Audio file

Your Recording 16.wav

Transcript

00:00:06 Speaker 2

In this lesson, we're going to look at why we are learning by persona.

00:00:10 Speaker 2

So we're going to look at understanding the Sagemaker structure in terms of the collection of tools we have.

00:00:16 Speaker 2

Are key personas. We're going to zoom in in a little bit more detail on their actual activities and we're going to look at the responsibilities of each persona and that will link back a little bit more into how we're going to use Siege Maker for each of those activities.

00:00:27 Speaker 2

As we saw in the last lesson, Sagemaker AI is going to help us in performing specific tasks in our machine learning pipeline, such as preparing our data, building our model, deploying our model, and monitoring it for accuracy. But there are two paths that we can take in terms of using Sagemaker. We could take the path of using the Sagemaker SDK for Python, or we could take the path of using the Sagemaker Console user interface, part of the AWS Management Console. However, we would strongly encourage you to use the Sagemaker SDK for Python And take a code first approach as most.

00:00:57 Speaker 2

Data scientists and practitioners would. This means that we will typically be starting with a Jupyter Notebook, an interactive Python environment where we can call that Software developer kit to invoke the Sagemaker functions we need. The management console UI is just too confusing and not often used by many ML practitioners.

00:01:15 Speaker 2

Now the personas that we saw in the last lesson with the data engineer, the Mlops engineer and the data scientist, and we saw that they are targeted at different parts of the machine learning pipeline.

00:01:25 Speaker 2

We saw the data engineer. It's focused on getting the data into a usable form, which will typically mean some kind of data transformation and ingestion.

00:01:33 Speaker 2

The data scientist is then going to take that data, explore that data engineer that data so it's best suited for being used to train a model. The Mlops engineer is going to take the model produced by the data scientist and is going to ensure that we can deploy that model out into production.

00:01:47 Speaker 2

So let's zoom in on the activities of each one of those personas in a little more detail.

00:01:52 Speaker 2

Let's start with the data engineer.

00:01:54 Speaker 2

We will have identified a data source. Now that data source could be something like a relational database like SQL Server or Oracle or MySQL or Postgres something like that. Or maybe a non relational source like Dynamo DB or Redis or MongoDB something like that.

00:02:10 Speaker 2

It'll be the job of the data engineer to extract that data. Now in the case of maybe a relational database, that might be a database dump, or it might be some kind of custom query to get the data that we need out and into another form.

00:02:22 Speaker 2

And this data transformation piece is where we really need the data engineer, because the form in which data will come from the data source when we extract it might not be the form that you want to use in your model training. Let's look at data transformation in a little more detail.

00:02:36 Speaker 2

If we are extracting from a structured data source, let's say for example a MySQL database, then it is in a structured form. Now that means that to interact with it we would need to use the structured data interfaces exposed by that platform. So in MySQL that would typically be SQL query language. But what if I wanted in semi structured? I want it in tabular form like a CSV file, a comma separated value file. I'm going to need to extract that data and the data that I extract into a semi structured form like CSV.

00:03:04 Speaker 2

What am I going to bring all data? Or maybe I only need to bring out data that is relevant to the training? For example, do I bring out address information? If I'm trying to determine if a customer is going to churn from a SaaS product, is that relevant? Or do I leave the data in there for the data scientists to decide that? I'm definitely going to be concerned about regulatory compliance and ensuring that personally identifiable information does not leak from that structured source. So for example, if I'm trying to determine if customers would leave or not, I don't need to know their names. We can substitute that with an obfuscation value.

00:03:34 Speaker 2

Once that data is in a semi structured form is now available for training, I can make that available to the data scientist and our typical data source at that point would be S3. Amazon S3 is a location where we can store essentially an unlimited amount of data and we can store semi structured data in S3 very easily and easily ingestible by Sagemaker during the training process.

00:03:55 Speaker 2

Governance and privacy constraints are a huge deal for us. Imagine you're a bank and you have you're trying to determine if your customer is going to leave you and join another bank. Things like bank account numbers, personal names, date of birth, address. This is rich information that we absolutely would not want to allow to escape and be used for other purposes. So we would need to obfuscate or drop that type of data and replace it with something else, like a simple customer identifier number as a incrementing ordinal number, something like that.

00:04:22 Speaker 2

Let's think how a small organization might approach this data sourcing, extraction and transformation.

00:04:28 Speaker 2

In a small organization, the sourcing of the data, extracting the transformation of that data, that might be conducted just by a data scientist. There might only be one person responsible for data management and ultimately model training, and that's OK. In small organizations, that would be very efficient. But when we think about larger organizations, things might look a little different. In a larger organization, yes, we need to extract data from typically a structured data source and transform it and prepare it, but we need that to be a repeatable and scalable process.

