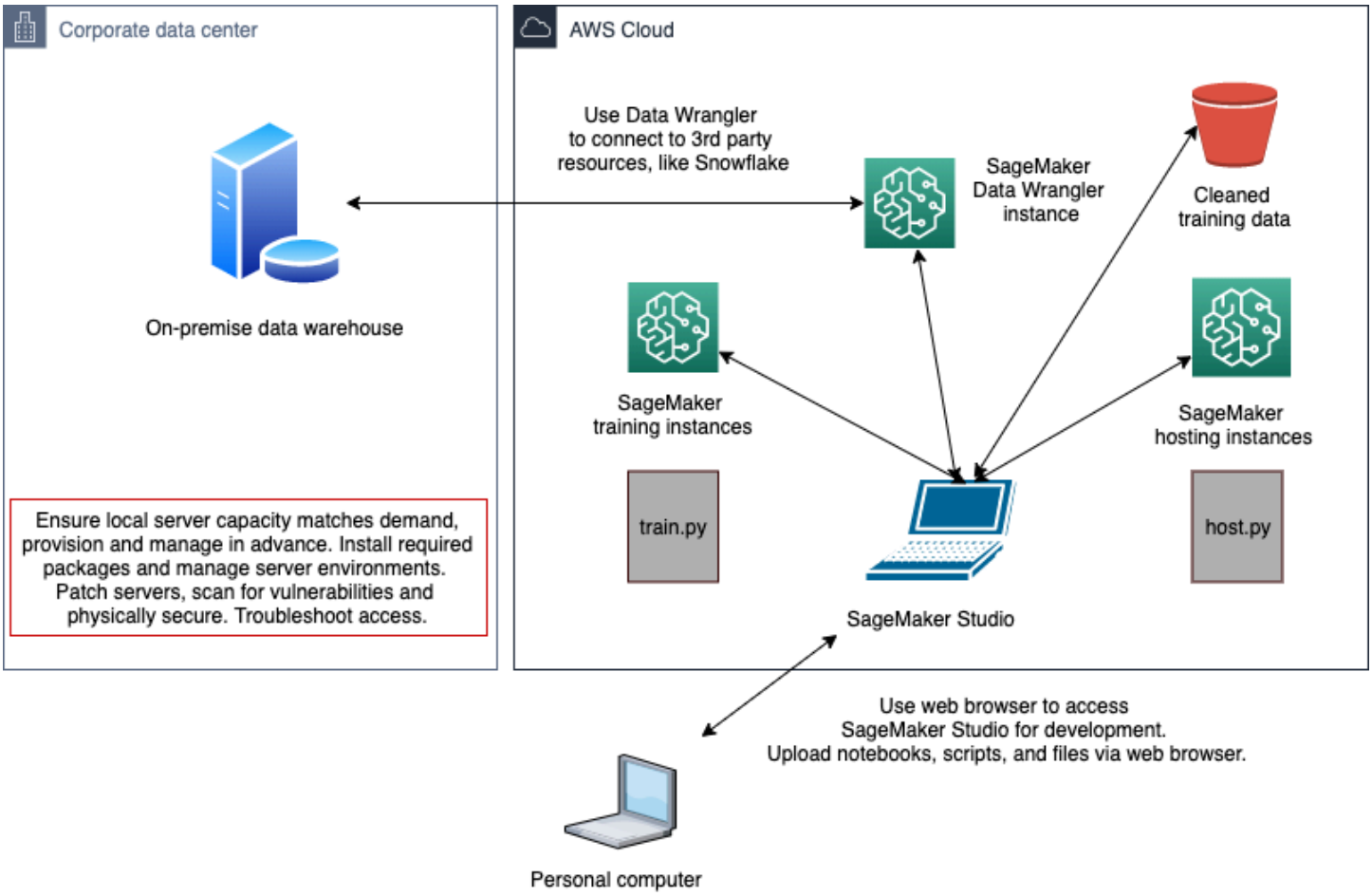
- Use Amazon S3 intelligent tiering for objects over 128 KB
- Use multiple AWS accounts, and connect them with Organizations
- Set billing alerts
- Enable SSO with your current Active Directory provider
- Turn on Studio

Advantages	• This is a fast way to realize the value of your locally-stored datasets, particularly during a cloud migration • Training and developing in the cloud gives you access to fully-managed features within Amazon SageMaker and the entire AWS Cloud • You can offload your on-premises resources more easily by leveraging capabilities in the cloud. This frees up your teams from procuring, provisioning, securing, and patching local compute resources, enabling them to dynamically scale these with the needs of your business.

