باسمه‌تعالی

سند معماری فیزیکی

توسعه بستر MLOps

به سفارش

شرکت سامانه گستر سحاب پرداز

ارائه‌دهنده

شرکت فناوری اطلاعات و ارتباطات آدین

بهار ۱۴۰۳

فهرست مطالب

نشانی: تهران، بلوار اشرفی اصفهانی، خ قموشی، خ بهار، دانشگاه علم و فرهنگ، ط ۶، پارک ملی علوم و فناوری‌های نرم و صنایع فرهنگی، واحد ۱۰۱۴

نشانی رایانامه: info@adin-co.ir

شماره تماس: ۰۲۱-۴۴۳۷۴۵۶۹

مقدمه

سند حاضر به‌منظور ارائه معماری فنی و فیزیکی در پروژه «**توسعه بستر MLOps**» تهیه شده است. در این بخش به بررسی معماری فیزیکی در بستر MLOps خواهیم پرداخت. بدین منظور راهکار پیشنهادی شرکت آدین و جزئیات مربوط به معماری پیاده‌سازی شده، جریان داده‌ای و فناوری‌های به‌کاررفته در معماری فیزیکی تشریح شده است. لازم به ذکر است که انتخاب فناوری‌ها در معماری فیزیکی MLOps، به نیازهای کاربر و پروژه‌های یادگیری ماشینی درحال‌توسعه بستگی دارد. معماری باتوجه‌به نیاز سازمان، ممکن است نیاز به سطوح مختلفی از مقیاس‌پذیری، انعطاف‌پذیری و کارایی داشته و قادر به رسیدگی به نیازهای جریان کار یادگیری ماشین و ادغام و استقرار مداوم مدل‌های یادگیری ماشین باشد.

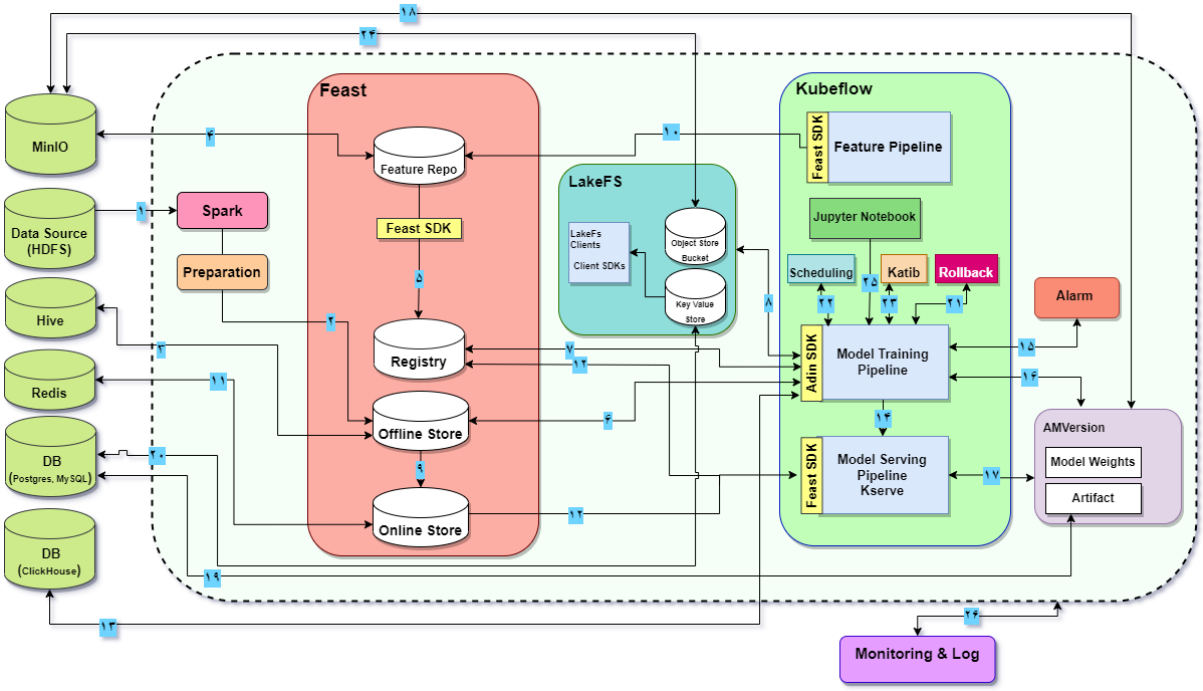
معماری فیزیکی

برای معماری فیزیکی (مطابق با معماری منطقی ارائه شده در مستندات قبل و نیز نیازها و مسائل مطرح شده در جلسات کارفرما)، نمونه‌ای مطابق شکل ۱ پیشنهاد می‌شود که هر یک از فناوری‌های پیشنهادی قابل‌استفاده در هر مؤلفه بستر MLOps نیز در آن آمده است. لازم به ذکر است که این پیشنهاد اولیه برای معماری فیزیکی است و باتوجه‌به پیشرفت کار، هر یک از اجزای آن ممکن است در طول فرایند اجرا و پیاده‌سازی تغییر کنند. پیشنهاد اولیه فناوری‌ها در لایه فیزیکی در شکل ۱ نمایش داده شده است.

