بسمه‌تعالی

**پیشنهاد پروژه** **MLOps**

**شرکت آدین**



اردیبهشت ۱۴۰۲

نسخه ۱.۰۰

# مشخصات کلی

MLOps (عملیات یادگیری ماشین) به مجموعه‌ای از فرایندها، ابزارها و شیوه‌ها جهت مدیریت چرخه مدل‌های یادگیری ماشین در یک محیط تولید، اشاره دارد. همچنین این چرخه شامل همکاری بین دانشمندان داده، مهندسان و تیم‌های DevOps است به‌نحوی‌که این اطمینان حاصل شود که مدل‌ها به طور مؤثر توسعه، استقرار، نظارت و به‌روزرسانی می‌شوند. هدف MLOها افزایش سرعت، قابلیت اطمینان و مقیاس‌پذیری مدل‌های یادگیری ماشین در تولید است، درحالی‌که خطرات ناشی از ریسک عدم موفقیت را نیز کاهش می‌دهد. همچنین به‌کارگیری MLOps فرایند مدیریت را ساده‌تر کرده، کیفیت را افزایش می‌دهد و استقرار مدل‌های یادگیری عمیق و یادگیری ماشین را در محیط‌های تولید در مقیاس بزرگ را خودکار می‌کند. هدف، بهبود خودکارسازی و ارتقای کیفیت مدل‌های تولید و درعین‌حال توجه به الزامات تجاری و نظارتی است.

MLOps به‌ویژه در مورد استقرار مدل‌های ML در تولید اهمیت دارد، زیرا به سازمان‌ها کمک می‌کند تا مطمئن شوند که مدل‌هایشان در طول زمان دقیق، قابل‌اعتماد و کارآمد هستند. به‌طورکلی، MLOps با خودکار کردن بسیاری از مراحل مربوط به استقرار و مدیریت مدل‌های ML، به دانشمندان و مهندسان داده اجازه می‌دهد تا با همکاری یکدیگر به ارائه سریع‌تر و کارآمدتر مدل‌های یادگیری ماشین دست یابند. برای رسیدن به اهداف فوق، تیم‌های MLOps معمولاً از ترکیبی از ابزارها، فرایندها و شیوه‌ها زیر را استفاده می‌کنند:

* کنترل نسخه برای مدل‌های ML و مجموعه‌داده‌ها
* آزمون خودکار و ادغام مداوم
* کانتینرسازی و ارکستراسیون برای استقرار مدل
* نظارت و ثبت گزارش برای ردیابی عملکرد مدل در تولید
* حاکمیت و انطباق برای مدل‌ها

مدیریت سیستم‌های هوش مصنوعی در مقیاس بزرگ کار آسانی نیست و در این مسیر چالش‌های مهمی وجود دارد که تیم‌ها باید با آن مواجه شوند. برخی از این چالش‌ها در کاربرد یادگیری ماشین (ML) در یک محصول مقیاس‌پذیر عبارت‌اند از:

کیفیت و کمیت داده: اطمینان از وجود داده‌های آموزشی باکیفیت بالا برای آموزش مدل‌های ML یک چالش بزرگ است.

عملکرد مدل: مدل‌های ML گاهی اوقات ممکن است برای تعمیم به داده‌های جدید مشکل داشته باشند و ممکن است در صورت استقرار در تنظیمات دنیای واقعی ضعیف عمل کنند.

تفسیرپذیری مدل: مدل‌های ML می‌توانند پیچیده و غیرقابل‌درک باشند، و تفسیر پیش‌بینی‌ها و تشخیص مشکلات، آن‌ها را به چالش می‌کشد.

منابع محاسباتی: آموزش و به‌کارگیری مدل‌های ML بزرگ و پیچیده می‌تواند به مقدار قابل‌توجهی از منابع محاسباتی نیاز داشته باشد که می‌تواند در یک محصول مقیاس‌پذیر چالش‌برانگیز باشد.

استقرار مدل: استقرار مدل‌های ML در یک محیط تولید و ادغام آن‌ها در یک محصول می‌تواند چالش‌برانگیز باشد و نیاز به بررسی دقیق زیرساخت‌ها، حریم خصوصی داده‌ها و امنیت دارد.

نگهداری مدل: مدل‌های ML باید با تغییر داده‌ها و نیازمندی‌های محصول، به‌روزرسانی و بازآموزی شوند که می‌تواند در یک محصول مقیاس‌پذیر یک چالش باشد.

بایاس: مدل‌های ML گاهی اوقات می‌توانند سوگیری‌های موجود را در داده‌های آموزشی را تداوم بخشند و توجه به این مسائل هنگام توسعه یک محصول مقیاس‌پذیر ML بسیار مهم خواهد بود.

## MLOPs برای چه مشتریانی مناسب است؟

به‌طورکلی، MLOps یک چارچوب قدرتمند است که می‌تواند به تیم‌ها کمک کند تا پروژه‌های یادگیری ماشینی را به طور مؤثر مدیریت و مقیاس کنند. از سوی دیگر عواملی وجود دارد که ممکن است MLOps بهترین گزینه برای حل برخی از مسائل باشد. در اینجا چند نمونه از این دلایل ذکر می‌شود:

* **بهبود سرعت ایجاد و استقرار مدل:** در نتیجه مدیریت جامع چرخه زندگی یادگیری ماشین، به دلیل افزایش ارزش یادگیری ماشین.
* **زمان سریع‌تر جهت بازاریابی:** MLOps دانشمندان و مهندسان داده را قادر می‌سازد تا سریع‌تر تکرار کنند و مدل‌ها را سریع‌تر به تولید بفرستند. همچنین روش‌های یکپارچه‌سازی و تحویل مداوم، استقرار این سیستم‌ها را در تولید آسان‌تر می‌کند.
* **بهبود کیفیت مدل:** با خودکارسازی آموزش، اعتبارسنجی و استقرار مدل‌ها، احتمال خطای انسانی کاهش می‌یابد که منجر به عملکرد بهتر مدل می‌شود. همچنین، جهت اطمینان از ارائه پیش‌بینی‌های باکیفیت بالا، سیستم MLOps باید قادر به اندازه‌گیری دریفت مدل[[1]](#footnote-1) باشد. این امر احتمال دیدگاه‌های نادرست در فرضیه‌های مدل را کاهش می‌دهد.
* **همکاری بهتر:** MLOps همکاری بین دانشمندان داده، توسعه‌دهندگان و تیم‌های عملیاتی را تشویق می‌کند و تیم‌های متقابل را قادر می‌سازد تا به طور مؤثرتری با هم کار کنند. MLOها همکاری و شفافیت را در میان تیم‌های dataOps، مهندسین یادگیری ماشین، تحلیلگران تجاری/مدیران محصول، مهندسین تضمین کیفیت و مهندسین زیرساخت افزایش می‌دهند.
* **مقیاس‌پذیری:** خودکار کردن فرایند استقرار، مقیاس‌پذیری مدل‌ها را جهت رسیدگی به مقادیر بزرگ و افزایشی داده و جمعیت بیشتر کاربران آسان‌تر می‌کند.

## چالش‌های MLOps

* **پیچیدگی**: پیاده‌سازی MLOها می‌تواند پیچیده باشد و به مهارت‌های تخصصی و درک عمیق فرایندهای یادگیری ماشین و توسعه نرم‌افزار نیاز دارد. MLOps به ابزارها و مهارت‌های فراوانی نیاز دارد که به پیچیدگی آن می‌افزاید. جهت ساختن یک چارچوب موفق MLOps، به یک معمار باتجربه MLOps و مهندسان ML نیاز داریم که از قبل با اصول مفهوم DevOps آشنا هستند.
* **چالش‌های یکپارچه‌سازی**: ادغام جریان‌های کاری ML در فرایندهای توسعه نرم‌افزار موجود می‌تواند دشوار باشد و ممکن است نیاز به تغییرات قابل‌توجهی در جریان‌های کاری و ابزارهای موجود داشته باشد.
* **امنیت داده‌ها**: MLOها به مدیریت دقیق داده‌های حساس از جمله ذخیره‌سازی ایمن، کنترل دسترسی و نظارت بر استفاده از داده‌ها نیاز دارند.
* **محدودیت‌های منابع:** پیاده‌سازی MLOها نیازمند سرمایه‌گذاری قابل‌توجهی در فناوری، فرآیندها و کارکنان است که ممکن است جهت همه سازمان‌ها امکان‌پذیر نباشد. MLOps جهت استقرار مؤثر مدل ML در دنیای واقعی ضروری است، اما هزینه آن به دلیل پیچیدگی و منابع موردنیاز می‌تواند بسیار گران باشد. توانایی ایجاد و حفظ یک چرخه حیات MLOps به طور مؤثر به منابع کافی (افراد، زیرساخت‌ها و ابزارها) نیاز دارد. عامل اصلی تعیین‌کننده در استفاده یا عدم استفاده از MLO ها، بررسی کامل بازگشت سرمایه ابتکار یادگیری ماشین است.