00:04:57 Speaker 2

So a data engineer in a larger organization is going to be thinking not just about a one time operation, interactively querying a database and downloading a file and uploading it to S3. Instead that data engineer will be focused on using some kind of automation, which could just be Python script or it could be some data pipeline tool, could be using something like Amazon AWS Glue, something like that. And they might be building extraction and transformation pipelines so that we can feed new data from our structured data source interlocation where it can be ingested for.

00:05:27 Speaker 2

Training a model. So it's just to start thinking about data engineering and how it might look in small versus larger organizations and how the focus in a large organization will be about automating a data pipeline.

00:05:38 Speaker 2

Now a data engineer has a number of tools available to them. The data engineer may or may not be a coder. They might have Python skills or they might not. Now if they want to use a low code approach, in other words, they don't want to write code, but they want to use ready made transformations that they can simply link together, then they could make use of something called the Sagemaker Data Wrangler. And if you've ever drawn A Visio chart or some other kind of flow

chart, then that's about as complex as the data Wrangler is about linking together ready made data transformations.

00:06:08 Speaker 2

And connecting them into a sequence so that they can be run again and again. But you don't need to see any code. It's a really neat tool and we'll be taking a look at it in more detail later on.

00:06:17 Speaker 2

But if they're going to take a coding approach, which we do recommend, then we could use Sagemaker to specifically utilize Jupyterlab. Now, Jupyterlab is just a hosting environment where I can run what we call a Jupyter notebook. And that Jupyter Notebook provides me with an interactive Python shell where I can annotate what I'm doing in Markdown. So it's kind of a mix between a Markdown document and an interactive Python shell. That's great. But if you've got, let's say, a data set with 1,000,000 rows of data and maybe 50 features wide.

00:06:46 Speaker 2

And you need to transform that.

00:06:48 Speaker 2

Maybe I need to drop some columns, or maybe I need to compute data for missing values. That would be quite a computationally intensive job. And running that kind of compute intensity in your Jupyter notebook, well, that's not really what the Jupyter notebook is for. What we prefer to do is delegate that kind of large data processing into a background job where you can specify exactly the amount of compute you want to allocate to that job so that it will complete in a reasonable amount of time. So the data engineer or indeed the data scientist may choose to create a Sagemaker processing job to offload that processing and then.

00:07:18 Speaker 2

We simply find the results once it's completed.

00:07:21 Speaker 2

Now, if data engineer or data scientist is going to be working on a number of different products that relate to the same data set, then we might choose to use the feature store. Now, this just means that as we engineer our data and prepare our data so that we've got really a collection of features to be used in feature engineering, then if we need to use them again in a different project, wouldn't it be great if we could just leverage that feature store as our data source rather than going through the data transformation all over again, duplicating our effort? Now we'll talk more about the features stored later on, but right now we just need to know.

00:07:51 Speaker 2

That this is one of the available options for us when you've engineered some features for maybe 1 project, but now you want to use those same data set and same features in another project.

00:08:00 Speaker 2

And we have the concept of Sagemaker pipelines now. Sagemaker pipelines are really just an automation framework so that we can link together a number of different disparate actions. So if I wanted to extract data from a data source, transform it, and then maybe upload it into S3 ready for the data scientist, then I could create a Sagemaker pipeline that's made-up of those three steps.

00:08:20 Speaker 2

Let's now think about the Mlops engineer.

00:08:23 Speaker 2

The Enelops engineer is focused on getting the model that has been developed by the data scientist out and into production where it can deliver value by generating predictions.

00:08:33 Speaker 2

Once that model is deployed, we want to ensure that the model is producing accurate predictions. We want to know about it if we get a drop in accuracy so that we can create a feedback loop which might involve retraining the model so that it is using maybe new data and can produce more accurate results.

00:08:51 Speaker 2

So when the Mlops engineer is thinking about model accuracy, they're thinking about how they can deploy and manage measurement so we can measure the predictions that are generated and determine.

00:09:02 Speaker 2

If the accuracy is still good, we can continue.

00:09:04 Speaker 2

If we've had a drop in accuracy below a pre agreed threshold, then we can trigger the retraining of our model. Designing of these feedback loops where we detect the accuracy of the model and then generate a retraining of the model is very much what the Mlops engineer is all about.

00:09:20 Speaker 2

Now if a model has been retrained and it passes the accuracy requirements, then wouldn't it be good if that automatically deploys a new model?