نشانی: تهران، بلوار اشرفی اصفهانی، خ قموشی، خ بهار، دانشگاه علم و فرهنگ، ط ۶، پارک ملی علوم و فناوری‌های نرم و صنایع فرهنگی، واحد ۱۰۱۴

نشانی رایانامه: info@adin-co.ir

شماره تماس: ۰۲۱-۴۴۳۷۴۵۶۹



شکل ۱ . نمودار فیزیکی بستر MLOps

مؤلفه‌های اصلی بستر MLOps در شکل ۱ نشان‌داده‌شده است که در آن فناوری‌های مختلفی را برای ذخیره‌سازی ویژگی، آموزش مدل، استنتاج، هشدار و نظارت با یکدیگر ادغام می‌کند. هر فناوری یا جریانی می‌تواند در یک محیط تولید ارائه شده و بر عملکرد و دقت آنها نظارت شود. در این نمودار فلش‌ها جهت جریان داده یا تعامل بین اجزای مختلف در معماری فیزیکی پیشنهادی MLOps را نشان می‌دهند. در ادامه شرحی از جریان‌های داده‌ای شماره‌گذاری شده در نمودار ارائه می‌شود:

۱. اتصال منبع داده (HDFS) به Feast به کمک Spark: این جریان نشان‌دهنده انتقال داده خام از منبع اصلی داده **HDFS** به **Spark** است. Apache Spark داده‌ها را برای پردازش از این File System توزیع شده دریافت می‌کند.

۲،۳. اتصال Spark به‌عنوان منبع داده Feast به انبار ویژگی آفلاین، و ذخیره نتایج موقت در Hive:

داده‌ها از طریق Spark از منبع اصلی داده به مخزن ویژگی آفلاین Feast منتقل می‌شوند. پس از دریافت داده‌ها از طریق Spark، داده‌ها پردازش شده و سپس به مخزن ویژگی آفلاین Feast ارسال می‌شوند. این

داده‌ها در Hive به عنوان Store موقت ذخیره شده و به‌عنوان ویژگی‌های جدید ایجاد شده یا به‌روز شده برای استفاده در یادگیری ماشین مورداستفاده قرار می‌گیرند.

نکات: Feast می‌تواند داده‌ها را از منابع مختلف بارگیری کرده، عملیات مختلفی بر روی آن‌ها انجام داده و در نهایت ویژگی‌ها را ذخیره کند. برای استفاده از Feast به عنوان Offline feature store، باید یک منبع داده (Data source) و یک محل ذخیره‌سازی موقت (Store) به آن معرفی کرد. هر منبع داده، از Store های خاص خود پشتیبانی می‌کند. با استفاده از اسپارک به‌عنوان یک منبع داده برای Feast، می‌توان داده‌ها را از جداول یا فایل‌ها به کمک اسپارک خواند و پس از پردازش، آنها را در Store ذخیره کرد، که در این حالت می‌توان داده خروجی را از طریق Hive یا در حافظه ذخیره کرد. با توجه به این که این معماری برای حجم بالا طراحی شده است، در معماری ذخیره در حافظه دیده نشده و تنها Hive پشتیبانی می‌شود.

استفاده از Spark به عنوان منبع داده، امکان استفاده از قابلیت‌های پردازش قدرتمند اسپارک را در چارچوب Feast Feature Store برای برنامه‌های کاربردی یادگیری ماشینی فراهم می‌آورد. درحالی‌که ادغام اسپارک با Feast مزیت استفاده از قابلیت‌های پردازش داده سریع و کارآمد اسپارک را فراهم می‌کند، لازم است که پوشش و پایداری این ادغام نیز در نظر گرفته شود. زیرا منبع داده اسپارک در Feast، رسمی نبوده و به صورت contrib عرضه شده و پوشش تست کاملی نداشته و نباید به‌صورت پیش‌فرض با پایداری کامل فرض شود؛ بنابراین، درحالی‌که اسپارک می‌تواند افزودنی قدرتمندی به تنظیمات Feast باشد، بسیار حایز اهمیت است که ادغام آن در محیط عملیاتی کارفرما به‌دقت تست و تأیید شده تا از عملکرد آن اطمینان حاصل شود.

۴،۵. رجیستری Feast و محل ذخیره‌سازی ویژگی‌ها: برای مدیریت تعاریف ویژگی‌های کنترل‌شده با نسخه، ساخت و بازیابی مجموعه‌های داده‌های آموزشی از انباره آفلاین، و بازیابی ویژگی‌های آنلاین به کار می‌رود. همچنین، انباره ویژگی‌ها، تعاریف و محل ذخیره‌سازی ویژگی‌ها را مشخص می‌کند. رجیستری Feast، کاتالوگی از تعاریف ویژگی و متاداده‌هاست. Feast با یک رجیستری برای ذخیره متادیتای مرتبط با ویژگی‌ها تعامل می‌کند که می‌تواند شامل به‌روزرسانی رجیستری با تعاریف ویژگی جدید یا متادیتا در مورد ویژگی‌ها باشد. به طور پیش‌فرض، Feast از پیاده‌سازی رجیستری مبتنی بر فایل استفاده می‌کند که نمایش protobuf رجیستری را به‌عنوان یک فایل سریالیزه‌شده ذخیره می‌کند. این فایل رجیستری می‌تواند در یک سیستم فایل

محلی یا در ذخیره‌سازی ابری (مانند S۳، GCS یا Azure) ذخیره شود. در نتیجه می‌توان با استفاده از پروتوکل S۳ آن را به MinIO متصل کرد.