## MLOps برای چه دسته مسائلی مناسب نیست؟

به‌طورکلی، MLOps یک چارچوب قدرتمند است که می‌تواند به تیم‌ها کمک کند تا پروژه‌های یادگیری ماشینی را به طور مؤثر مدیریت و مقیاس کنند. بااین‌حال، همیشه برای هر مشکل یا سناریویی مناسب نیست، و تیم‌ها باید قبل از اتخاذ MLO به‌دقت نیازهای خود را در نظر بگیرند. مسائل خاصی وجود دارد که ممکن است MLOps بهترین گزینه برای آنها نباشد. در اینجا چند نمونه هستند:

**پروژه‌های در مقیاس کوچک:** MLOps برای کمک به تیم‌ها در مقیاس پروژه‌های یادگیری ماشینی طراحی شده است که می‌تواند مدل‌ها و منابع داده زیادی را شامل شود. اگر روی یک پروژه در مقیاس کوچک فقط با یک یا دو مدل کار می‌کنید، ممکن است MLOها بیش از حد باشد.

**تیم‌های غیرفنی:** MLOها به سطح خاصی از تخصص فنی برای پیاده‌سازی و نگهداری نیاز دارند. اگر تیم شما مهارت‌های لازم را ندارد، ممکن است اتخاذ و استفاده مؤثر از MLOها چالش‌برانگیز باشد.

**نمونه‌سازی سریع:** اگر در حال آزمایش مدل‌ها و الگوریتم‌های مختلف یادگیری ماشین هستید، ممکن است به‌سختی و ساختار MLOها نیاز نداشته باشید. در عوض، ممکن است رویکرد چابک‌تری را ترجیح دهید که به شما امکان می‌دهد ایده‌ها را سریع تکرار و آزمایش کنید.

**سیستم‌های قدیمی:** MLOها ممکن است با برخی از سیستم‌های قدیمی سازگار نباشند که می‌تواند ادغام در زیرساخت‌های موجود شما را دشوار کند.

**مدل‌های ساده:** MLOps برای مدل‌های پیچیده یادگیری ماشینی که نیاز به آموزش و بهینه‌سازی گسترده دارند، بسیار مفید است. برای مدل‌های ساده که به داده‌ها یا منابع محاسباتی زیادی نیاز ندارند، MLO ‌ها ممکن است غیرضروری باشند.

**دردسترس‌بودن و کیفیت داده‌ها**: MLOps فرض می‌کند که شما به داده‌های باکیفیت و قابل‌اعتمادی دسترسی دارید که می‌توانند برای آموزش و آزمایش مدل‌های یادگیری ماشین خود استفاده شوند. اگر داده‌های شما ناقص، پر از نویز یا کیفیت پایین باشد، MLOها ممکن است مؤثر نباشند و ممکن است لازم باشد قبل از استفاده از MLO ها، روی پاک‌سازی و آماده‌سازی داده‌ها سرمایه‌گذاری کنید.

**انطباق با مقررات:** MLOها می‌توانند به تیم‌ها در مدیریت چرخه عمر مدل‌های یادگیری ماشین کمک کنند، اما ممکن است برای انطباق با مقررات در صنایع خاص با قیود سخت‌گیرانه، مانند مراقبت‌های بهداشتی یا مالی کافی نباشد. در این موارد، ممکن است اقدامات بیشتری لازم باشد تا اطمینان حاصل شود که مدل‌ها اخلاقی، شفاف و مطابق با قوانین و مقررات قابل‌اجرا هستند.

**محدودیت‌های هزینه و منابع:** MLOها می‌توانند به منابع زیادی دسترسی داشته و به سرمایه‌گذاری قابل‌توجهی در زیرساخت‌ها، ابزارها و کارکنان نیاز دارند. اگر تیم شما با منابع یا بودجه محدود کار می‌کند، ممکن است اتخاذ و نگهداری مؤثر MLOها چالش‌برانگیز باشد.

**محیط‌های پویا:** MLOps فرض می‌کند که زیرساخت و منابع داده شما نسبتاً پایدار و قابل‌پیش‌بینی هستند. اگر در یک محیط بسیار پویا، مانند یک برنامه IoT یا یک خط لوله داده‌های جریانی کار می‌کنید، MLOها ممکن است مناسب نباشند، و رویکردهای جایگزین مانند یادگیری آنلاین یا یادگیری تقویتی ممکن است مناسب‌تر باشند.

به طور خلاصه، درحالی‌که MLOها می‌توانند برای مدیریت و مقیاس پروژه‌های یادگیری ماشینی بسیار مؤثر باشند، اما راه‌حل یکسانی نیست. تیم‌ها باید قبل از اتخاذ MLOها نیازها، منابع و محدودیت‌های خود را به‌دقت در نظر بگیرند و اگر MLO برای مشکل آنها مناسب نیست، رویکردهای جایگزین را بررسی کنند.

## گزینه‌های مطرح MLOps و مزایای انتخابی Kubeflow

Kubeflow و MLflow هر دو پروژه‌های منبع‌باز برای مدیریت گردشکار یادگیری ماشین هستند. بااین‌حال، بررسی آنها در ابعاد مختلف نشانگر تفاوت‌های اساسی میان این دو پروژه است:

* هدف: Kubeflow به طور خاص برای اجرا و استقرار مدل‌های یادگیری ماشین در Kubernetes، یک سیستم منبع‌باز محبوب برای خودکارسازی استقرار، مقیاس‌بندی و مدیریت برنامه‌های کاربردی کانتینری طراحی شده است. از سوی دیگر، MLflow یک پلتفرم کلی‌تر برای مدیریت کل چرخه زندگی یادگیری ماشینی است، از جمله آزمایش‌های ردیابی، کد بسته‌بندی در اجراهای تکرارپذیر، و به‌اشتراک‌گذاری و استقرار مدل‌ها.
* ویژگی‌ها: Kubeflow مجموعه‌ای از ابزارها و APIها را برای ساخت، استقرار و مدیریت مدل‌های یادگیری ماشین در Kubernetes ارائه می‌کند، از جمله نوت‌بوک‌های Jupyter، آموزش TensorFlow و سرویس‌دهی مدل. MLflow طیف گسترده‌تری از ویژگی‌ها را برای چرخه زندگی یادگیری ماشین ارائه می‌کند، از جمله ردیابی آزمایش، بسته‌بندی کد و اشتراک‌گذاری، و استقرار مدل در پلتفرم‌های مختلف از جمله Docker، AzureML، و Amazon SageMaker.
* یکپارچه‌سازی: Kubeflow به‌خوبی با سایر اجزای اکوسیستم Kubernetes، مانند Helm (یک مدیر بسته برای Kubernetes) ادغام می‌شود، و همچنین ابزارهایی را برای کاهش حجم کاری یادگیری ماشین در مجموعه‌ای از ماشین‌ها فراهم می‌کند. MLflow را می‌توان با انواع کتابخانه‌ها و پلتفرم‌های یادگیری ماشین استفاده کرد و یک REST API برای ادغام با ابزارهای دیگر ارائه می‌دهد.
* به طور خلاصه، Kubeflow برای مدیریت بارهای کاری یادگیری ماشین در Kubernetes مناسب است، درحالی‌که MLflow یک پلتفرم کلی‌تر برای مدیریت کل چرخه زندگی یادگیری ماشین است.