00:09:28 Speaker 2

This is getting us thinking that an ML OPS engineer is all about making sure we can get models to production as quickly and as efficiently as possible, whilst going through the required steps to ensure quality and safeguards.

00:09:41 Speaker 2

Now, Mlops engineers are often compared to DevOps engineers, and when we think about what a DevOps engineer is doing, they're accelerating the code release pipeline. In traditional software

development, you will have the developer creating their code, committing it to a code repository, and that should trigger an automated pipeline, a pipeline that will check their codes formatted correctly, check their code. It has the right syntax, it doesn't have security vulnerabilities, maybe it runs some unit testing, things like that, and ultimately deploys that once the software has passed all of those checks.

00:10:10 Speaker 2

So the DevOps engineer might link together a git version repository like GitHub or GitLab and then trigger a pipeline using some kind of pipeline tool. Maybe it's Jenkins or maybe it's AWS Code Pipeline, and ultimately deploy the software artifact and if things don't go well, roll back, have a safe rollback procedure to the previous version. It's all about getting the software out there.

00:10:30 Speaker 2

The Mlops engineer is often considered similar to the DevOps engineer because it's just a shift in context that rather than deploying software, we're deploying a machine learning model. Now we're still storing our code in a git compatible version repository. We're still deploying an artifact to a compute endpoint. And how we do that deployment? Do we modify the existing deployment? Do we deploy something brand new and change over to point at the new thing while the old one is still there and a kind of blue-green deployment? Do we have safe roll back to the previous version?

00:10:59 Speaker 2

So there are absolute similarities in what the MLX engineers doing to the DevOps engineer and is absolutely similar tooling that we are going to be using to achieve that outcome.

00:11:10 Speaker 2

So we can see the DevOps engineer gets the code artifact or executable out there. With Mlops engineer gets the model out there.

00:11:16 Speaker 2

The Mlops engineer will be responsible for deploying, scaling and monitoring that model. Now, deployment as we know means getting the model that was produced by training process by the data scientist and ensuring it's deployed onto a compute platform that provides it with sufficient resource to produce predictions in an acceptable amount of time.

00:11:35 Speaker 2

If that model needs more compute resource because there is a greater workload of inference requests, then do we need auto scaling? In which case we could add more compute to that endpoint. Then again that would sit with the Mlops engineer.

00:11:47 Speaker 2

The Mlops engineer would own the CICD pipeline. Remember when I say CICD, that's continuous integration continues deployment, a term taken from the traditional software development world. But what we really are saying here is that when we have Python code that the data scientist has

developed for feature engineering and training, a model that will be committed to a git compatible repository.

00:12:07 Speaker 2

Ideally, we are looking for a trigger based on commitment to that git repository to start the process of training and registering a model with the model registry, so that you've got this automation pipeline that is being triggered based upon the actions of the scientist updating their code and committing it to Git.

00:12:25 Speaker 2

Now, once we've trained a model, we would want that pipeline to store the model artifact, typically an S3, and then register that model into the Sagemaker model registry. That gives us a way of then tracking the model artifact that was produced. So you can see those kind of two version control things going on here. There's the version control of our Python code that produced the model, and then there's the version control of the artifact, the actual model itself that was produced.

00:12:52 Speaker 2

And that model artifact will reside in the model registry.

00:12:55 Speaker 2

And then that way we can have a model approver, which may or may not be the data scientist or the Mlops engineer or could be another persona in a large organization that can approve versions of the model for production.

00:13:08 Speaker 2

And then we can start thinking, OK, well, if a model resides in the model registry and it's pending approval, who's going to change its status to being approved for production? And when they do that, that could act as a trigger to another pipeline that could actually deploy or update the model in production.

00:13:25 Speaker 2

So governance is shared across the different personas, but the Mlops engineer is going to take on a lot of key governance tasks. And depending on the size of our organization, may be another governance persona involved a model approver or model governance officer or something like that. But it's the ML OPS engineer that's going to create the structure that will allow us to have approval for use in specific environments built around the model registry approving and CICT pipeline.

00:13:53 Speaker 2

So governance has got to be enforced across the entire ML pipeline, right from the moment that we took data from our structured data source and transformed it as a data engineer to make it ready, what data was coming out of the structured source. We don't want personally identifiable information, we don't want credit card numbers or things like that as not relevant to training our model.