۶،۷،۸. انباره آفلاین، رجیستری Feast و نسخه‌گذاری داده LakeFS برای داده‌های آموزشی: اگرچه Feast به طور مستقیم مدیریت نسخه‌بندی داده‌ها را انجام نمی‌دهد، ولی می‌تواند در کنار سیستم‌های کنترل نسخه همچون LakeFS کار کند. فرادادەهای متناظر در رجیستری Feast این اجازه را می‌دهد که نسخه‌گذاری صحیح داده‌ها برای آموزش و سرویس‌دهی مدل‌های یادگیری ماشین صورت گیرد. از طریق Feast SDK، تعاریف ویژگی‌های ثبت‌شده در رجیستری بازیابی شده و به کمک تابع get_historical در Feast داده‌های نسخه‌بندی شده برای آموزش مدل‌ها به کار می‌رود. بنابراین پس از این که Offline store داده‌ها را از طریق اسپارک خواند و پردازش کرد، و به صورت موقت در Hive ذخیره کرد، از طریق Adin SDK این داده‌ها از Hive بازیابی می‌شوند و به کمک LakeFS نسخه‌گذاری شده و در Object store ذخیره می‌شوند، و داده‌های موقت ایجاد شده در Offline store در این مرحله پاک می‌شوند.

تنظیمات محل ذخیره‌سازی داده‌ها و فراداده‌ها در فایل feature_store.yaml پروژهٔ Feast مشخص شده، و لازم است که LakeFS بداند که داده‌ها و متاداده‌ها را از کجا پیدا کرده و با آنها ارتباط برقرار کند. داده‌های تحت کنترل نسخه‌ای که توسط LakeFS مدیریت می‌شوند، توسط خط لوله آموزش مدل که بخشی از Kubeflow است، مورداستفاده قرار می‌گیرند. این امر باعث اطمینان از سازگاری و قابلیت ردیابی در داده‌های آموزشی می‌شود. معمولاً این تنظیمات در مخزن کنترل‌شده با نسخه مدیریت می‌گردد و با استفاده از Adin SDK در خطوط لوله Kubeflow، این همگام‌سازی انجام شده تا پیوستگی در دو زیر ساخت Feast و LakeFS برای نسخه‌بندی داده‌ها مبتنی بر ویژگی‌های Feast انجام شود.

۹. انباره آفلاین، آنلاین و اتصال آنها در Feast: انباره آفلاین، جایی است که ویژگی‌ها ذخیره می‌شود و شامل ویژگی‌هایی است که می‌تواند برای تجزیه‌وتحلیل یا آموزش استفاده گردد. انباره آنلاین، جایی است که Feast ویژگی‌ها را برای دسترسی با تأخیر کم ذخیره می‌کند. این مخزن برای تسریع در استنتاج طراحی شده است. بعد از آموزش و پیاده‌سازی مدل، آخرین ویژگی‌ها برای استنتاج در انباره آنلاین قرار می‌گیرد و هرگاه ویژگی جدیدی اضافه شد، انباره آنلاین به‌روز نگهداری می‌شود. SDK Feast جهت بازیابی مجموعه داده‌های آموزشی از انباره آفلاین برای آموزش مدل‌ها به کار برده می‌شود.

۱۰. استخراج ویژگی‌های مدنظر و ذخیره ساختار آن در Feature Repo: هنگامی که پس از بررسی داده‌ها توسط مهندس ویژگی (Feature Engineer) ویژگی‌های مورد نیاز استخراج می‌شود، توسط فراخوانی دستور feast apply این ویژگی‌ها در Feature Repo ذخیره شده و از طریق Registry برای استفاده در Offline Store و Online Store در دسترس قرار می‌گیرند.

۱۱. Redis به‌عنوان انباره ویژگی آنلاین Feast: Redis به‌عنوان مخزن ویژگی آنلاین در حافظه، داده‌های بلادرنگ را به‌Feast ارسال می‌کند. این بدان معنی است که Feast می‌تواند هم با داده‌های دسته‌ای از Spark و هم با منابع داده بلادرنگ کار کند. ویژگی موردنیاز این بخش، سرعت بالای پاسخ‌گویی آن است. درعین‌حال حجم داده موردنیاز در این بخش نسبت به انباره ویژگی آفلاین به‌مراتب پایین‌تر است. تکنولوژی مورداستفاده انباره ویژگی آنلاین استفاده از یک پایگاه‌داده درون حافظه است که باتوجه‌به حجم موردنیاز مشتری، معمولاً Redis پاسخ‌گوی نیاز مشتری خواهد بود. در صورت نیاز به بالابردن حجم داده موردنیاز برای پاسخ‌گویی به مشتری‌های مختلف در سرویس ابری، می‌توان از Redis cluster و یا واردکردن داده روی چند سرویس مستقل Redis استفاده کرد.

۱۲. اتصال انباره ویژگی آنلاین به KServe: ویژگی‌های ذخیره شده در انباره ویژگی آنلاین، در استنتاج توسط KServe که بخشی از Kubeflow است، مورداستفاده قرار می‌گیرند. به علت استفاده از Feast SDK برای دریافت ویژگی‌ها، ابتدا از طریق Registry ساختار ویژگی‌ها دریافت شده و سپس خود ویژگی‌ها از Online store دریافت می‌شوند.

۱۳. ذخیره داده‌های نسخه‌گذاری شده پیش از آموزش در ClickHouse: به علت نیاز به استفاده داده‌ها توسط ابزارهای تحلیلی EDA همچون MetaBase، کتابخانه طراحی‌شده آدین، داده‌های پیش از آموزش را در پایگاه‌داده ClickHouse ذخیره می‌کند تا توسط این ابزارها قابل دسترس باشد.

۱۴. آموزش برنامه‌ریزی شده با جای‌گذاری خودکار مدل زیر بار برای استنتاج: ابتدا آموزش برنامه‌ریزی شده صورت می‌گیرد و پس از بررسی نتایج مدل جدید، در صورت برتری نتایج آن، جای‌گذاری خودکار مدل جدید با مدل زیر بار قبلی برای استنتاج انجام می‌شود.