	• Generally speaking, you can *deploy more models more quickly* by training and hosting in the cloud.
Disadvantages	• Expending more resources storing data locally than potentially necessary • Remember that locally-managed servers are well known for failing often. If you intend to train your ML models locally, you should anticipate a high volume of node drop-outs in your data centers. This is because ML models can be very volatile and burst. One large job can easily consume 1TB of RAM while another can require smaller memory, but execute for potentially days. On top of this, your data science team is not going to know at the beginning of the year how many experiments they intend to run through the full year. This is because the ML field is constantly influx, using inspiration from other scientific disciplines and solving complex, rapidly evolving use cases. If your team gets a great idea that could move your business forward, you don't want to tell them to hold off on developing it until you can provision more servers. • Cost mitigation can be important here. Customers should be aware of any data duplication across environments, and take action to reduce costs aggressively.
When to move	When the cost of managing, securing, and storing your data on-premises exceeds the cost of archiving and storing it in the cloud

Develop in the cloud while connecting to data hosted on-premises

Customers who see the value of outsourcing management and upkeep of their enterprise ML development platforms, such as through using managed services like Amazon SageMaker, can still connect in to their on-premises data store at the beginning and middle phases of their enterprise migration.



Developing in the cloud

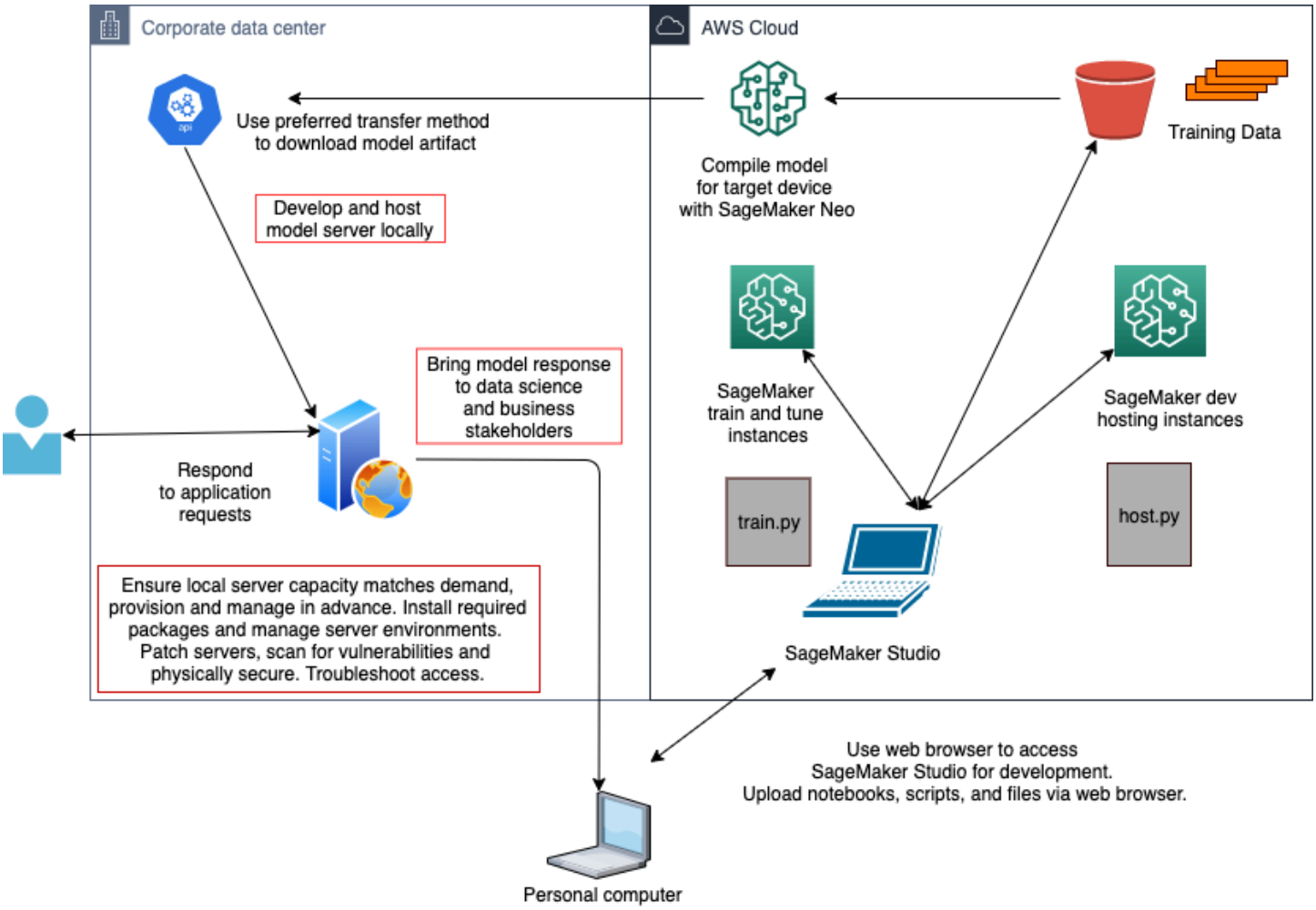
Data Wrangler and Snowflake

In this pattern we demonstrate using SageMaker Studio's fully managed development experience, in particular SageMaker Data Wrangler**.** Data Wrangler enables everyday analysts and data scientists with 300+ built-in transformations and a fully-featured UI to transform their datasets for machine learning.