## نمونه‌های جهانی استفاده از Kubeflow

* از جمله شرکت‌ها و سازمان‌هایی که از Kubeflow به‌طورجدی برای آسان‌سازی فرانید چرخه یادگیری ماشین استفاده می‌کنند می‌توان به مجموعه‌ای در شرکت سامسونگ با نام Samsung SDS اشاره کرد. در واقع این مجموعه Kubeflow را به‌عنوان پلتفرم ابری منطبق با Kubernetes انتخاب کرده و به مشتریان خود بر بستر Samsung Cloud Platform پیاده‌سازی و خدمات ارائه می‌دهد. [[2]](#footnote-2)
* این مجموعه که زیرمجموعه‌ای از شرکت سامسونگ است در زمینه‌های مختلفی سرویس، آموزش و خدمات به Enterpriseها ارائه می‌کند. در زمینه یادگیری ماشین به‌طورجدی از Kubeflow استفاده کرده و به‌صورت آموزش و در کنار شرکت‌های بزرگ همکاری می‌کند. در سایت این شرکت درباره سرویس AI خود به شرکت‌های بزرگ، Kubeflow را پلتفرم یادگیری ماشین مبتنی بر Kubernetes معرفی می‌کند.
* از جمله شرکت‌های بزرگ دیگر می‌توان به Arrikto اشاره کرد. این شرکت مستقر در امریکا در واقع برای یادگیری ماشین Kubeflow as a Service ارائه می‌دهد.
* Arrikto، برخلاف سایر شرکت‌های MLOps، تیم‌های MLOps را قادر می‌سازد تا مدل‌های یادگیری ماشین را ۳۰ برابر سریع‌تر از پلتفرم‌های سنتی ML به بازار عرضه کنند. Arrikto به‌عنوان یک مشارکت‌کننده پیشرو در Kubeflow، گردش‌های کاری خودکار، خطوط لوله قابل‌تکرار، استقرار مداوم از دسکتاپ به ابر و دسترسی ایمن به داده‌ها را ارائه می‌کند. Arrikto's Enterprise Kubeflow به‌عنوان یک توزیع چند گره در AWS، GCP و Azure در دسترس است و پلتفرم ترجیحی MLOps است که امروزه توسط بسیاری از شرکت‌های Fortune 500 در تولید استفاده می‌شود. این شرکت بیش از ۳۰۰ مشتری در ۱۷ کشور جهان دارد و توسط Unusual Ventures و Odyssey VP حمایت می‌شود.

# معماری و فنّاوری‌های مرتبط

## معماری مفهومی و منطقی

به‌طورکلی مراحل اصلی در معماری مفهومی MLOps عبارت‌اند از:

1. Data Preparation - آماده‌سازی داده‌ها:

این مرحله شامل جمع‌آوری، تمیزکردن و پیش‌پردازش داده‌هایی است که برای آموزش و ارزیابی مدل‌های ML استفاده می‌شود. آماده‌سازی داده‌ها ممکن است شامل کارهایی مانند عادی‌سازی داده‌ها، تشخیص نقاط پرت و مهندسی ویژگی‌ها باشد.

1. Model Selection - انتخاب مدل:

در این مرحله، دانشمندان و مهندسان داده، معماری، الگوریتم و فراپارامترهای مدل ML مناسب را بر اساس ویژگی‌های داده‌ها و مسئله‌ای که باید حل شود، انتخاب می‌کنند.

Model Training - آموزش مدل:

این مرحله شامل استفاده از داده‌های آماده شده برای آموزش مدل انتخاب شده، با تنظیم پارامترهای مدل برای به‌حداقل‌رساندن خطا بین خروجی‌های پیش‌بینی‌شده و خروجی‌های واقعی است.

1. Model Evaluation - ارزیابی مدل:

این مرحله شامل ارزیابی عملکرد مدل آموزش‌دیده با مقایسه پیش‌بینی‌های آن با مجموعه‌ای از نتایج شناخته‌شده (که داده‌های آزمایشی نیز نامیده می‌شود)، و تعیین میزان تعمیم مدل به داده‌های نادیده جدید است.

1. Model Deployment - استقرار مدل:

این مرحله شامل در دسترس قراردادن مدل برای کاربران نهایی است، خواه از طریق استقرار مدل در یک محیط تولید یا با ارائه یک API برای دسترسی دیگران به مدل باشد.

1. Model Monitoring and Maintenance - نظارت و نگهداری مدل:

این مرحله شامل نظارت بر عملکرد مدل مستقر در طول زمان و انجام به‌روزرسانی‌ها یا تنظیمات موردنیاز برای حفظ دقت و عملکرد مدل است.

Model Retainment- بازنشستگی مدل:

این مرحله شامل بازنشستگی مدل‌هایی است که دیگر موردنیاز نیستند یا دیگر عملکرد خوبی ندارند، تا اطمینان حاصل شود که فقط مدل‌هایی با عملکرد خوب به کار گرفته می‌شوند.

معماری منطقی MLOps معمولاً شامل اجزای زیر است:

* **مدیریت کد منبع:** این مؤلفه کد منبع مدل‌های یادگیری ماشین، الگوریتم‌ها و خطوط لوله را ذخیره می‌کند.
* **یکپارچه‌سازی مداوم/ استقرار مستمر (CI/CD**): این جزء ادغام و استقرار مدل‌های یادگیری ماشین را خودکار می‌کند.
* **آموزش مدل:** این جزء با استفاده از داده‌ها و الگوریتم‌های ذخیره شده در جزء مدیریت کد منبع، مدل‌های یادگیری ماشین را آموزش می‌دهد.
* **استقرار:** این جزء، مدل‌های آموزش‌دیده را در یک محیط تولید مستقر می‌کند، جایی که می‌توان از آنها برای پیش‌بینی استفاده کرد.
* **مانیتورینگ**: این مؤلفه عملکرد مدل‌های مستقر شده را نظارت می‌کند و در مورد دقت و قابلیت اطمینان آنها بازخورد ارائه می‌کند.
* **مدیریت مدل:** این مؤلفه چرخه عمر مدل‌های یادگیری ماشین را مدیریت می‌کند، از جمله نسخه‌سازی، به‌روزرسانی و بازنشستگی مدل‌ها در صورت لزوم. همچنین مدل‌های آموزش‌دیده و فراداده‌های مربوط به مدل‌ها مانند نسخه، دقت و داده‌های آموزشی را ذخیره می‌کند.

آماده‌سازی داده‌ها اولین مرحله در چرخه حیات یادگیری ماشینی (ML) است و شامل جمع‌آوری، تمیزکردن و پیش‌پردازش داده‌هایی است که برای آموزش و ارزیابی مدل‌های ML استفاده می‌شود. هدف از این مرحله آماده‌سازی داده‌ها در قالبی است که به‌راحتی توسط مدل ML مصرف شود و اطمینان حاصل شود که داده‌ها از کیفیت بالایی برخوردار هستند.

## Kubeflow

Kubeflow یک پلتفرم منبع‌باز برای توسعه، استقرار و مدیریت مدل‌های یادگیری ماشینی (ML) در Kubernetes (سیستم ارکستراسیون کانتینر) است به‌نحوی‌که توسعه، آزمایش و استقرار مدل‌های ML را در محیط‌های تولید برای سازمان‌ها آسان کرده و کارایی و مقیاس‌پذیری گردش‌های کاری ML خود را بهبود بخشند.

ویژگی‌های کلیدی Kubeflow عبارت‌اند از:

* یک رابط کاربرپسند برای آموزش و استقرار مدل‌های ML با استفاده از چارچوب‌های مطرح مانند TensorFlow، PyTorch و scikit-learn و ...
* پشتیبانی از آموزش توزیع‌شده مدل با استفاده از Kubernetes که امکان آموزش سریع‌تر و استفاده بهتر از منابع را فراهم می‌کند.
* ابزارهایی برای ثبت و تجسم پیشرفت آموزش مدل، و همچنین برای تنظیم هایپرپارامتر و استقرار مدل.
* ادغام با کتابخانه‌ها و ابزارهای معروف یادگیری ماشین، مانند TensorFlow Extended (TFX) و Seldon و ...
* توانایی اجرای در سیستم شخصی، در فضای ابری یا در محیط‌های ترکیبی.
* به هدف استقرار کلیه کامپوننت‌های چرخه یادگیری ماشین بر بستر کوبرنتیز ساخته شده است. ازاین‌رو می‌توان گفت یک مجموعه ابزار کاملاً منطبق و همگام با زیست‌بوم ابری است. همچنین برای CNCF و دریافت گواهینامه معتبر زیست‌بوم ابری این نهاد درخواست فرستاده است.

به‌طورکلی، Kubeflow یک پلتفرم قدرتمند و منعطف برای مدیریت چرخه یادگیری ماشین است که به دانشمندان و مهندسان داده اجازه می‌دهد تا با هم کار کنند تا مدل‌های یادگیری ماشینی خود را به شیوه‌ای کارآمدتر و مؤثرتر بسازند، مستقر کنند و مدیریت کنند.