00:14:11 Speaker 2

We need to ensure that we have lineage. Now what I mean by the word lineage is we can understand how we got to a certain point. If you trace it back from a point of inference, we produced a prediction how What data was presented for inference? What version of the model was used? What algorithm version was used to train that version of the model? What data set was used to train that version of the model? What developer or data scientist Python code version was used to train that model?

00:14:38 Speaker 2

So if we've got all of those version control pieces of information stored, we could determine the lineage, we could determine the history of how we got to this point, and therefore we have traceability. This can lead to explainability. How did we get the results we're looking at? And we've got reproducibility. If I train a model using the same data set, same version of the algorithm, and the same version of the code, which we could absolutely do because we've tracked it, then we will get the same model artifact as a result.

00:15:04 Speaker 2

Because the Mlops engineer is building a pipeline of activities, this means it can be run again and again and again consistently.

00:15:12 Speaker 2

In those pipeline steps, we can be performing whatever compliance checks are required in our organization. There is a common phrase, AWS, that manual changes is the enemy of production. In other words, it's a risk without a reward when we make manual changes.

00:15:28 Speaker 2

Automation, on the other hand, ensures consistency in reliability, so anything we need to do more than once should be automated.

00:15:35 Speaker 2

So checks don't get skipped if they're built into a pipeline. If I commit my code to a git repository that triggers A pipeline, and that pipeline runs a check to see that I haven't left a password or an API key in my code, boom, it's called each and every time.

00:15:50 Speaker 2

We've briefly talked about monitoring and we will talk about monitoring in more detail later on, but we know that we have to have some kind of monitoring once our model hits production. Yes, we are interested in infrastructure monitoring like CPU busy and RAM utilized, but infrastructure metrics don't tell the whole picture. From a machine learning perspective, I want to know how long inference is taking. Is it taking longer than normal? I want to know if the prediction that was produced by that model is no longer above an accuracy threshold that I've determined to be for that model to remain.

00:16:20 Speaker 2

Production. I also need to be thinking about if I produce a prediction, are we able to explain how the model came to that conclusion? Is there explainability in the model or is the model a pure black box where we don't know how it operated and got to that conclusion? Generally, we are looking for explainability so that we can determine that our model does not hold any bias and is fair for all classes of input.

00:16:44 Speaker 2

In large organizations, it's likely that in addition to the machine learning operations engineer, there will be some kind of compliance officer, governance officer or model approver who will ensure that they own the approval of a model going into production. This is typical for they will want to ensure that the model artifact has that lineage, has that history, that we know how we got to that point and we can determine the data set and the data set didn't contain any personally identifiable information. The governance or compliance officer will want to ensure that that is.

00:17:14 Speaker 2

Case so this way that the model approver can determine only when they are satisfied that all the checks and balances are being met would they approve a model which would then trigger the deployment of the model into production.

00:17:27 Speaker 2

This approved reject functionality is part of the Sagemaker Model registry. By default, when we register a model into the model registry, it will go in with a status of pending approval, and it is up to whoever is granted the permission to approve or reject a model to go in and explicitly approve or reject that model.

00:17:44 Speaker 2

Remember that the compliance or governance officer will be wanting to ensure that the model does not produce any biased results. Imagine this model was going to aid a bank in determining who received a bank loan. Now, if the model was biased through some fault of the class imbalance in the original data set or some things that have gone wrong during feature engineering, we might end up with a model that could, for example, negatively be biased against one particular ethnic group.

00:18:10 Speaker 2

And that would be completely unfair and completely wrong. So we have to ensure that we've got tooling in place that's going to assess the impact of each class group that you might have and ensure that we're having an equal level of fairness across those different class groups. And we can only get that through explainability. And therefore, the monitoring that you do should really have biased reporting and explainability reporting built right in so that we can show that we are not biased against anyone group and that our model is good and fair.

00:18:39 Speaker 2

Before it hits any actual real world uses.

00:18:42 Speaker 2

So how does aligns with our Sagemaker product features?

00:18:46 Speaker 2

But we know that we are going to register a model with a model registry so that we have version controlling of the model artifact itself. Now it could be the governance officer or compliance officer that gains access to the model registry and has the permission to approve or reject a model into production. And we know that the model registry approval will ultimately be a trigger for deployment.

00:19:06 Speaker 2

Sagemaker pipelines are a construct that allow us to automate a sequence of events. For example, in training to be able to train the model and then register it into the model registry. That could be done as a 2 step pipeline in inference. Then maybe deployment of the model and enabling of monitoring could be something that inference pipeline could do. Now we'll be covering Sagemaker pipelines in great detail in a later lesson. For now, we just need to know that it's a native automation framework that allows us to do a number of things in sequence that are invoked as a single action.