Data Wrangler enables customers to browse and access data stores across Amazon S3, Amazon Athena, Amazon Redshift, and third-party data warehouses like Snowflake. This hybrid ML patten provides customers the ability to develop in the cloud while accessing data stored on premises, as organizations develop their migration plans.

Train in the cloud, to deploy ML models on-premises

Remember, you can download whatever type of model artifact you need, but if you are deploying on-premises, *you need to develop and host your own local web server.* We **strongly recommend you decouple* hosting your model artifact from your application.**



Training in the cloud

This scenario builds on your previous experience developing and training in the cloud, with the key difference of exporting your model artifact to deploy locally. We recommend using development

and/or test endpoints in the cloud to give your teams the maximum potential to develop the best models they can.

Note that Amazon SageMaker *lets you specify any type of model framework, version, or output artifact you need to*. SageMaker does not have an opinion on what type of model you should or should not use, we simply make it easy to develop, train, tune, and deploy them all.

That being said, you'll find all model artifacts wrapped as *tar.gz* archives after training jobs, as this compressed file format saves on job time and data costs.

If you are using a managed deep learning container, also known as *script mode*, for training and tuning, but you still want to deploy that model locally, plan on building your own image with your preferred software version, scanning, maintaining, and patching this over time. If you are using your own image, you will need to own updating that image as the software version, such as TensorFlow or PyTorch, undergoes potentially major changes over time.

Lastly, keep in mind that it is an unequivocal best practice **to decouple hosting your ML model from hosting your application.**

Once you use dedicated resources to host your ML model, specifically ones that are separated from your application, this greatly simplifies your process to push better models. This is a key step in your innovation flywheel.

Based on your customer's demands and research improvement within machine learning, you need the ability to develop, train, and host better and better ML models. If these two microservices are tightly coupled, *you are cutting yourself off from the potential upside of a more performant model.*

In addition to improving your model performance with updated research trends, you need the ability to redeploy a model with updated data.

The global coronavirus pandemic has only further demonstrated the reality that markets are changing all of the time – you need your ML model to stay up to date with the latest trends in your customer base. The only way you can deliver on that requirement is to retrain your model with updated data, and redeploy this.

Each business application will require a slightly different retrain and retune process, balancing both the cost of the job with the benefits of higher accuracy. Whatever pace you set for your team around model updates, make sure you the process to redeploy these is in the order of hours-to-days, not weeks-to-months.

As models grow and shrink in size, hitting potentially billions of parameters and hundreds of gigs in byte size, or shrinking down to hundreds of parameters and staying under a few MB in size, you want the elasticity of the cloud to seamlessly map the state-of-the-art model to an efficient hardware choice.

Advantages	• Can use SageMaker Neo to compile your model for a target device • Feels like you have more control upfront • Taking advantage of cloud-based training and tuning features, like spot, debugger, model and data parallel, Bayesian optimizat ion, etc. • Can enable your organization to progress on their cloud migration plan while the application development moves to the cloud
Disadvantages	• Need to develop, manage, maintain, and respond to operational issues with locally managed webservers • Own the burden of building and maintaini ng up-to-date versions of model software frameworks, such as TensorFlow or PyTorch. • Bigger risk of tightly coupling compute for your model with compute for your applicati on, making it more challenging for you to deploy new models and new features to your application over time • Need to develop your own data-drift detection and model monitor capabilities • Not taking advantage of cloud-based features for global deployment, see next section for more details • Need to develop your own application monitoring pipeline that extracts key

	metrics, business details, and model responses, to share with business and data science stakeholders
When to move	• When your ability to deploy new applicati ons on-premises is hindered by your need to procure, provision, and manage local infrastructure • When your model is loses accuracy and/or performance over time, due to your inability to quickly retrain and redeploy • When the cost of monitoring, updating, maintaining, and troubleshooting

Monitor ML models deployed on-premises with SageMaker Edge Manager

Customers can train ML models in the cloud, deploy these on-premises, and monitor and update them in the cloud using SageMaker Edge Manager. SageMaker Edge Manage makes it easy for customers to manage ML models deployed on Windows, Linux, or ARM-based compute environments. Install the edge manager agent onto the CPU of your intended device, and leverage AWS IOT Core or another transfer method to download the model to device, and execute local inferencing.