## Kubeflow برای چه دسته مسائلی مناسب نیست؟

Kubeflow یک پلتفرم یادگیری ماشینی قدرتمند است که می‌تواند برای طیف وسیعی از وظایف مورداستفاده قرار گیرد. بااین‌حال، انواع مختلفی از مشکلات وجود دارد که ممکن است Kubeflow بهترین انتخاب برای آنها نباشد. توجه به این نکته مهم است که مناسب‌بودن Kubeflow برای یک کار خاص به نیازهای آن بستگی دارد. اگر مطمئن نیستید که Kubeflow گزینه مناسبی برای پروژه شماست، بهتر است با یک متخصص در زیرساخت یادگیری ماشین مشورت کنید. برخی مسائل که احتمالاً Kubeflow برای آنها مناسب نیست، در ذیل آمده است:

* **پروژه‌های در مقیاس کوچک:** اگر روی یک پروژه یادگیری ماشین کوچک کار می‌کنید، ممکن است Kubeflow زیاده‌روی کند. راه‌اندازی و مدیریت یک خوشه Kubeflow می‌تواند زمان بر باشد و ممکن است ارزش تلاش برای یک پروژه کوچک را نداشته باشد.
* **مدل‌های ساده:** اگر مدل شما ساده است و به منابع محاسباتی زیادی نیاز ندارد، ممکن است به مقیاس‌پذیری و انعطاف‌پذیری که Kubeflow ارائه می‌کند نیاز نداشته باشید.
* **یادگیری ماشینی غیر توزیعی:** اگر نیازی به اجرای حجم کار یادگیری ماشینی خود در یک خوشه توزیع شده ندارید، ممکن است ابزارهای ساده‌تر و سبک‌تری در دسترس باشند.
* **بارهای کاری غیر کانتینری:** Kubeflow برای کار با حجم کاری کانتینری طراحی شده است، بنابراین اگر حجم کاری شما کانتینری نباشد، ممکن است ادغام آن با Kubeflow دشوار باشد.
* **محیط‌های غیر کوبرنتیز:** Kubeflow بر روی Kubernetes ساخته شده است، بنابراین اگر از Kubernetes به‌عنوان پلتفرم ارکستراسیون خود استفاده نمی‌کنید، ممکن است بهترین انتخاب برای شما نباشد. به‌طورکلی، اگر روی یک پروژه یادگیری ماشینی پیچیده و در مقیاس بزرگ کار می‌کنید که به محاسبات توزیع شده و کانتینری نیاز دارد، Kubeflow یک انتخاب عالی است. بااین‌حال، اگر پروژه شما کوچک، ساده است یا به این ویژگی‌ها نیاز ندارد، ممکن است ابزارهای دیگری وجود داشته باشند که با نیازهای شما سازگارتر باشند.
* **یادگیری ماشین بی‌درنگ:** Kubeflow برای پردازش دسته‌ای طراحی شده است و ممکن است بهترین انتخاب برای برنامه‌های یادگیری ماشین بلادرنگ نباشد. برنامه‌های بلادرنگ نیاز به تأخیر کم و توان عملیاتی بالا دارند و Kubeflow ممکن است برای این نیازها بهینه نشده باشد.
* **بارهای کاری محاسباتی با کارایی بالا (HPC):** اگرچه Kubeflow را می‌توان برای برخی از بارهای کاری HPC استفاده کرد، اما ممکن است بهترین انتخاب برای همه انواع بارهای کاری HPC نباشد. Kubeflow برای کار با کلاسترهای توزیع شده طراحی شده است و ممکن است برای بارهای کاری HPC تک گره بهینه نشده باشد.
* **پژوهش در حوزه یادگیری ماشین:** Kubeflow در درجه اول برای تولید گردشکار یادگیری ماشین طراحی شده است و ممکن است بهترین انتخاب برای تحقیقات یادگیری ماشین نباشد. محققان اغلب به انعطاف‌پذیری و کنترل بیشتری بر محیط‌های یادگیری ماشینی خود نیاز دارند و Kubeflow ممکن است سطح سفارشی‌سازی موردنیاز را ارائه ندهد.
* **پیش‌پردازش داده:** Kubeflow بر آموزش و استقرار یادگیری ماشین متمرکز است و ممکن است بهترین انتخاب برای کارهای پیش‌پردازش داده نباشد. ابزارهای دیگری مانند Apache Spark و Apache Beam وجود دارند که برای پیش‌پردازش داده‌ها مناسب‌تر هستند.

## ویژگی‌های بستر Kubeflow

* **محیط MLOps مبتنی بر ابر**

Kubeflow محیط‌های توسعه مدل ML را بهینه‌سازی شده برای ابر ارائه می‌دهد و امکان پیوند مبتنی بر Kubernetes را با نرم‌افزارهای منبع‌باز مختلف فراهم می‌کند.

* **راحتی استفاده برای داده‌های بزرگ**

محیط‌های استاندارد از طیف وسیعی از چارچوب‌های یادگیری ماشین از TensorFlow، PyTorch، scikit-learn و Keras پشتیبانی می‌کنند. خط لوله برای کل فرایندهای توسعه، یادگیری و استقرار مدل‌های یادگیری ماشینی خودکار است تا از پیکربندی/ایجاد ساده و همچنین استفاده مجدد از مدل‌ها اطمینان حاصل شود.

* **ویژگی‌های افزودنی**

Kubeflow منبع‌باز گسترده‌ای از اجرای کار یادگیری توزیعی/نظارت تا مدیریت خدمات استنتاج/تحلیل و مدیریت صف شغل در هر پلتفرمی ارائه می‌دهد. کاربران همچنین می‌توانند از زمان‌بندی‌های کار (FIFO، Bin-packing و Gang-based)، نظارت بر منابع GPU، گزارش موتور Kubeflow و ویژگی‌های اضافی دیگری که معمولاً در نرم‌افزارهای منبع‌باز در دسترس نیستند، استفاده کنند.

* **سایر ویژگی‌ها**
* نوت‌بوک Jupyter (توسعه مدل، یادگیری و استنتاج)
* خودکارسازی گردش‌کار (بر اساس خطوط لوله یادگیری ماشین)
* مدیریت زمان‌بندی کار GPU و صف کار
* نظارت بر منابع GPU
* مانیتورینگ/ ثبت‌نام Kubeflow و اجرای/نظارت کار یادگیری توزیع شده
* ساخت و مدیریت چارچوب ML (TensorFlow، PyTorch و غیره)
* مدیریت/تجزیه‌وتحلیل خدمات استنتاج و مدیریت آزمایش‌های مدل/گره‌های یادگیری

به‌طورکلی، هر یک از این مراحل نقش مهمی در چرخه زندگی یادگیری ماشین ایفا می‌کند و برای سازمان‌ها مهم است که یک فرآیند MLOps قوی و کارآمد را برای مدیریت چرخه عمر ML از ابتدا تا انتها اجرا کنند.

MLOps (عملیات یادگیری ماشین) مجموعه‌ای از شیوه‌ها و ابزارهایی است که به سازمان‌ها کمک می‌کند تا چرخه زندگی یادگیری ماشین (ML) را به طور مؤثر مدیریت کنند. اجزای مفهومی MLOها به هم مرتبط هستند و برای دستیابی به هدف ارائه سیستم‌های ML قابل‌اعتماد، مقیاس‌پذیر و قابل نگهداری با هم کار می‌کنند. در زیر شرح مختصری از رابطه بین اجزای مفهومیMLOها ارائه شده است:

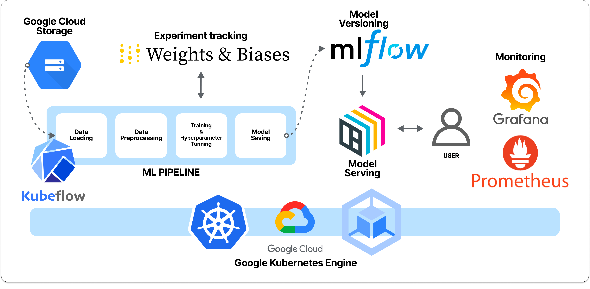
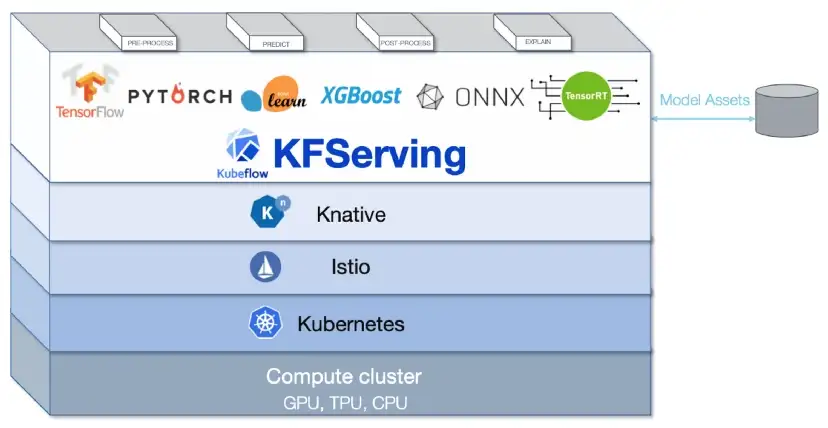
## معماری فیزیکی

تکمیل شده سند معماری فیزیکی در انتهای فاز یک تحویل می‌گردد.