00:19:35 Speaker 2

Then we have Sagemaker endpoints and we know that Sagemaker endpoints is a way of us hosting our model so that we can handle inference requests. Now at this point, it's optional. If you want to host your model in AWS, it makes sense to use Sagemaker endpoints as they are a fully managed service. But just because you train a model on Sagemaker doesn't mean you have to host it there. So just wanted to be really clear here that endpoints are only needed if you decide that you also want to use Sagemaker for hosting, so that Sagemaker does both training, development and hosting together.

00:20:05 Speaker 2

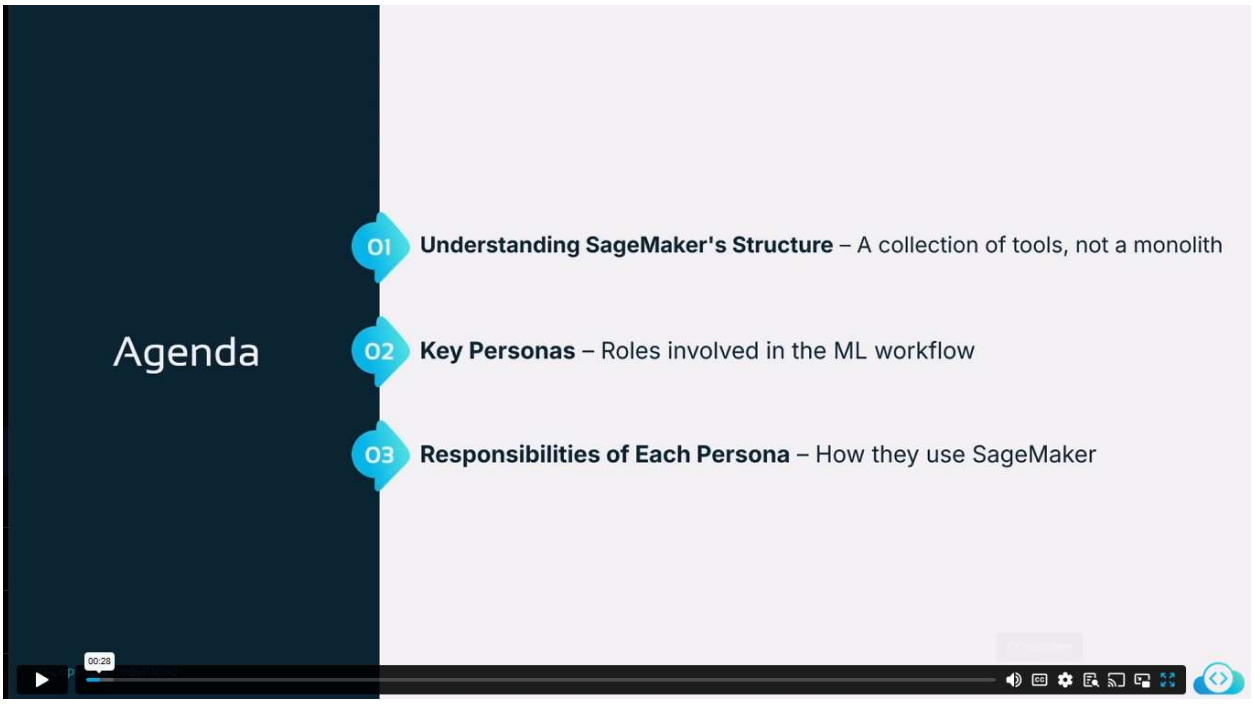
Once our model is deployed, we're going to want to be able to monitor it and we want to monitor for the quality of the data of the inference request coming in. Maybe inference requests that are coming in no longer look like the data that we trained our model on. And we want to ensure that the predictions that we produce are within an accuracy threshold that we determine.