While customers do still need to provision, manage, procure and physically secure the local compute environments in this pattern, Edge Manage simplifies the monitoring and updating of these models by bringing the control plane up to the cloud.

You can bring your own monitoring algorithm to the service and trigger retraining pipelines as necessary, using the service the redeploy that model back down to the local device.

This is particularly common for technology companies developing models for personal computers, such as laptops and desktops. Compute designers can deploy ML models directly onto the device, using SageMaker Edge Manager, to improve the experience of unique customers based on their behavior and profile.

Hybrid patterns for deployment

In the previous section we discussed hybrid patterns for training, with the intention of deploying the model itself either in the cloud or on-premises. In this pattern we focus mostly on hosting the model in the cloud, but interacting with applications that may be hosted on-premises.

Hybrid ML patterns around deployment can be *really* interesting and complex. These are common among those technology companies who may have built their entire platform on-premises, only to realize 10+ years into the game that investing in server management is simply not a wise business decision for the overwhelming majority of companies out there.

Choosing the “best” local deployment option has a lot of variety. You want to think about where your customers sit geographically, then you want to get your solution as close to them as you can. You want to balance speed with cost, cutting-edge solutions with ease of deployment and managing.

We’d also like to call out that the speed of different teams is going to vary. We see this pattern as a plus for data science teams who, for whatever reason, may be ready and able to move to the cloud in the short-term. Hosting in the cloud to applications on-premises can enables the data scientists, while the application hosting team separately considers when, where, and how to move the rest of the application up to the cloud.

This section shows an architecture for hosting an ML model via SageMaker in an AWS Region, serving responses to requests from applications hosted on-premises. After that we’ll look at additional patterns for hosting ML models via Lambda at the Edge, Outposts, Local Zones, and Wavelength.

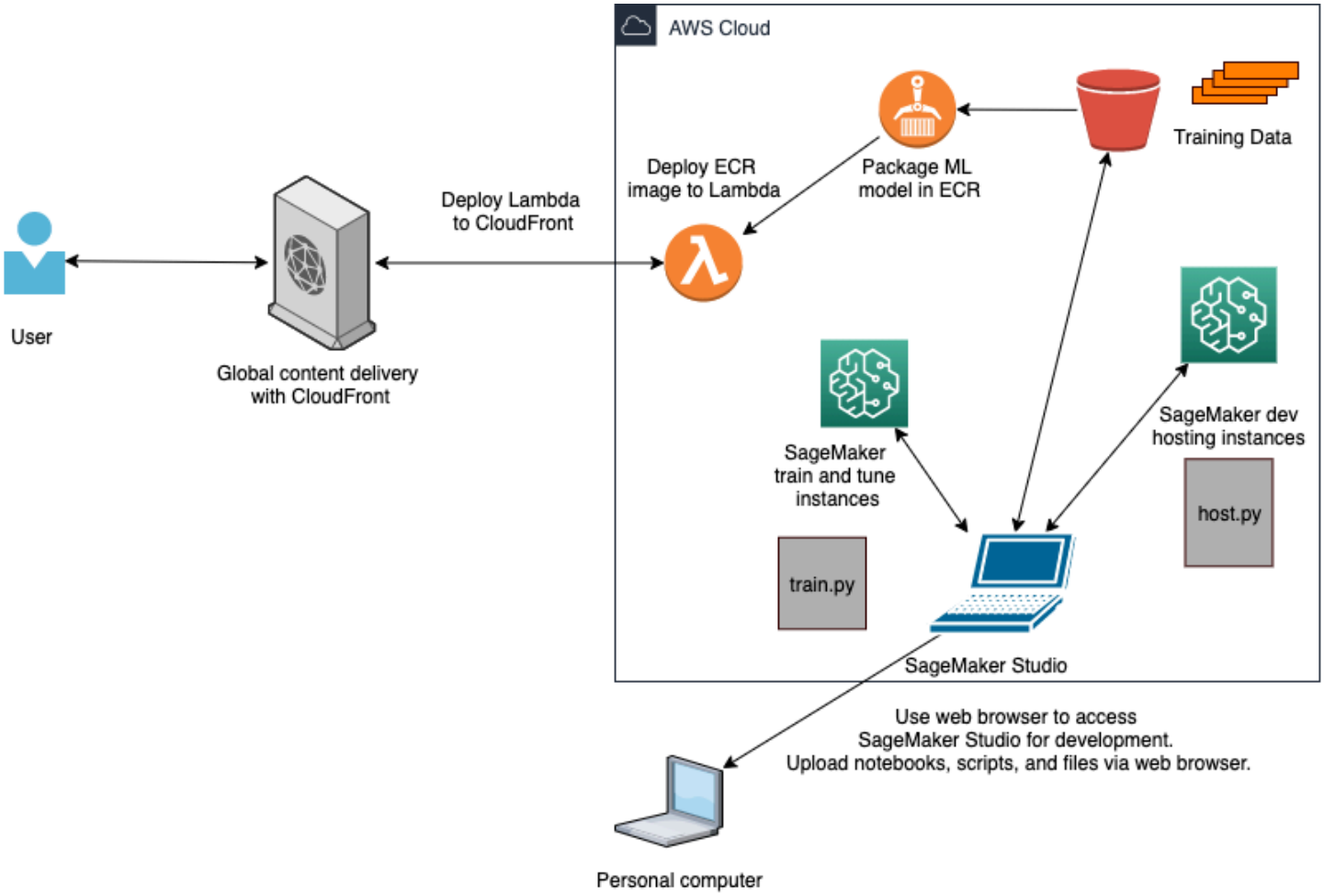
Serve models in the cloud to applications hosted on-premises