## ساختار شماتیک بستر Kubeflow و تعامل با سایر سرویس‌ها در لایه فیزیکی

اجزای اصلی Kubeflow عبارت‌اند از:

* Kubernetes: Kubeflow مبتنی بر Kubernetes ساخته شده و یک پلتفرم مقیاس‌پذیر و قابل‌اعتماد برای استقرار و مدیریت کانتینرها فراهم می‌کند.
* خطوط لوله: پلتفرمی برای ساخت، استقرار و مدیریت گردش‌کار یادگیری ماشینی سرتاسری است. این یک رابط بصری برای ایجاد و مدیریت گردش‌های کاری پیچیده و همچنین ابزارهایی برای آماده‌سازی داده‌ها، آموزش مدل و استقرار فراهم می‌کند.
* Kubeflow Serving: Serving بستری برای استقرار و مدیریت مدل‌های یادگیری ماشین در تولید است. این یک پلتفرم مقیاس‌پذیر و قابل‌اعتماد برای ارائه مدل‌ها به‌عنوان میکروسرویس، با پشتیبانی از استنتاج بلادرنگ و دسته‌ای فراهم می‌کند.
* اپراتورهای آموزشی Kubeflow: اپراتورهای آموزشی Kubeflow راهی برای استقرار و مدیریت مشاغل آموزشی توزیع شده در Kubernetes ارائه می‌دهند. آنها به شما اجازه می‌دهند با استفاده از چارچوب‌های یادگیری ماشینی محبوب مانند TensorFlow، PyTorch و XGBoost، مدل‌ها را آموزش دهید.
* Kubeflow Metadata: Kubeflow Metadata راهی برای ردیابی و مدیریت فراداده‌های مرتبط با گردش کار یادگیری ماشین ارائه می‌دهد. این به شما امکان می‌دهد اطلاعات مربوط به داده‌های مورداستفاده برای آموزش، مدل‌های آموزش‌دیده و فراپارامترهای مورداستفاده را ذخیره کنید.
* نوت‌بوک‌های Kubeflow Jupyter: نوت‌بوک‌های Kubeflow Jupyter راهی برای اجرا و مدیریت نوت‌بوک‌های Jupyter در Kubernetes ارائه می‌دهند. این به شما امکان می‌دهد گردش‌های کاری یادگیری ماشین خود را در یک محیط آشنا توسعه و آزمایش کنید.
* Kubeflow Katib: Kubeflow Katib یک چارچوب تنظیم هایپرپارامتر است که به شما امکان می‌دهد فرآیند یافتن هایپرپارامترهای بهینه را برای مدل‌های یادگیری ماشین خود به طور خودکار انجام دهید. از انواع الگوریتم‌های تنظیم پشتیبانی می‌کند و می‌تواند با فریم‌ورک‌های معروف یادگیری ماشینی مانند TensorFlow، PyTorch و XGBoost استفاده شود.
* داشبورد Kubeflow: داشبورد Kubeflow یک رابط مبتنی بر وب برای مدیریت و نظارت بر استقرار Kubeflow شما ارائه می‌دهد. این یک نمای جامع از تمام اجزای موجود در استقرار شما، و همچنین معیارها و گزارش‌های بلادرنگ برای هر جزء ارائه می‌کند.



معماری فیزیکی MLOها به نیازهای خاص سازمان و پروژه‌های یادگیری ماشینی درحال‌توسعه بستگی دارد. زیرساخت باید مقیاس‌پذیر، انعطاف‌پذیر و قادر به رسیدگی به نیازهای جریان کار یادگیری ماشین و ادغام و استقرار مداوم مدل‌های یادگیری ماشین باشد. معماری فیزیکی MLOps به زیرساخت فیزیکی، سخت‌افزار و اجزای نرم‌افزاری مورداستفاده برای پیاده‌سازی معماری منطقی MLOها اشاره دارد. این می‌تواند شامل موارد زیر باشد:

**زیرساخت محاسباتی:** این می‌تواند شامل سرورهای داخلی، ماشین‌های مجازی مبتنی بر ابر یا ترکیبی از هر دو باشد.

**ذخیره‌سازی داده‌ها:** این می‌تواند شامل پایگاه‌های داده، دریاچه‌های داده یا سیستم‌های فایل مورداستفاده برای ذخیره داده‌های آموزشی، داده‌های آزمایشی و خروجی‌های مدل باشد.

**شبکه‌سازی:** این می‌تواند شامل اجزای سخت‌افزاری و نرم‌افزاری باشد که از ارتباط بین اجزای مختلف معماری MLOps پشتیبانی می‌کند.

**نظارت:** این می‌تواند شامل ابزارهای نظارت، تحلیلگرهای گزارش و داشبوردهایی باشد که برای نظارت بر عملکرد مدل‌های یادگیری ماشین و گردش کار MLOps استفاده می‌شوند.

**امنیت:** این می‌تواند شامل فایروال‌ها، سیستم‌های کنترل دسترسی، رمزگذاری و سایر اقدامات امنیتی برای محافظت از داده‌ها و مدل‌های حساس درگیر در MLOها باشد.

**ابزارها و پلتفرم‌ها:** این می‌تواند شامل ابزارها و پلتفرم‌های نرم‌افزاری منبع‌باز و تجاری باشد که برای یادگیری ماشین استفاده می‌شوند، مانند TensorFlow، PyTorch و غیره.

برخی ابزار/پلتفرم برتر MLOps جهت مدیریت چرخه یادگیری ماشین عبارتنداز:

* Amazon SageMaker
* Azure Machine Learning
* TensorFlow Extended (TFX)
* MLFlow
* Google Cloud ML Engine
* Data Version Control (DVC)
* H2O Driverless AI
* Kubeflow
* و ...

# ویژگی‌های کارکردی

Kubeflow یک پلتفرم منبع‌باز است که برای ساده‌سازی توسعه، استقرار و مدیریت گردش کار یادگیری ماشین در Kubernetes طراحی شده است. ویژگی‌های کاربردی را می‌توان به‌صورت زیر خلاصه کرد:

* **مدیریت داده‌ها شامل Data gathering و Data analysis:** MLOps ویژگی‌های مدیریت داده؛ مانند کنترل نسخه، خطوط لوله داده و پیش‌پردازش داده را ارائه می‌دهد که به شما امکان می‌دهد فرآیند مدیریت داده‌ها را ساده کنید و اطمینان حاصل کنید که داده‌های شما به‌درستی برچسب‌گذاری و ذخیره می‌شوند.
* **:Data transformation-preparation** با سیستم‌های ذخیره‌سازی داده‌های مختلف مانند ذخیره‌سازی اشیا، سیستم‌های فایل و پایگاه‌های داده ادغام می‌شود. این به کاربران اجازه می‌دهد تا به‌راحتی به داده‌های خود در محیط Kubeflow دسترسی داشته باشند و آنها را مدیریت کنند.
* **Offline/ Online feature store:**
* **سرویس‌دهی مدل:** MLOps یک زیرساخت سرویس‌دهی مدل مقیاس‌پذیر ارائه می‌دهد که به شما امکان می‌دهد مدل‌های یادگیری ماشین را به‌عنوان ریز سرویس‌ها در Kubernetes استقرار دهید. این ویژگی به شما امکان می‌دهد مدل‌های خود را از طریق REST API در معرض دید قرار دهید و ادغام آنها در برنامه‌های خود را آسان می‌کند.
* **تنظیم Hyperparameter**: MLOps مجموعه‌ای از ابزارها را برای خودکارسازی فرآیند تنظیم پارامترها ارائه می‌کند که به شما امکان م
* ی‌دهد مجموعه بهینه‌ای از پارامترها را برای مدل خود به‌سرعت پیدا کنید.
* **Model monitoring**: MLOps به شما امکان می‌دهد آزمایش‌های مختلف را ردیابی و مقایسه کنید و به شما امکان می‌دهد عملکرد مدل‌های خود را در طول زمان پیگیری کنید و تصمیم‌گیری آگاهانه در مورد اینکه کدام مدل‌ها را اجرا کنید، بگیرید. همچنین ابزارهای تجسم را برای کمک به شما در نظارت بر پیشرفت گردش کار یادگیری ماشینی، تجسم نتایج آزمایش‌های و اشکال‌زدایی مشکلات مدل‌های خود ارائه می‌دهد.
* **استقرار خودکار**  MLOpsمی‌تواند استقرار خطوط لوله یادگیری ماشین را با استفاده از ابزارهایی مانند Argo Workflows خودکار کند و کاربران را قادر می‌سازد تا به‌راحتی گردش‌های کاری یادگیری ماشینی پیچیده را ایجاد، اجرا و مدیریت کنند.
* **نسخه‌سازی مدل:** MLOps یک سیستم کنترل نسخه برای مدل‌های یادگیری ماشین ارائه می‌کند که به کاربران امکان می‌دهد نسخه‌های مختلف مدل‌های خود را پیگیری کنند و به‌راحتی بین آنها جابه‌جا شوند.