۱۵. هشدار: هشدارها می‌توانند بر اساس عملکرد، سلامت داده‌ها یا مدل‌های آموزش‌دیده تشخیص داده شده و لازم است توسط یک سرویس جداگانه نظارت شوند. سیستم هشدار، با تشخیص انواع مشکلات Drift در مسیر آموزش مدل، آن را به سیستم‌های مانیتورینگ گزارش می‌دهد و امکان هشداردهی بلادرنگ و تحلیل پس/پیش از واقعه را فراهم می‌کند.

۱۶، ۱۷، ۱۸، ۱۹. رجیستری مدل به‌کاررفته در خط لوله آموزش و استنتاج مدل: سیستم رجیستری مدل AMVersion در MLOps، یک بخش ضروری برای مدیریت مدل‌های یادگیری ماشین باهدف ارائه یک مخزن مرکزی برای ذخیره‌سازی، نسخه‌بندی و ردیابی مدل‌های ML است که قابلیت ذخیره متادیتا به‌صورت فایل (MinIO) یا پایگاه‌داده (Postgers) را در دو نسخه متفاوت فراهم می‌آورد. این بخش در خط لوله آموزش و استنتاج مدل از اهمیت ویژه‌ای برخوردار بوده و ویژگی‌های زیر را برای این خط لوله فراهم می‌کند:

ذخیره‌سازی متمرکز: سیستم ثبت مدل یک مرکز متمرکز ارائه می‌دهد که در آن تیم‌ها می‌توانند مدل‌های ML خود را ذخیره و مدیریت کنند. این متمرکزسازی، دسترسی و کنترل آسان‌تر بر مدل‌ها را تسهیل می‌کند و اطمینان حاصل می‌شود که اعضای تیم می‌توانند به‌راحتی، مدل‌ها را پیدا و استفاده کنند.

کنترل نسخه مدل: همانند کد، مدل‌های ML باگذشت زمان از طریق تکرارها و بهبودهای گوناگون تکامل می‌یابند. سیستم ثبت مدل که از نسخه‌بندی مدل‌ها پشتیبانی می‌کند، به تحلیل‌گران داده اجازه می‌دهد تا نسخه‌های مختلف یک مدل را ردیابی کنند و عملکرد آن را در سراسر نسخه‌ها مقایسه کنند و در صورت لزوم به‌راحتی به نسخه‌های قبلی بازگردند.

متادیتای مدل: در کنار ذخیره مدل واقعی، سیستم رجیستری، ثبت اطلاعات مرتبط با هر مدل را ذخیره می‌کند. این متادیتا می‌تواند شامل اطلاعاتی در مورد تاریخ ایجاد مدل، داده‌های استفاده شده برای آموزش آن، تاریخچه نسخه، معیارهای عملکرد و غیره باشد. این مجموعه غنی از متادیتا برای درک زمینه و عملکرد یک مدل ضروری است. قابلیت ذخیره متادیتا به‌صورت فایل (MinIO) یا پایگاه‌داده (Postgers) در دو نسخه متفاوت فراهم می‌شود.

مدیریت چرخه حیات مدل: سیستم رجیستری مدل از کل چرخه حیات یک مدل ML از توسعه و آموزش تا استقرار و نگهداری پشتیبانی می‌کند. این چرخه شامل ردیابی عملکرد مدل در طول زمان و مدیریت به‌روزرسانی‌ها یا جایگزینی‌های مدل‌ها در سیستم‌های تولیدی می‌شود.

یکپارچه‌سازی با خط لوله‌های Kubeflow: سیستم ثبت مدل با سایر اجزا مانند خط لوله‌های Kubeflow یکپارچه می‌شود. این یکپارچه‌سازی اجازه می‌دهد که فرایندهای خودکار برای آموزش مدل‌ها، ارزیابی عملکرد آنها و سپس ثبت خودکار مدل‌های موفق در سیستم ثبت مدل را انجام دهند.

کنترل دسترسی: در محیط‌هایی که چندین تیم یا کاربر با مدل‌های ML کار می‌کنند، سیستم ثبت مدل می‌تواند کنترل‌های دسترسی را اعمال کند و اطمینان حاصل کند که فقط کاربران مجاز می‌توانند به مدل‌های خاصی دسترسی پیدا کنند، آنها را تغییر دهند یا مستقر کنند. این برای حفظ یکپارچگی و امنیت مدل‌ها، به‌ویژه در برنامه‌های حساس، حیاتی است.

به‌اشتراک‌گذاری مدل‌ها: با ارائه یک مخزن مشترک برای مدل‌ها، سیستم ثبت مدل همکاری بهتری را بین تحلیل‌گران داده و توسعه‌دهندگان تسهیل می‌کند. تیم‌های مختلف می‌توانند مدل‌های خود را به اشتراک گذاشته، از مدل‌های آماده در پروژه‌های جدید استفاده کرده و بر روی مدل‌های مشترک به طور مؤثرتری همکاری کنند.

کشف و قابلیت استفاده مجدد مدل: ثبت مدل امکان پیداکردن مدل‌های موجود که ممکن است با پروژه‌های کنونی آنها مرتبط باشد را برای کاربران فراهم کرده، کار تکراری را کاهش داده و فرایند توسعه را تسریع می‌بخشد. همچنین کاربران می‌توانند بر اساس معیارهای مختلف از طریق سیستم ثبت AMVersion جستجو کنند تا مدل‌هایی را که نیازهای خاص آنها را برآورده می‌کند، پیدا کنند.