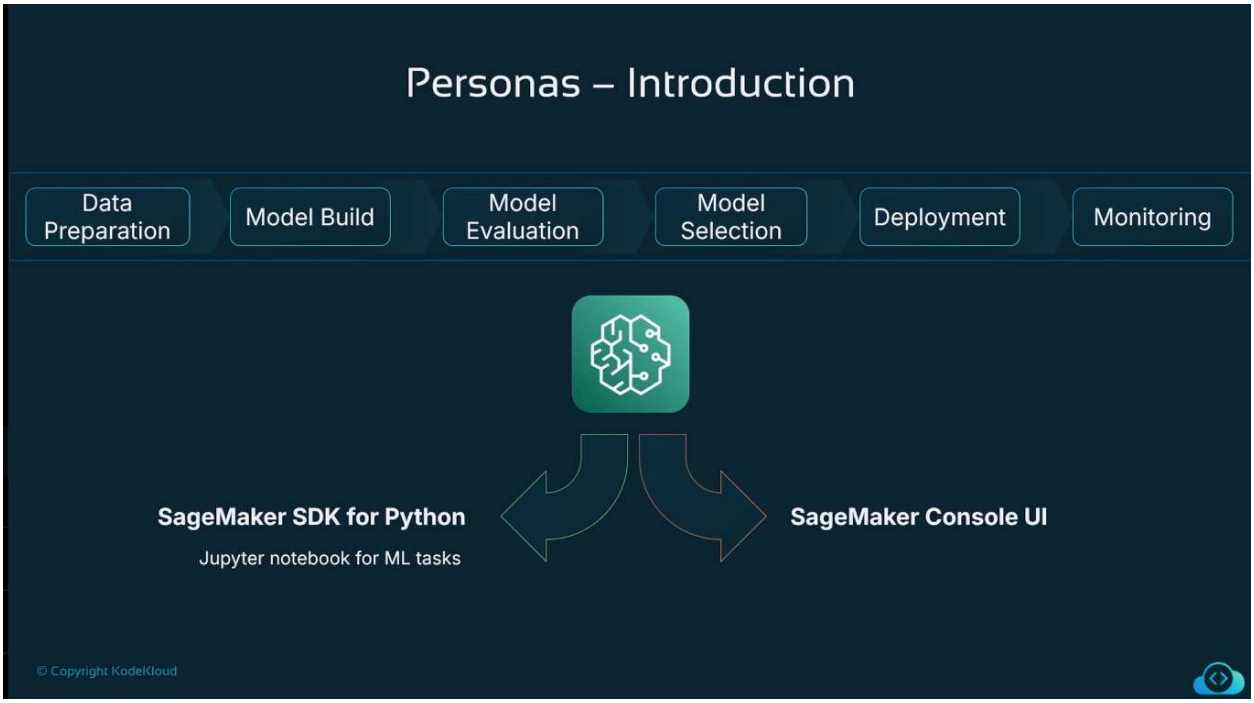
00:20:23 Speaker 2

We saw that the importance of having a model that is not biased against one particular class group that's exposing your data. Now that could be socioeconomic group, that could be an ethnicity group, whatever group is being considered as an input feature. But we have to make sure that our model is not biased against anyone particular group for outcome and it's explainable so that we can show how the model came to that particular target prediction and it was done so without a bias. Now Sagemaker Clarify will help me do that again. We'll talk more about that.

00:20:53 Speaker 2

Later on, but it's part of what we talk about with our monitoring process.


Agenda
01
Understanding SageMaker's Structure – A collection of tools, not a monolith
02
Key Personas – Roles involved in the ML workflow
03
Responsibilities of Each Persona – How they use SageMaker
00:28


Personas – Introduction
Data Preparation
Model Build
Model Evaluation
Model Selection
Deployment
Monitoring
SageMaker SDK for Python
Jupyter notebook for ML tasks
SageMaker Console UI
© Copyright KodeKloud

Personas – Introduction

Data Engineer

- Design, develop, and manage data warehouses, data lakes, and ETL/ELT pipelines
- Build and optimize data ingestion and transformation pipelines

MLOps Engineer

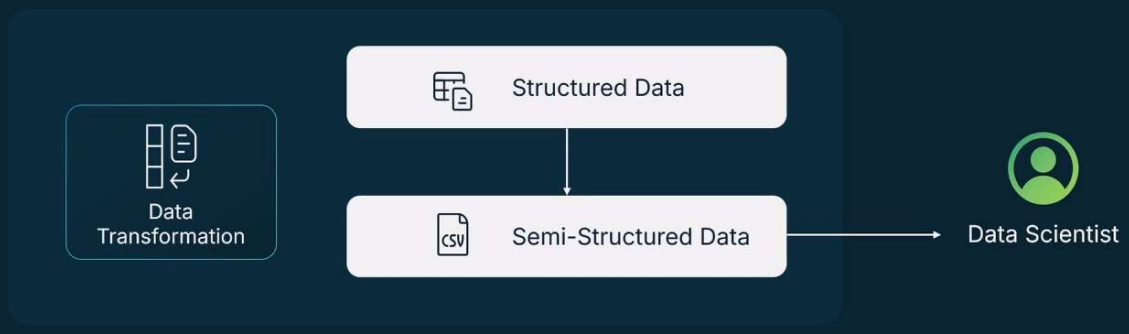
- Create ML pipeline
- CI/CD
- Model version control