# ویژگی‌های غیرکارکردی

به‌طورکلی، ویژگی‌های غیرکاربردی Kubeflow برای اطمینان از اینکه جریان‌های کاری ML می‌توانند به شیوه‌ای قابل‌اعتماد، ایمن و مقیاس‌پذیر مستقر و مدیریت شوند، حیاتی هستند. Kubeflow با ارائه این ویژگی‌ها، سازمان‌ها را قادر می‌سازد تا بر توسعه و استقرار مدل‌های ML باکیفیت بالا بدون نگرانی در مورد زیرساخت‌های اساسی و پیچیدگی‌های عملیاتی تمرکز کنند. Kubeflow علاوه بر ارائه طیف گسترده‌ای از ویژگی‌های کاربردی، تعدادی ویژگی غیرعملکردی را نیز ارائه می‌کند که برای اطمینان از قابلیت اطمینان، مقیاس‌پذیری و امنیت گردش‌های کاری ML حیاتی هستند.

برخی از ویژگی‌های کلیدی غیرکاربردی Kubeflow عبارت‌اند از:

* **مقیاس‌پذیری:** به کاربران امکان می‌دهد تا بسته به تقاضا، جریان‌های کاری ML خود را افزایش یا کاهش دهند که کمک می‌کند تا اطمینان حاصل شود که گردش‌های کاری می‌توانند حجم زیادی از داده و قدرت پردازش را مدیریت کنند.
* **امنیت:** تعدادی از ویژگی‌های امنیتی، از جمله مکانیسم‌های احراز هویت و مجوز، رمزگذاری داده‌ها در حال انتقال و در حالت استراحت، و ذخیره امن اسرار و اعتبار را ارائه می‌دهد.
* **قابلیت اطمینان MLOps**: برای ارائه یک پلتفرم بسیار قابل‌اعتماد برای گردش کار ML طراحی شده است. این ویژگی‌هایی مانند تحمل خطا، خودترمیم و مقیاس خودکار را ارائه می‌دهد که به اطمینان از اینکه جریان‌های کاری ML می‌توانند بدون وقفه یا خرابی کار کنند کمک می‌کند.
* **قابلیت حمل:** کاربران می‌توانند به‌راحتی گردش کار ML خود را از یک محیط به محیط دیگر منتقل کنند. این امر از طریق استفاده از Containerization و Kubernetes به دست می‌آید که یک پلتفرم ثابت برای اجرای گردش‌های کاری ML بدون توجه به زیرساخت‌های اساسی فراهم می‌کند.
* **تحمل خطا:** MLOps از مکانیسم‌های تحمل خطا داخلی Kubernetes مانند کنترل‌کننده‌های تکرار و مجموعه‌های حالتی استفاده می‌کند تا اطمینان حاصل کند که بارهای کاری یادگیری ماشین در صورت خرابی گره یا مشکلات شبکه مختل نمی‌شود.
* **سفارشی‌سازی:** MLOps با قابل‌تنظیم با طیف وسیعی از APIها و نقاط افزونه به شما امکان می‌دهد عملکرد خود را با سیستم‌های موجود ادغام کنید. این به شما امکان می‌دهد پلتفرم را با نیازهای خاص خودتنظیم کنید و راه‌حل‌های سفارشی ایجاد کنید که نیازهای منحصربه‌فرد شما را برآورده کند.
* **مانیتورینگ****، نظارت و مدیریت لاگ‌ها:** مانیتورینگ فرآیند ردیابی عملکرد مدل‌های ML در تولید است. نظارت شامل جمع‌آوری داده‌ها در مورد پیش‌بینی‌های مدل، مقایسه آن‌ها با نتایج واقعی، و شناسایی هر گونه اختلاف یا خطا است. مانیتورینگ به شناسایی مشکلات احتمالی سیستم، مانند جابه‌جایی مدل کمک می‌کند و پاسخ سریع برای حفظ عملکرد سیستم را ممکن می‌سازد.

# فرضیات و محدودیت‌ها

باتوجه‌به محدودیت‌ها در موارد زیر، حداکثر متناسب با سخت‌افزار تأمین شده در فضای زیرساخت فضای Deployment تعهد از سمت پیمانکار صورت می‌گیرد که پس از تست در فضای stage قابلیت تخمین وجود خواهد داشت.

* تعداد پایپ لاین‌ها
* تعداد مدل‌ها
* انواع مدل‌ها مورد پشتیبانی
* نرخ داده ورودی مورداستفاده در یادگیری ماشین
* نرخ و تعداد سرو مدل
* نرخ استخراج و تعداد فیچرهای موجود در مخزن آنلاین و آفلاین

همچنین در تحویل دهی پروژه MLOPs،‌ محدودیت‌های زیرساختی زیر لحاظ خواهند شد:

* حداکثر تعداد کلاستر
* حداکثر تعداد پاد
* حداکثر تعداد مدل برای یادگیری
* حداکثر تعداد مدل برای استنتاج
* حداکثر تعداد درخواست

## مشکل مقیاس‌پذیری استقرار مدل

## درحالی‌که با ساختن مدل‌هایی برای هر مورداستفاده از دقت استنتاج و حفظ حریم خصوصی داده بهتر بهره می‌برید، استقرار هزاران تا صدها هزار مدل در یک خوشه Kubernetes چالش‌برانگیزتر است. علاوه بر این، تعداد فزاینده‌ای از موارد استفاده از سرویس‌دهی مدل‌های مبتنی بر شبکه عصبی وجود دارد. برای دستیابی به تأخیر معقول، این مدل‌ها بهتر روی پردازنده‌های گرافیکی ارائه می‌شوند. بااین‌حال، ازآنجایی‌که پردازنده‌های گرافیکی منابع گران قیمتی هستند، سرویس‌دهی به بسیاری از مدل‌های مبتنی بر GPU پرهزینه است.طراحی اصلی KFServing یک مدل را برای هر InferenceService مستقر می‌کند. اما، وقتی با تعداد زیادی مدل سروکار داریم، پارادایم «یک مدل، یک سرور» چالش‌هایی را برای یک خوشه Kubernetes ایجاد می‌کند. برای مقیاس‌بندی تعداد مدل‌ها، باید تعداد InferenceServices را مقیاس کنیم، چیزی که می‌تواند به‌سرعت محدودیت‌های خوشه را به چالش بکشد. سرویس چند مدل برای رسیدگی به سه نوع محدودیت طراحی شده است که KFServing با آن مواجه خواهد شد: ۱) محدودیت منابع محاسبه، ۲) حداکثر محدودیت pods، 3) حداکثر محدودیت آدرس IP.

## محاسبه محدودیت منابع

هر InferenceService یک سربار منبع دارد، به دلیل اینکه در هر پاد کارهای جانبی تزریق می‌شود. این معمولاً حدود ۰٫۵ CPU و منبع حافظه ۰٫۵ G را به‌ازای هر نسخه InferenceService اضافه می‌کند. به‌عنوان‌مثال، اگر ۱۰ مدل را مستقر کنیم، هر کدام با ۲ کپی، سربار منبع ۱۰ \* ۲ \* ۰٫۵ = ۱۰ CPU و 10 \* 2 \* ۰٫۵ = 10 گیگابایت حافظه است. سربار منبع هر مدل 1 CPU و 1 گیگابایت حافظه است. استقرار بسیاری از مدل‌ها با استفاده از رویکرد فعلی، به‌سرعت از منابع محاسباتی خوشه استفاده می‌کند. با سرویس چند مدل، این مدل‌ها را می‌توان در یک InferenceService بارگذاری کرد، سپس میانگین سربار هر مدل 0.1 CPU و 0.1 گیگابایت حافظه است. برای مدل‌های مبتنی بر GPU، تعداد GPUهای موردنیاز به‌صورت خطی با افزایش تعداد مدل‌ها افزایش می‌یابد که مقرون‌به‌صرفه نیست. اگر بتوان چندین مدل را در یک سرور مدل مجهز به GPU مانند TritonServer بارگذاری کرد، به GPUهای بسیار کمتری در کلاستر نیاز داریم.