۲۰. ذخیره متادیتای نسخه‌گذاری داده در Postgres: این جریان نشان‌دهنده انتقال اطلاعات جهت نسخه‌گذاری داده توسط LakeFS به Postgres است و برای ذخیره متادیتای ساختاریافته از نسخه‌های مختلف داده استفاده می‌شود.

۲۱. بازگشت به نسخه پیشین (Rollback): این ویژگی، به کاربران این امکان را می‌دهد که در صورت بروز مشکل در استقرار جدید در محیط تولید، به صورت غیرخودکار به نسخه قبلی مدل بازگردند. قابلیت‌های

بازگشت به نسخه پیشین از طریق Kubeflow Pipelines انجام شده و امکان بازگشت خودکار به نسخه‌های قبلی مدل را در صورت خرابی استقرار یا کاهش عملکرد فراهم می‌کند. این مکانیزم اطمینان می‌دهد که سرویس‌دهی مدل حتی در صورت بروز مشکلات غیرمنتظره توسط مدل‌ها یا ویژگی‌های جدید، به طور قابل اعتماد و دقیق ادامه یابد. اجرای این فرایند به‌صورت دستی انجام می‌گیرد. بدین‌نحو که ابتدا «آموزش برنامه‌ریزی شده با جایگذاری خودکار مدل زیربار برای استنتاج» غیرفعال شده و خط لوله «آموزش برنامه‌ریزی شده» فعال میگردد تا مدل جدید به صورت خودکار زیربار استنتاج نرود و به صورت دستی، خط لوله استنتاج با مدل قبلی بارگزاری می‌شود.

۲۲. برنامه‌ریزی در خطوط لوله Kubeflow: شامل فرایند تعریف، مدیریت زمان و چگونگی اجرای مختلف اجزاء یا مراحل یک جریان کاری یادگیری ماشین در بستر Kubeflow است. خطوط لوله Kubeflow، یک پلتفرم برای ساخت، استقرار و مدیریت جریان‌های کاری ML چندمرحله‌ای را فراهم می‌کند که در آن هر مرحله به‌عنوان یک کانتینر مجزا تعریف شده است. برنامه‌ریزی در این زمینه می‌تواند شامل هماهنگ‌سازی این مراحل و زمان‌بندی اجرای خطوط لوله باشد. در ادامه انواع نحوه برنامه‌ریزی در خطوط لوله Kubeflow ذکر شده است:

هماهنگ‌سازی بین کانتینرها در خط لوله (نمودار جهت‌دار بدون دور): خطوط لوله Kubeflow به‌عنوان یک نمودار جهت‌دار بدون دور تعریف می‌شوند، جایی که هر گره یک مرحله در خط لوله را نشان می‌دهد و یال‌ها، ترتیب اجرا را تعریف می‌کنند. برنامه‌ریزی خط لوله این اطمینان را حاصل می‌کند که تک‌تک مراحل به ترتیب صحیح اجرا شده و وابستگی‌های بین مراحل رعایت شود. به‌عنوان‌مثال، مرحله پیش‌پردازش داده‌ها باید قبل از آموزش مدل کامل شود.

اجرای شرطی: خطوط لوله Kubeflow از عملیات شرطی پشتیبانی می‌کنند، به‌نحوی‌که برخی از مراحل فقط در صورت برآورده‌شدن شرایط خاصی اجرا می‌شوند. این منطق شرطی، بخشی از برنامه‌ریزی و هماهنگ‌سازی خط لوله است و اجازه می‌دهد جریان‌های کاری پویا بر اساس نتایج میانی شکل گیرند. در اینجا چند نمونه از اجراهای شرطی در خط‌های لوله Kubeflow ارائه شده است. این خط‌های لوله شرطی امکان مدیریت جریان‌های کاری هوشمندانه و کارآمد در Kubeflow را فراهم می‌آورند و به طور پویا به نیازهای عملیاتی و شرایط داده‌ها پاسخ می‌دهند.

- **اجرای مبتنی بر عملکرد مدل** خط لوله می‌تواند یک مدل را تنها در صورتی استقرار دهد که به آستانه عملکرد خاصی مانند دقت برسد.
- **شرط بررسی کیفیت داده** این نوع خط لوله پس از پیش‌پردازش، کیفیت داده‌ها را بررسی می‌کند و تنها در صورتی ادامه می‌یابد که داده‌ها استانداردهای خاصی را برآورده سازند.
- **شرط استفاده از منابع** اجرای مراحلی که نیازمند منابع محاسباتی زیادی هستند، می‌تواند بر اساس در دسترس بودن منابع کافی شرطی شود.
- **اجرای مبتنی بر زمان** برخی وظایف در خط لوله ممکن است فقط در زمان‌های خاصی مانند ساعات غیرپیک اجرا شوند تا از منابع بهینه استفاده شود و هزینه‌ها کاهش یابد.

اجرای موازی: مراحلی در خط لوله که به یکدیگر وابسته نیستند می‌توانند به‌صورت موازی برنامه‌ریزی شوند تا کارایی بهبودیافته و زمان کلی اجرای خط لوله کاهش یابد.

اجراهای Recurring: خطوط لوله Kubeflow می‌توانند برای اجرا در فواصل زمانی منظم برنامه‌ریزی شوند، این روند برای جریان‌های کاری که نیاز به اجرای دوره‌ای دارند، مانند پردازش داده‌های روزانه یا آموزش مجدد مدل‌ها به‌صورت هفتگی، بسیار مفید است.