The most common use case for a hybrid pattern like this is enterprise migrations. You might have a data science team with tens of models, if not more than one hundred, ready to deploy via the cloud, while your application team is still refactoring their code to host on cloud-native services. Using this approach, you can deploy the newest models to your end users right away, while your application team takes the time it needs to reimagine their future on the cloud.

Advantages	• Are deploying ML models to application consumers

	• Can use custom hardware for ultra-low response times with AWS Inferentia • Can serve thousands of models on the cheap with multi-model endpoints • Can deploy complex feature transforms with inference pipelines • Can use built-in autoscaling and model monitor • Can easily develop retrain and tune pipelines
Disadvantages	Risk of your local application infrastructure maintenance hindering the speed of your model development
When to move	When your ability to deploy new applicati ons on-premises is hindered by your need to procure, provision, and manage local infrastru cture

Host ML Models with Lambda at Edge to applications on-premises



Hosting ML modes with Lambda at Edge

This pattern takes advantage of a key capability of the AWS global network – the content delivery network known as Amazon CloudFront**.** On top of over 80 Regions, each with multiple Availability Zones, around the world, customers can also deliver content to consumers via over 230 "points of presence" available through Amazon CloudFront. Deploying content to CloudFront is easy, customers can package up code via AWS Lambda and set it to trigger from their CloudFront distribution.

What's elegant about this approach is that *CloudFront manages which of the 230+ points of presence will execute your function.*

Once you've set your Lambda function to trigger off of CloudFront, you're actually telling the service to replicate that function across all available regions and points of presence. This can take up to 8 minutes to replicate and become available.

This means that you can physically place the content you want to deliver to customers closest to where they are geographically. This is a huge value-add for global companies looking at improving their digital customer experience worldwide.

Advantages	• Can use CloudFront, opening you up to serving on hundreds of points of presence around the world, and saving you from having to manage these • Works nicely with Docker images on SageMaker, because you can create from Amazon ECR
Disadvantages	• Can't use GPUs, so you many in fact introduce a bit of latency in some cases, particularly where customers may be better served by an ML model on Inferentia hosted in a nearby AWS Region. • Lambda has a hard-limit on the largest amount of memory you can allocate to a function, which is 10.24GB. For many "classical" ML models, such as XGBoost or linear regressors, 10GB is more than sufficient. However, for some more complex deep learning models, especially those in the 10s to 100s of billions of parameters, 10GB is woefully short in terms of RAM.
When to move	• When you need more advanced drift and monitoring capabilities • When you want to introduce complex feature transforms, such as with inference pipelines on SageMaker

- When you want to serve thousands of models per use case

AWS Local Zones

Local Zones are a way to extend your cloud resources to physical locations that are geographically closer to your customers. You can deploy ML models via ECS or EKS to serve inference with ultra-low latency near your customers, using AWS Local Zones. These are a lot like an AWS Region, but will be closer to your desired customer base.

AWS Wavelength

Wavelength is ideal when you are solving applications around mobile 5G devices, either anticipating network drop-offs or serving uses real-time model inference results.

Wavelength provides ultra-low latency to 5G devices, and you can deploy ML models to this service via ECS or EKS. Wavelength embeds storage and compute inside the telecom providers, which is the actual 5G network.

Training with a third-party SaaS provider to host in the cloud

There are a lot of great SaaS providers for ML out there in the market today, like H20, DataRobot, Databricks, SAS, and others.