Data Scientist

- Experimentation
- Feature engineering
- Training
- Inference

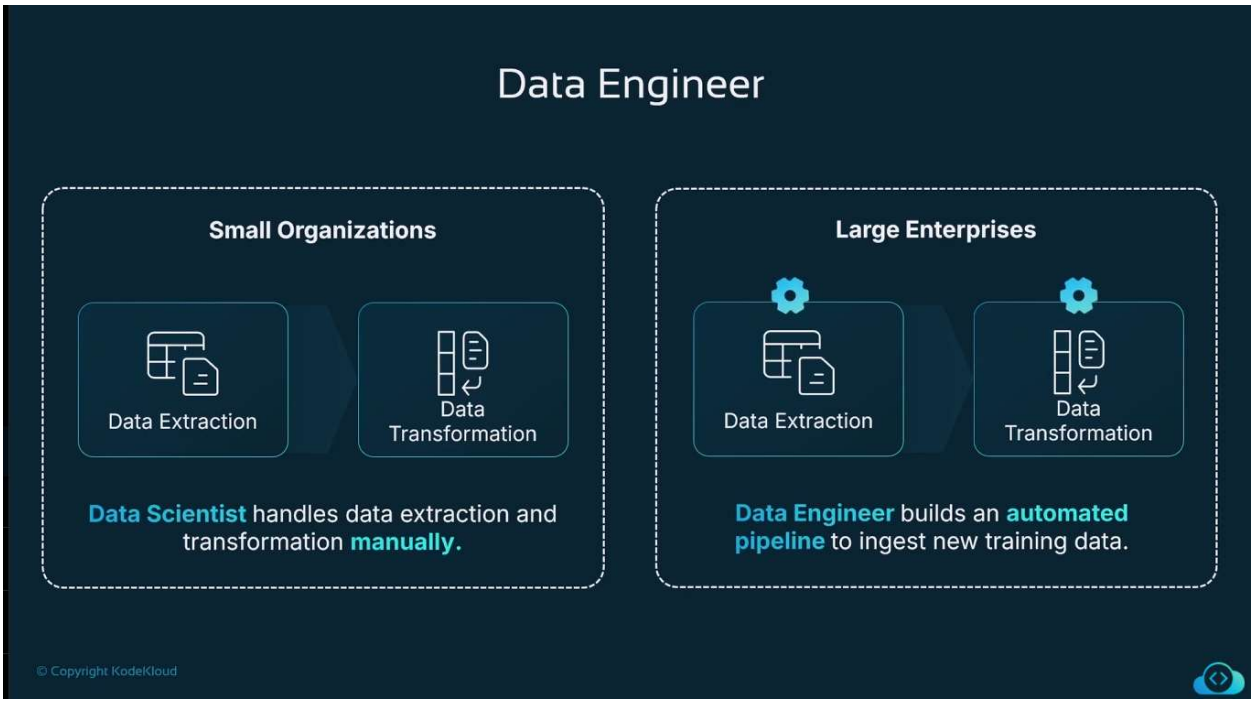
Data Engineer

Data Engineer



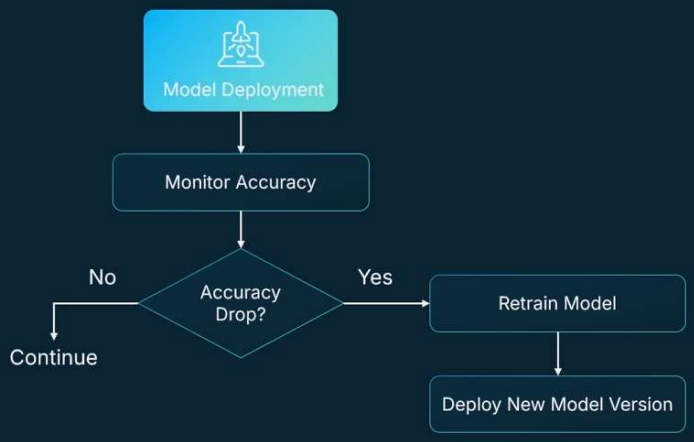
Data Engineer

Governance/Privacy constraints may require obfuscating or dropping parts of the source data.