برنامه‌ریزی Trigger-based: فراتر از Trigger زمان‌بندی شده، خطوط لوله همچنین می‌توانند برای اجرا در پاسخ به رویدادهای خارجی یا Trigger تنظیم شوند. به‌عنوان‌مثال، یک خط لوله می‌تواند به‌گونه‌ای تنظیم شود که هر زمان داده‌های جدیدی بارگذاری می‌شود، اجرا شود.

درخواست‌ها و محدودیت‌های منبع: هنگام برنامه‌ریزی مراحل خط لوله، می‌توان درخواست‌ها و محدودیت‌های منبع برای CPU، حافظه و سایر منابع مشخص کرد. بدین‌ترتیب در هر مرحله منابع لازم اختصاص‌داده‌شده و همچنین از انحصار منابع توسط هر مرحله جلوگیری می‌شود.

۲۳. Katib - تنظیم ابرپارامترهای خط لوله آموزش مدل: Katib مؤلفه‌ای از Kubeflow برای تنظیم ابرپارامترها (Hyperparameter Tuning) است. این سیستم فرآیند بهینه‌سازی ابرپارامترهای مدل را در طول آموزش به صورت خودکار انجام می دهد. در نمودار شکل ۱، Katib بخشی از فرآیند درون Kubeflow است

که از آن برای زمان‌بندی و هماهنگ‌سازی آزمایش‌ها برای بهینه‌سازی ابرپارامترها استفاده می‌شود و می‌تواند به طور خودکار بسیاری از کارهای آموزشی (آزمایش‌ها) را با ابرپارامترهای مختلف اجرا کند.

۲۴. ذخیره داده نسخه‌گذاری شده در MinIO: این جریان نشان‌دهنده انتقال داده نسخه‌گذاری شده توسط LakeFS به MinIO است و معمولاً برای ذخیره حجم زیادی از داده‌های غیرساختاریافته پیش از آموزش مدل استفاده می‌شود.

۲۵. محیط توسعه آزمایشی: در قالب پایپلاین CI/CD و نوت‌بوک‌های تعاملی با استفاده از Jupyter Notebooks درKubeflow، کاربران می‌توانند تضمین کنند که آزمایش‌ها به راحتی قابل تکرار بوده و در میان اعضای تیم با حفظ تنظیمات محیط و وابستگی‌ها به اشتراک گذاشته شوند. استفاده از Jupyter Notebooks در Kubeflow، ترکیبی از سهولت و انعطاف‌پذیری با استحکام و قابلیت مقیاس‌پذیری در Kubeflow را ترکیب می‌کند به‌نحوی‌که هم فاز توسعه و هم فاز استقرار پروژه‌های یادگیری ماشین را بهبود می‌بخشد. همچنین، با یکپارچه‌سازی فرآیندهای CI/CD از GitLab با Kubeflow، تیم‌ها می‌توانند به جریان‌های کاری هموارتر، خودکار و قابل تکرار دست یابند که این امر به طور قابل توجهی بار کاری در استقرار و مدیریت مدل‌های یادگیری ماشین را کاهش می‌دهد. این رویکرد همچنین در حفظ سازگاری، قابلیت اطمینان و چرخه‌های توسعه سریع‌تر در پروژه‌های یادگیری ماشین کمک می‌کند.

نکته: لازم به ذکر است Jenkins و CI/CD در شکل معماری ذکر نشده است و صرفا جهت تسهیل فرایند پیاده‌سازی معماری مورد استفاده قرار می‌گیرند.

۲۶. مانیتورینگ و لاگ: منظور از مانیتورینگ، فرایند ردیابی عملکرد مدل‌های ML در تولید است. نظارت، شامل جمع‌آوری داده‌ها در مورد پیش‌بینی‌های مدل، مقایسه آن‌ها با نتایج واقعی و شناسایی هرگونه اختلاف یا خطا است. برای جمع‌آوری و امکان تحلیل لاگ‌ها از ELK استفاده می‌شود. همچنین برای پایش متریک‌ها از Prometheus و Grafana استفاده خواهد شد.

موارد خاص توافق‌شده در اجرای پروژه

با توجه به برخی محدودیت‌های موجود در اجرای پروژه، امکان پشتیبانی از برخی موارد به دلایل مختلف در این فاز فراهم نمی‌باشد. این موارد در جلسات متعدد با کارفرما مورد بحث‌وبررسی قرار گرفته و به‌صورت خلاصه در ادامه بیان می‌شود:

- در حال حاضر، تنوع معماری وجود ندارد و در این فاز معماری موجود در این سند پیاده‌سازی و تحویل می‌شود.
- کنترل دسترسی به ستون‌ها (در منبع داده) خارج از دامنه این پروژه است، و به طور کلی کنترل دسترسی داده‌ها توسط MLOps پشتیبانی نمی‌شود.
- بر روی داده‌های خام نسخه‌بندی انجام نمی‌شود.
- در ارتباط با بحث ETL و Auto Hyperparameter tuning، طبق جمع‌بندی در جلسات صورت‌گرفته، این مفاهیم در پروژه فعلی نیستند. گزینه Hyperparameter tuning به‌صورت خودکار در نظر گرفته شده بود، اما در نهایت از محدوده پروژه حذف شده است.
- Metabase از پروژه MLOps حذف گردید؛ در صورت نیاز، سرویس‌دهی در گرانیت انجام می‌پذیرد.
- داده‌ها به طور معمول از حالت آفلاین به آنلاین منتقل می‌شوند.
- برای انتقال مستقیم داده‌ها به سیستم آنلاین، هیچ امکانی در MLOps موجود نیست؛ تنها یک API برای استفاده کاربران در دسترس قرار داده می‌شود تا داده‌های خود را به‌دلخواه در محیط آنلاین وارد کنند.
- تابه‌حال، هیچ توافقی برای تضمین سطح خدمات (SLA) برای uptime مطرح نشده است.
- (multi-tenancy) به طور کامل پشتیبانی نمی‌شود.
- نباید همواره در online feature store کل داده به‌صورت دسته‌ای (batch) پردازش شود، زیرا با وجود حجم زیاد داده، این کار به طور معمول امکان‌پذیر نیست. همچنین در مورد online feature store، در برخی موارد اطلاعات باید برای استنتاج به‌روزرسانی شود.
- مانیتورینگ مدل در حالت online inference نیاز نیست.