Hosting a model in Amazon SageMaker that was trained from a third-party SAAS provider is easy. Ensure your provider allows export of proprietary software frameworks, such as with jars, bundles, images, etc. Follow steps to create a Docker file using that software framework, port into the Elastic Container Registry, and host on SageMaker.

Keep in mind that providers will have different ways of handling software, in particular images and image versions. While SageMaker provides managed versions of machine learning software, such as TensorFlow, PyTorch, and SKLearn, and exposes these within both training and hosting environments, other providers may differ in these capabilities.

Control plane patterns for hybrid ML

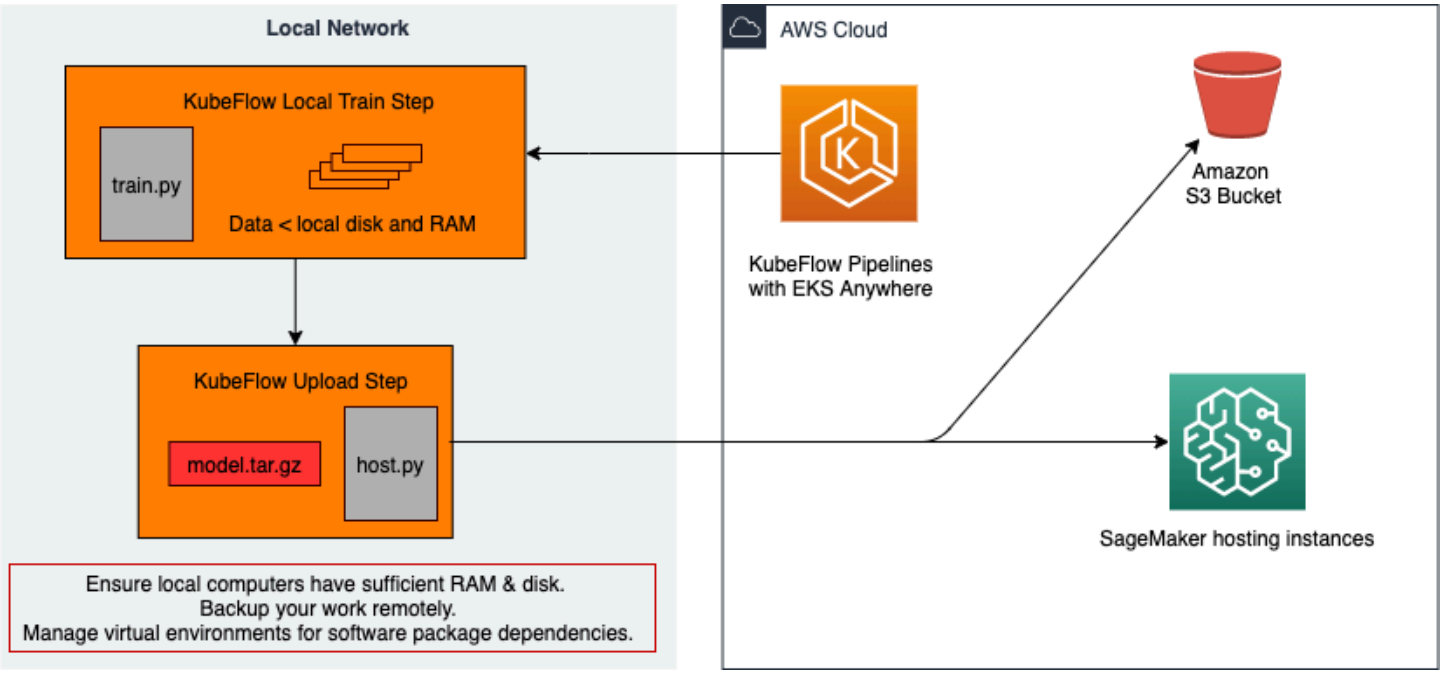
At AWS we use the concept of a "control plane," or set of features dedicated to operations and management, while keeping this distinct from the "data plane," or the datasets, containers, software, and compute environments. An especially challenging aspect of navigating hybrid ML is the need for control planes that can bridge primary and secondary compute environments, seamlessly transitioning from on-premises to the cloud and back.

To make matters even more challenging, customers' needs for operationalizing ML workloads are as varied and diverse as the businesses they exist within. Today it is not feasible for a single workflow orchestration tool to solve every problem, so most customers standardize on one workflow paradigm while keeping options open for others that may better solve given use cases. One such common control plane is Kubeflow in conjunction with EKS Anywhere. EKS Anywhere is currently in private preview, anticipated to come online in 2021.