Data Engineer
Small Organizations
Data Extraction
Data Transformation
Data Scientist handles data extraction and transformation manually.
Large Enterprises
Data Extraction
Data Transformation
Data Engineer builds an automated pipeline to ingest new training data.
© Copyright KodeKloud

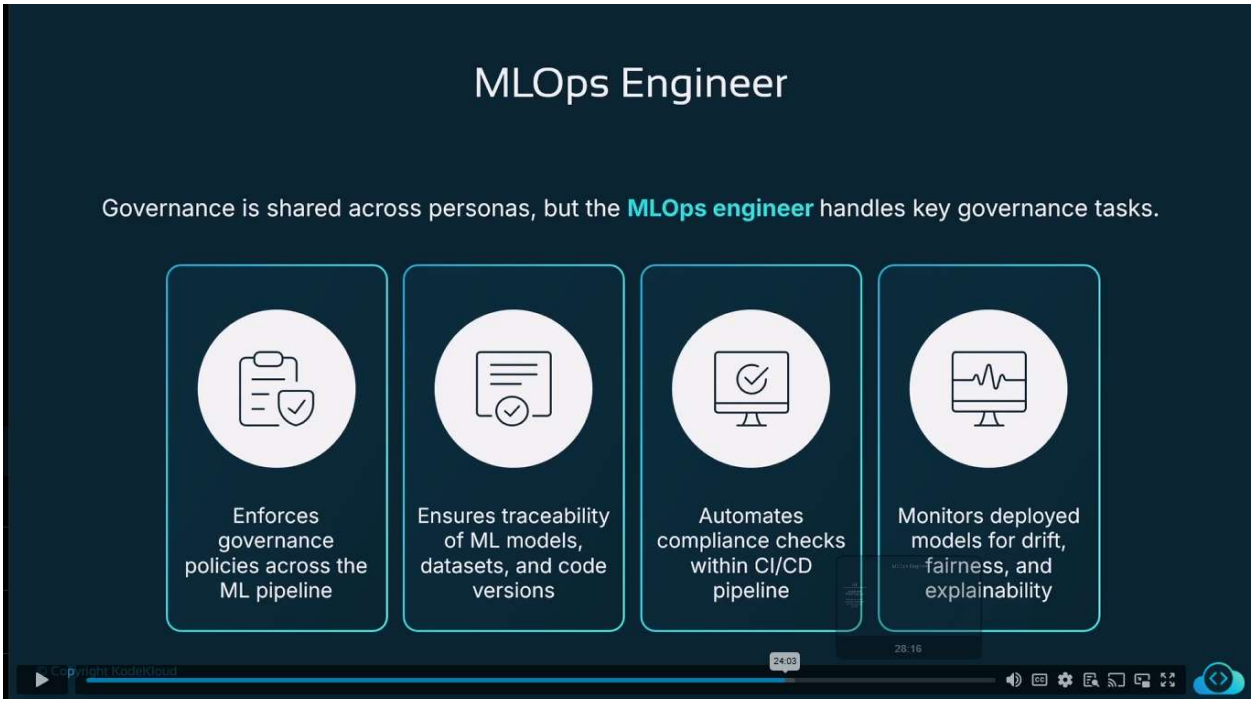

Data Engineer
01
SageMaker Data Wrangler
Simplifies data prep, cleaning, and feature engineering with a visual interface
02
SageMaker Processing
Runs preprocessing and transformation scripts with managed infrastructure
03
SageMaker Feature Store
Centralized repository for storing and sharing features across teams
04
SageMaker Pipelines
Automates the ETL process as part of an ML workflow
12:02

MLOps Engineer



MLOps Engineer


MLOps Engineer
Deploying, scaling, and monitoring ML models in production
Automating CI/CD pipelines and aligning Git repositories
Managing model versioning, governance, and seamless updates
Copyright KodeKloud


MLOps Engineer
Governance is shared across personas, but the MLOps engineer handles key governance tasks.
Enforces governance policies across the ML pipeline
Ensures traceability of ML models, datasets, and code versions
Automates compliance checks within CI/CD pipeline
Monitors deployed models for drift, fairness, and explainability
Copyright KodeKloud

MLOps Engineer

Compliance Officer oversees governance in highly regulated environments.

01

Ensures ML pipeline alignment with policies and regulations

02

Approves/Rejects models for deployment

03

Monitors ethical and legal compliance of ML applications

MLOps Engineer

01	02	03	04	05
SageMaker Model Registry	**SageMaker Pipelines**	**SageMaker Endpoint Deployment**	**SageMaker Model Monitor**	**SageMaker Clarify**
Manages model versions, approvals, and deployment status	Automates end-to-end ML workflows, including CI/CD integration	Deploys models to real-time, batch, or asynchronous endpoints	Monitors deployed models for drift in data quality or prediction accuracy	Assesses model explainability and fairness in production