- در معماری فعلی، داده‌ها از Spark به Hive منتقل می‌شوند و نسخه‌هایی از داده‌ها در MinIO با LakeFS و در ClickHouse ذخیره می‌شوند. این روش، هرچند غیربهینه است، اما در حال حاضر قابل‌استفاده می‌باشد. در صورت نیاز، برای بهینه‌سازی آن در فازهای بعدی برنامه‌ریزی می‌کنیم.
- پروسه تحویل گیری بر اساس دو مسئله ساده تنظیم می‌شود. این دو مسئله باید به‌وضوح تعریف شده و مدنظر قرار گیرند تا از سادگی و شفافیت روند اطمینان حاصل شود.
- در سند آزمون، فقط باید عناوین مطرح شوند و نوع آزمون که آیا آزمون یکپارچگی (integration test) است باید مشخص گردد. این سند خودبه‌خود می‌تواند به‌صورت کد تهیه شود و فرآیند تست و تأیید در پیاده‌سازی کد صورت گیرد.
- آزمون‌های پذیرش (acceptance tests) تنها برای تعدادی از مؤلفه‌های پروژه موردنیاز هستند، نه برای تمامی آن‌ها.
- سطوح بلوغ بر اساس معیارهای تعریف شده روی کدهای آدین خواهد بود و برای کدهای عمومی و به‌کارگیری آنها در پایپلاین‌های Kubeflow نیاز به سطوح بلوغ نیست.
- پس از آنکه مدل به مرحله تولید رسید و عملیاتی شد، دیگر پایش و هشداردهی روی عملکرد مدل از وظایف MLOps تلقی نمی‌گردد و این امور باید خارج از حوزه MLOps صورت پذیرد.
- امکان کنترل دسترسی روی پایپلاین‌های MLOps وجود ندارد. باتوجه‌به مدل‌سازی، نسخه‌بندی و امکان برگشت به نسخه قبلی (Rollback)، به کمک API در محیط Kubeflow، با اجرای Run توسط API وجود دارد و برای حل مشکل دسترسی، تمرکز بر روی اجراها صورت خواهد گرفت که جایگزین مناسبی برای پایپلاین‌ها است، و در این زمینه مشکلی وجود ندارد. به‌ویژه برای دو پایپلاین تولیدی و آزمایشی می‌توان محدودیت‌های دسترسی را از طریق IDE و به‌صورت Run در داشبورد Kubeflow تعریف کرد تا از اجرا یا حذف توسط افراد غیرمجاز جلوگیری شود.

- فرآیند rollback مدل به یک نسخه یا زمان مشخصی امکان‌پذیر است که در آن نسخه‌ای معین freeze می‌گردد و در صورت نیاز، امکان unfreeze (بازگشت به حالت قبلی) نیز فراهم است. همچنین، امکان بازگشت به وضعیت پیشین (rollback) مدل به‌وسیله خود API Kubeflow ارائه می‌شود تا در هر زمان که نیاز بود بر اساس شاخص‌های کارایی کسب‌وکار یا سایر عوامل فراخوانی

شود. لازم است در زمان استفاده از این API، فرآیند CT متوقف شود و برای ادامهٔ آن، یک API دیگر ارائه شود.

- برای اطمینان از دریافت متریک‌ها و مشاهده آنها در زمان‌بندی، پیشنهاد می‌شود که فرض کنیم به متریک‌های مدنظر سرویس‌ها دسترسی وجود دارد. درصورتی‌که در جریان پیشرفت پروژه متوجه شویم که متریک‌ها ارائه نمی‌شوند، اگر این مسئله با تغییرات کوچک قابل‌حل باشد، پذیرفته خواهد شد؛ در غیر این صورت اگر تغییرات زمان‌بندی را تحت‌تأثیر قرار دهد، باید مجدد در این‌باره بحث‌وبررسی صورت گیرد.
- درصورتی‌که لاگ‌ها در stdout ثبت نشوند، بررسی شود که چگونه لاگ‌زنی انجام می‌گیرد تا بر اساس آن تصمیم‌گیری نماییم. همین روند باید برای تنظیمات زیر نیز پیگیری شود:

  - تنظیم زمان قطع ارتباط (connection timeout)
  - سیاست‌های Retry policies/circuit breaker
  - محدودیت نرخ (rate limit) و فشار پس‌زنی (back pressure)
  - connection pool
  - تعیین حجم بافرها، تعداد Threadها و غیره
  - استفاده از پسوردهای Hash شده
  - ثبت اطلاعات مهم در لاگ‌ها، هشدارها و سایر موارد دارای تاریخچه

- در مورد ظرفیت پاسخگویی یا نگهداری اطلاعات، تعیین محدودیت‌ها بر اساس SLO موردنیاز است. چون SLO مشخصی ارائه نمی‌شود و کاملاً بستگی به کد کاربر MLOps دارد؛ بنابراین باید به نحوی کنترل شود.
- بررسی آنکه کلاینت‌ها به‌درستی به سرویس‌های HA شده متصل می‌شوند یا خیر، در این فاز حذف شده است.
- قابلیت قراردادن پرچم ویژگی (feature flag) در این فاز حذف شده است.
- اصل کمترین اختیار (principle of least privilege) در این فاز حذف شده است.
- در اتصال به Spark ترجیح HDFS است، زیرا داده‌های خام در آنجا موجود است.
- توضیحات اضافی از HelmChartها بایدحذف شوند.