Kubenetes is compelling for customers who want full control over their containers, especially the ability to execute these containers in a wide variety of locations. It's not, however, the only option in the game. SageMaker offers a native approach for workflow orchestration, known as SageMaker Pipelines. SageMaker Pipelines is ideal for advanced SageMaker users, especially those who are already onboarded to the IDE SageMaker Studio. In addition to sophisticated compute and machine learning needs, Studio also offers a UI to visual workflows built with SageMaker Pipelines. Apache Airflow is also a compelling option for ML workflow orchestration.

Orchestrate Hybrid ML Workloads with Kubeflow and EKS Anywhere

In this example we're demonstrating training within local on-premises resources, and orchestrating it using Kubeflow. After the model is trained, we upload into the cloud for hosting. This was used by Cisco in a hybrid ML workflow.



Training within on-premises resources and using Kubeflow

Advantages	• Flexibility. With Kubernetes you can create and operate Kubernetes in your data center and on your existing hardware using the same consistent EKS experience.
Disadvantages	• Operational overhead. In this environme nt customers need to manage, secure, and provision local compute resources. As stated earlier this is challenging to optimize with your business needs.
When to move	• When you are ready to embrace an end-to-end ML workflow based in the cloud, especially to take advantage of fully managed features such as SageMaker Studio and advanced training / hosting capabilities. • When another workflow orchestration tool seems more promising.

Additional AWS services for hybrid ML patterns

AWS Outposts

Outposts is a key way to enable hybrid experiences within your own data center. Order AWS Outposts, and Amazon will ship, install, and manage these resources for you. You can connect in to these resources however you prefer, and manage them from the cloud.

You can deploy ML models via ECS to serve inference with ultra-low latency within your data centers, using AWS Outposts. You can also use ECS for model training, integrating with SageMaker in the cloud and ECS on Outposts.

Outposts helps solve cases where customers want to build applications in countries where there is not currently an AWS Region, or for regulations that have strict data residency requirements, like online gambling and sports betting.

AWS Inferentia

A compelling reason to consider deploying your ML models in the cloud is the ease of accessing custom hardware for ML inferencing, specifically AWS Inferentia. The global pandemic has demonstrated the frailty of global supply chains, creating shortages of chips and making it harder for advanced ML practitioners to access the compute they need. Leveraging the cloud pushing the supply problem onto service providers, freeing up your teams from worrying about when their specialized devices are going to arrive. In fact, the majority of Alexa operates in a hybrid ML pattern, hosting models on AWS Inferentia and serving hundreds of millions of Alexa-enabled devices around the world. Using AWS Inferentia, Alexa was able to reduce their cost of hosting by 25%.

You can use SageMaker's managed deep learning containers to train your ML models, compile them for Inferentia with Neo, host on the cloud, and develop your retrain and tune pipeline as usual.

AWS Direct Connect

Ability to establish a private connection between your on-premises resources and your data center. Remember to establish a redundant link, as wires do go south!

Amazon ECS / EKS Anywhere

Both Amazon ECS and Amazon EKS feature *Anywhere* capabilities. This means that you can use the cloud as your control plane, to define, manage, and monitor your deployed applications, while executing tasks both in the Region and on-premises.

After installing an agent on-premises with AWS Systems Manager, in the case of ECS Anywhere, you will see your local compute resources visible in the AWS Management Console as "managed instances." Because Amazon SageMaker natively uses the Elastic Container Registry within ECS, customers can develop and/or train in the cloud, while using ECS Anywhere to deploy both in the cloud *and on-premises.* This means that customers can use ECS Anywhere to deploy their models both in the cloud and on-premises at the same point in time!

Hybrid ML Use Cases

Enterprise migrations

One of the single most common use cases for hybrid patterns is actually enterprise migrations. For some of the largest and oldest organizations on the planet, without a doubt there is going to be a difference in ability and availability in moving towards the cloud across their teams. While we believe the value of the cloud is thoroughly well proven, teams will vary in terms of how many, and how quickly they can move. Take into account that new workloads, new projects, and digital initiatives frequently start in the cloud, and it is only obvious that enterprises can and should anticipate some hybrid application development while they develop and execute their migrations.

Manufacturing

Applications within agriculture, industrial, and manufacturing are ripe opportunities for hybrid ML. After companies have invested tens, and sometimes hundreds, of thousands of dollars in advanced machinery, it is simply a matter of prudence to develop and monitor ML models to predict the health of that machinery. On top of predictive maintenance, companies can use vision-based ML models to assist productivity, reduce error rates, and improve safety of their workforce. Companies today can and do deploy ML models to cameras hosted within work and manufacturing sites to detect compliance with health policies, alert works to improperly configured equipment, and stop the roll-out of faulty products. Customers can look at solutions using Outposts, Local Zones, and Lambda at Edge to host their models in the cloud with ultralow latency, incorporating modeling results in their workflow.