## حداکثر محدودیت Pod

Kubelet دارای حداکثر تعداد پادها در هر گره با محدودیت پیش‌فرض روی ۱۱۰ است. طبق بهترین روش Kubernetes، یک گره نباید بیش از ۱۰۰ پاد را اجرا کند.

با این محدودیت، یک خوشه معمولی ۵۰ گره با محدودیت Pod پیش‌فرض می‌تواند حداکثر 1000 مدل را اجرا کند با این فرض که هر InferenceService به طور متوسط دارای 4 Pod است (دو کپی ترانسفورماتور و دو نسخه پیش‌بینی‌کننده).

## حداکثر محدودیت آدرس IP

خوشه‌های Kubernetes همچنین دارای محدودیت آدرس IP در هر خوشه هستند. هر پاد در InferenceService به یک IP مستقل نیاز دارد. به‌عنوان‌مثال، یک خوشه با ۴۰۹۶ آدرس IP می‌تواند حداکثر 1024 مدل را با فرض اینکه هر InferenceService به طور متوسط دارای 4 پاد باشد (دو ماکت ترانسفورماتور و دو نسخه پیش‌بینی‌کننده) استقرار دهد.

## مزایای استفاده از سرویس چند مدل

ویژگی سرویس‌دهی چند مدل برای رفع سه محدودیت بالا طراحی شده است. میانگین سربار منابع در هر مدل را کاهش می‌دهد؛ بنابراین استقرار مدل مقرون‌به‌صرفه‌تر می‌شود و تعداد مدل‌هایی که می‌توانند در یک خوشه مستقر شوند دیگر با حداکثر محدودیت pods و حداکثر محدودیت آدرس IP محدود نخواهد شد.

# تست و ارزیابی MLOPs

تکمیل شده سناریوی تست ویژگی‌های کارکردی و غیرکارکردی در انتهای فاز یک تحویل می‌گردد.

## ارزیابی واحد، تجمیعی و کارایی بخش‌های کارکردی MLOPs

نکته: ممکن است در ابزارهای اعلان شده در سند معماری تغییراتی بنابر نیاز پروژه حاصل شود. به‌عنوان مثال، در بخش feature store از ابزار دیگری به جز Feast استفاده شود.

**Unit test**

در این تست دو مرحله کلی pipeline و serving وجود دارد:

* Pipeline: تمام مراحل یادگیری و در بعضی مواقع ارزیابی‌های مدل صورت می‌گیرد در مرحله اول مدل طراحی می‌شود. در مثال فعلی سه بخش داریم preprocess, model\_fit و evaluation که در بخش preprocess داده‌ها از دیتاست یا دیتابیس دریافت، آماده‌سازی و به داده‌های آموزش و آزمون تقسیم و در نهایت در data wherhouse/data lake ذخیره می‌شوند. تمامی مراحل در یک تابع انجام می‌شود. ازآنجایی‌که هیچ ورودی و خروجی نداریم و تمام دیتاها در پایگاه‌داده ذخیره می‌شوند، برای آزمایش این مؤلفه با چک‌کردن دیتای موجود آزمایش می‌کنیم که خروجی‌ها وجود دارند یا خیر.

باتوجه‌به اینکه هر مؤلفه یک تابع است، ما برای هر کدام یک unit test طراحی می‌کنیم و در کنار pipeline در هنگام آموزش یا اجرای experiment آنها آزمایش‌های لازم را از توابع می‌گیرند و برای هر یک از مولفه‌های موجود در pipeline تست به‌صورت جداگانه نوشته می‌شود و نتایج هر تست نیز در database ذخیره می‌شود.

* Serving: در این مرحله آزمون برای هر کدام از متدهای کلاس استنتاج می‌باشد. کلاس استنتاج دارای دو متد اصلی load data و predict می‌باشد. برای هر کدام از این بخش‌ها unit test جداگانه طراحی می‌شود تا عملکرد استنتاج را ارزیابی کند. برای load data عملکرد به این گونه است که آیا مدل واقعاً در مرحله استنتاج کار می‌کند یا خیر. برای predict نیز یک ورودی به‌صورت آزمایشی به مدل داده شده و خروجی موردنظر بررسی خواهد شد. نتایج این آزمایش‌ها نیز در database ذخیره می‌شود.
* Feast:در این ارزیابی روند اجرای جریان داده و اتصال بخش‌های مختلف Feast چک خواهد شد.

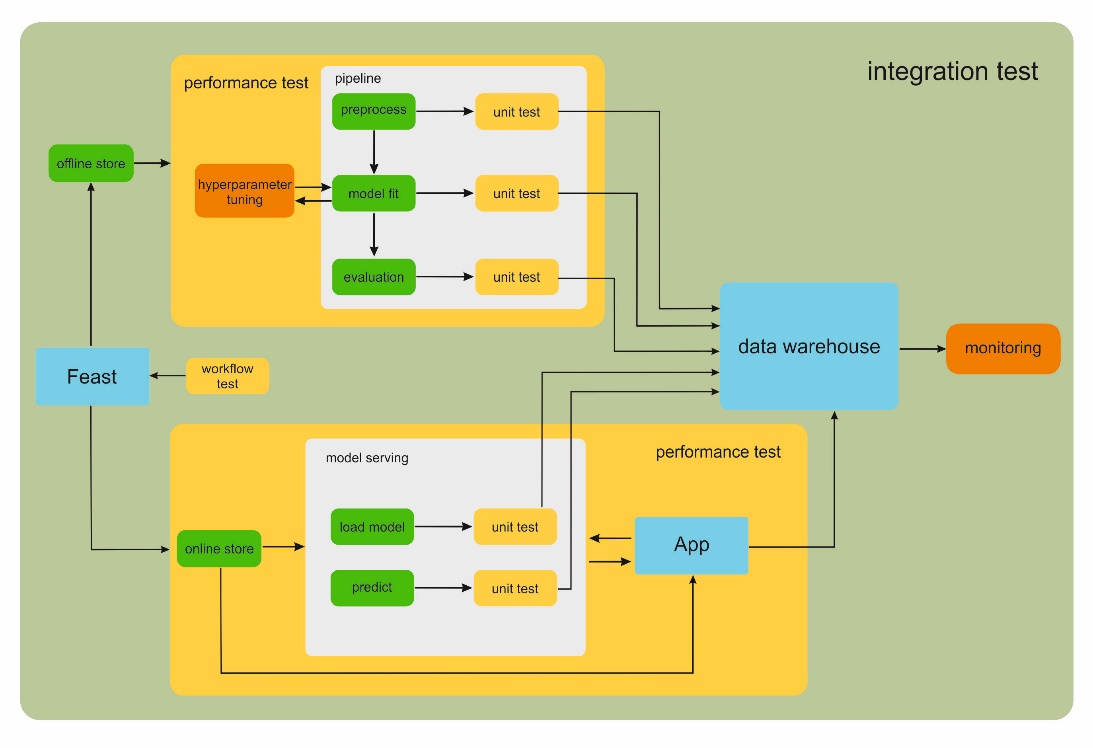
**Performance test**

در بخش ارزیابی عملکرد نیز ما دو مرحله کلی داریم که کاملاً مجزا عمل می‌کنند:

* Pipeline: همان‌طور که در شکل مشاهده می‌کنید perforamnce test در این بخش عملکرد و همچنین مدت‌زمان اجرای مدل را آزمایش می‌کند. به این‌گونه که مدل، داده‌های موردنیاز را دریافت کرده و pipeline شروع به اجرا می‌کند در مرحله model fit بهترین پارامترها به‌وسیله hyperparamet tuning انتخاب می‌شود و مدل فرایند آموزش خود را به اتمام می‌رساند. در این فرایند بررسی‌کردن آنکه باید بهترین خروجی ممکن از مدل دریافت شود و بالاترین کارایی را داشته باشد، وظیفه perfomance test می‌باشد.
* Serving: در این استنتاج همان‌طور که مشاهده می‌کنید سه بخش کلی داریم. بخش اول online store، بخش دوم که serving نهایی و بخش سوم که یک app برای serve کردن مدل نهایی تعبیه شده است. این آزمون به این گونه عمل می‌کند که داده‌های آنلاین از Feast دریافت شده و به دو بخش app و serving ارسال می‌شود. داده‌های online کاملاً دارای برچسب یا جواب صحیح می‌باشد. وظیفه app درخواست از serving برای پاسخ داده‌ها می‌باشد و خروجی نهایی را با خروجی واقعی مقایسه کرده و کارایی مدل نهایی را مورد ارزیابی قرار می‌دهد. این امر باعث می‌شود تا مدل همواره در حال performance test باشد.