- دسترسی‌های دقیق بین سرویس‌ها (RBAC و RoleBinding ها) ضروری نیست و تمامی سرویس‌ها به یکدیگر دسترسی دارند.

تناظر موارد تحویل دادنی با مؤلفه‌های معماری فیزیکی

کلیه مفاد موجود در سند تحویل‌دادنی‌ها (LOM) که قابلیت تناظر با مفاد موجود در معماری فیزیکی MLOps را دارند، در جدول ۱ آمده است. لازم‌به‌ذکر است که برخی مفاد گروه DevOps، شامل استانداردسازی کدهای زیرساختی از این تناظر مستثنی شده‌اند.

جدول ۱ - تناظر موارد تحویل دادنی با مؤلفه‌های معماری فیزیکی

ردیف متناظر	گروه	عنوان تحویل دادنی	مؤلفه معماری متناظر
۱۶	ML	قابلیت Store Feature نسخه Spark	۱. منبع داده (HDFS) به Spark
			۲،۳. اتصال Spark به‌عنوان منبع داده Feast به انبار ویژگی آفلاین، و ذخیره نتایج موقت در Hive
۶۶	C.S	پایپ‌لاین CI/CD آموزش با قابلیت Offline Feature Store	۴،۵. رجیستری Feast و محل ذخیره‌سازی ویژگی‌ها
۱۰	ML	قابلیت Feature Store نسخه PostgreSQL	
۳۸	C.S	پایپ‌لاین CI/CD آموزش با قابلیت نسخه‌گذاری داده	۶،۷،۸. انباره آفلاین، رجیستری Feast و نسخه‌گذاری داده LakeFS برای داده‌های آموزشی
۱۰	ML	قابلیت Feature Store نسخه PostgreSQL	۹. انباره آفلاین، آنلاین و اتصال آنها در Feast
			۱۰. استخراج ویژگی‌های مدنظر و ذخیره ساختار آن در Feature Repo
۳۷	C.S	پایپ‌لاین CI/CD استنتاج با قابلیت Online Feature Store	۱۱. Redis به‌عنوان انباره ویژگی آنلاین Feast
۳۱	C.S	پایپ‌لاین CI/CD استنتاج با قابلیت نسخه‌گذاری مدل	۱۲. اتصال انباره ویژگی آنلاین به KServe
۳۹	C.S	پایپ‌لاین CI/CD استنتاج با قابلیت نسخه‌گذاری داده	
۳۸	C.S	پایپ‌لاین CI/CD آموزش با قابلیت نسخه‌گذاری داده	۱۳. ذخیره داده‌های نسخه‌گذاری شده پیش از آموزش در ClickHouse
۱۷	C.S	پایپ‌لاین CI/CD آموزش با قابلیت بازآموزی/آموزش پیوسته	۱۴. آموزش برنامه‌ریزی شده با جای‌گذاری خودکار مدل زیر بار برای استنتاج

۱۵. هشدار	قابلیت پایش کیفیت مدل و هشدار دهی	ML	۲۴
۱۶، ۱۷، ۱۸، ۱۹. رجیستری مدل به‌کاررفته در خط لوله آموزش و استنتاج مدل	پایپ‌لاین CI/CD آموزش با قابلیت نسخه‌گذاری مدل	C.S	۲۰
	قابلیت نسخه‌گذاری مدل نسخه ORM+DB	ML	۱۵
۲۰. ذخیره متادیتای نسخه‌گذاری داده در Postgres	قابلیت نسخه‌گذاری داده‌ها	ML	۱۷
۲۱. بازگشت به نسخه پیشین (Rollback)	قابلیت پایش کیفیت مدل و هشدار دهی	ML	۱۸
۲۲. برنامه‌ریزی در خطوط لوله Kubeflow	بازآموزی/آموزش پیوسته قابلیت	ML	۳۲
۲۳. Katib - تنظیم ابرپارامترهای خط لوله آموزش مدل	پایپ‌لاین CI/CD آموزش با قابلیت تنظیم هایپر پارامترها	C.S	۱۹
۲۴. ذخیره داده نسخه‌گذاری شده درMinIO	قابلیت نسخه‌گذاری مدل نسخه MinIO	ML	۲۲
۲۵. محیط توسعه آزمایشی	پایپ‌لاین CI/CD استنتاج مسئله دوم	C.S	۱۴
	پایپ‌لاین CI/CD استنتاج مسئله اول	C.S	۱۲
	پایپ‌لاین CI/CD آموزش مسئله اول	C.S	۱۱
	پایپ‌لاین CI/CD آموزش مسئله دوم	C.S	۱۳
	ارائه نوت‌بوک‌های تعاملی به‌منظور توسعه مدل	ML	۲۲
۲۶. مانیتورینگ و لاگ	پایش و لاگ عملکرد مولفه‌های زیرساختی MLOps	DevOps	۴۲
	پایش و لاگ عملکرد مولفه‌های عملکردی MLOps	DevOps	۴۳