Gaming

Customers who build gaming applications may see the value in adopting advanced ML services like Amazon SageMaker to raise the bar on their ML-applications, but struggle to realize this if their entire platform was build and is currently hosted on-premises. We continue to believe that gaming customers can anticipate strong performance enhancements through leveraging the AWS global delivery network to minimize end-user latency.

However, deploying a hybrid ML strategy to train and tune in the cloud while hosting on-premises as an interim step can enable data science teams to take advantage of competitive ML features while staying in touch with their current applications.

Mobile application development

With the introduction of AWS Wavelength, customers can deploy ML models *directly inside of the 5G network.* To solve applications such as anticipated network drop-off, or hosting ML models in the cloud for real-time inferencing with 5G customers, you can use ML models hosted on Amazon ECS to deploy and monitor models onto AWS Wavelength. This becomes a hybrid pattern when customers develop and train in a secondary environment, wherever that may be, with the intention to deploy onto AWS Wave

AI-enhanced media and content creation

Since the 2018 rise of transformer-based ML models that approach human-level accuracy across vision and text-based datasets, the media industry as a whole is grappling with the implications of these models and the best way to incorporate them. From using pretrained GPT-X models to write newspaper articles, poems, and short stories, to generating new images from text, to style transfer, to music generation, there are countless applications within media and entertainment for state-of-the-art vision and text models.

Customers can host these billion-plus parameter models via Amazon ECS on AWS Local Zones, responding to application requests coming from on-premises data centers, to provide world-class experiences to content creators.

Depending on where customers develop and retrain their models, using Local Zones with SOTA models may or may not be a true hybrid pattern, but used effectively it can enhance content generator's productivity and ability to create.

Autonomous Vehicles

Customers who develop autonomous machinery, vehicles, or robots in multiple capacity by default require hybrid solutions. This is because while training can happen anywhere, inference must necessarily happen at the edge. Customers in AV typically require massive training resources to deal with high resolution imagery, 3D point clouds, LiDAR data manipulation, and models with billions to trillions of parameters. These requirements make a cloud training environment attractive, where the ability to elastically provision and monitor compute resources is by nature efficiently mapped to business requirements. Inference, however, necessarily must happen at the edge. Typically, customers compile models for more efficient run-time on the device, including for specific hardware and operating system requirements. Amazon SageMaker Neo is a compelling

option here. Key questions for hybrid ML AV architectures include monitoring at the edge, retraining and retuning pipelines, in addition to efficient and automatic data labelling.

Conclusion

In this document, we explored hybrid ML patterns across the entire ML lifecycle. We looked at developing locally, while training and deploying in the cloud. We discussed patterns for training locally to deploy on the cloud, and even to host ML models in the cloud to serve applications on-premises.

At the end of the day, we want to support customer success in all shapes and forms. We firmly believe that the vast majority of workloads will end in the cloud in the long run, but because the complexity, magnitude, and length of enterprise migrations may be daunting for some of the oldest organizations in the world, we propose these hybrid ML patterns as an intermediate step on customer's cloud journey.

As always, at Amazon we listen to customer feedback and use it to iterate and improve. If you would like to engage the authors of this document for advice on your cloud migration, contact us at <hybrid-ml-support@amazon.com>.

Contributors

Contributors to this document include:

- Emily Webber, Sr Machine Learning Specialist SA, Amazon Web Services

Special thanks to the following who contributed ideas, revisions, perspectives, and customer anecdotes:

- Nav Bhasin
- Mark Roy
- David Ping
- Adam Boeglin
- Werner Goertz
- Sean Morgan
- Shelbee Eigenbrode
- Venkatesh Krishnaran
- Vin Sharma
- Sai Devulapalli
- Kevin Haas
- Phi Nguyen
- Xing Wang
- Ali Arsanjani
- Jeff Bartley

Document history

To be notified about updates to this whitepaper, subscribe to the RSS feed.

Change	Description	Date
Initial publication	Whitepaper first published.	August 15, 2021

Note

To subscribe to RSS updates, you must have an RSS plug-in enabled for the browser that you are using.

Notices

Customers are responsible for making their own independent assessment of the information in this document. This document: (a) is for informational purposes only, (b) represents current AWS product offerings and practices, which are subject to change without notice, and (c) does not create any commitments or assurances from AWS and its affiliates, suppliers or licensors. AWS products or services are provided “as is” without warranties, representations, or conditions of any kind, whether express or implied. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

AWS Glossary

For the latest AWS terminology, see the AWS glossary in the *AWS Glossary Reference*.