**Integration test**

در این بخش آزمایش کلی برای تمام بخش‌های MLops اعمال خواهد شد. برای integration test ابتدا شروع خط لوله و اتمام آن را مورد ارزیابی قرار می‌دهیم. به این شکل که کل pipeline اجرا شده دیتاهای لازم در data wherhouse/data lake ذخیره شوند و ارزیابی مدل صورت‌گرفته باشد.



# آموزش و تحویل

**شرایط، آزمون و تحویل گیری**

1. پیمانکار موظف به ارائه طرح آزمون پروژه است. این طرح آزمون معرف مجموعه آزمون‌هایی است که پیمانکار اعلام می‌نماید که خروجی خود را با آن تست کرده و موظف است نتایج تست را ارائه نماید. شایان‌ذکر است که در مراحل اجرایی قرارداد، نتایج حاصل از طرح آزمون می‌بایست همراه محصول تحویل کارفرما گردد. طرح آزمون پروژه باید به تأیید ناظر کارفرما برسد.
2. لازم به ذکر است که در صورت تست سناریو اعلامی توسط کارفرما و عدم دستیابی به نتایج اعلام شده، محصول/خروجی معیوب تلقی شده و پیمانکار موظف است در زمان پروژه، عیوب را رفع کرده تا در نهایت، به خروجی مدنظر دست یابند.
3. مراحل آزمون و تحویل اولیه تجهیزات/سامانه‌ها در محل پیمانکار یا شرکت ثالث مورد تأیید کارفرما و مطابق با طرح آزمون ارائه شده و با هزینه پیمانکار انجام خواهد شد. تأمین کلیه دستگاه‌ها/سامانه‌های اندازه‌گیری و آزمون بر عهده پیمانکار است.

**نصب، راه‌اندازی و تحویل سیستم**

پیمانکار موظف است خروجی‌های هر فاز از سیستم را جداگانه راه‌اندازی، تحویل و آموزش دهد و تأیید نهایی خروجی هر فاز منوط به تأیید عملکرد آن فاز خواهد بود.

* تبصره: تحویل‌گیری نهایی سامانه منوط به یکپارچگی بخش‌های مختلف و تأیید کارکرد نهایی سامانه است.
* ارائه مستنداتی پیرامون مراحل نصب و راه‌اندازی کلیه مولفه‌های اصلی بستر Mlops
* به‌روزرسانی مستندات به‌ازای هر نسخه از مولفه‌های اصلی بستر Mlops
* کلیه مراحل مندرج در مستندات توسط تیم سحاب با حضور نماینده شرکت آدین، اجرا و صحت سنجی خواهد شد.

**آموزش**

شرکت آدین موظف به برگزاری دوره آموزشی با شرایط ذیل است:‌

1. آموزش جامع بهره‌برداری و استفاده از محصول پروژه (مواردی همچون نحوه نصب، راه‌اندازی، نگهداری، توسعه و نیز بهره‌برداری برای تجهیزاتی/سامانه‌هایی که نیاز به آموزش دارند) می‌بایست حتماً در طرح آموزش پروپوزال دیده شود.
2. شرکت آدین موظف به تحویل سری کامل از مستندات آموزشی تمامی اجزاء سیستم با جزئیات کامل به‌صورت نسخه چاپی (Hard Copy) و نسخه الکترونیکی (Soft Copy-CD)، به تعداد شرکت‌کنندگان دوره آموزشی است.
3. آموزش‌های راهبری و کاربری برای کارشناسان ذی‌ربط در محل مورد تأیید نماینده فنی سحاب در خصوص تکمیل فاز آموزش، باید به‌صورت کارگاه و یا کلاس آموزشی ارائه گردد. آموزش می‌بایست در دو سطح اپراتوری و مدیر سیستم برای حداقل ۱۰۰ نفر - ساعت برگزار شود.
4. برگزاری جلسات آموزش مفاهیم کلی و نصب و مدیریت بستر Mlops
5. تحویل تمامی مستندات مربوط به فاز توسعه
6. تأییدیه نماینده فنی سحاب در خصوص تأیید صحت مستندات تولید شده

**فهرست مستندات قابل‌تحویل**

تحویل‌دادنی‌های مرسوم به‌عنوان مستندات، در مستند ضمائم پروژه به طور کامل ذکر شده است. مستندات و قالب تهیه آن می‌بایست مورد تأیید کارفرما قرار گیرد. برخی از مستندات مهم به شرح زیر است:

* مستند نیازمندی و معماری
* مستند راه‌اندازی و استقرار
* مستند طرح آزمون
* مستند نحوه استفاده

# منابع موردنیاز

منابع موردنیاز در Production به‌گونه‌ای در نظر گرفته شود که برای حل مسئله‌ای با ابعاد داده‌ای حداکثر ۱۰ برابر مسئله کشف رفتارهای مخاطره‌آمیز در دفاتر نمایندگی اپراتورهای همراه مناسب باشد. با توجه به توافقات صورت‌گرفته در جلسات حضوری، منابع Production توسط آدین متناسب با مقیاس ۱۰ برابری تعیین‌شده، در ذیل پیشنهاد شده است که ابعاد Stage نیز معادل ۲۰ درصد محیط Production توسط کارفرما فراهم خواهد شد.

* لازم به ذکر است که تعداد GPU در فضای Stage حداقل ۲ لحاظ گردد. همچنین حداقل ۳۰ درصد هارد از نوع SSD و مابقی SAS در نظر گرفته شود.
* **منابع پیشنهادی در محیط Production**

|  |  |
| --- | --- |
| مشخصات | تعداد سرور |
| 128 GB RAM | ۲۰ |
| 4 TB HDD | ۲۰ |
| 40 Core CPU | ۲۰ |
| GPU, RTX 4090 | ۴ |

* **منابع پیشنهادی در محیط Stage**

|  |  |
| --- | --- |
| مشخصات | تعداد سرور |
| 128 GB RAM | ۴ |
| 4 TB HDD | ۴ |
| 40 Core CPU | ۴ |
| GPU, RTX 4090 | ۲ |

# زمان‌بندی و شکست کارها

مراحل اجرای Kubeflow معمولاً شامل موارد زیر است:

* نصب: Kubeflow را روی زیرساخت هدف، از جمله خوشه‌های Kubernetes و هر گونه وابستگی لازم، نصب و پیکربندی کنید.
* آماده‌سازی داده‌ها: داده‌ها را جهت استفاده در Kubeflow آماده و سازماندهی کنید، از جمله داده‌ها، پاک‌سازی و عادی‌سازی.
* توسعه مدل: مدل‌های یادگیری ماشین را با استفاده از ابزارهای داخلی Kubeflow یا ادغام با سایر چارچوب‌های ML مانند TensorFlow، PyTorch و Jupyter توسعه دهید.
* آموزش مدل: مدل‌ها را روی داده‌های مقیاس بزرگ با استفاده از قابلیت‌های کوچک‌سازی Kubeflow، از جمله آموزش توزیع‌شده و شتاب GPU، آموزش دهید.
* استقرار مدل: با استفاده از ابزارهای استقرار Kubeflow، از جمله TensorFlow Serving، Seldon، و KFServing، مدل‌ها را جهت تولید مستقر کنید.
* نظارت و نگهداری: مدل‌های در حال تولید، از جمله عملکرد، دقت و جابه‌جایی داده‌ها را پایش کنید و مدل‌ها را در صورت لزوم به‌روزرسانی کنید.
* یکپارچه‌سازی مداوم و استقرار مستمر (CI/CD): فرآیند یکپارچه‌سازی، آزمایش و استقرار مدل‌ها را با استفاده از ابزارهایی مانند جنکینز یا GitLab به‌صورت خودکار انجام دهید.

# گانت چارت کلی پروژه

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **کلیات هر فاز** | فاز یک  (اردیبهشت - خرداد) | | فاز دو  (تیر - مرداد - شهریور) | | | فاز سه  (مهر - آبان - آذر) | | | فاز چهار  (دی - بهمن - اسفند) | | | فاز پنج  (فروردین - اردیبهشت - خرداد) | | |
|  | **اردیبهشت** | **خرداد** | **تیر** | **مرداد** | **شهریور** | **مهر** | **آبان** | **آذر** | **دی** | **بهمن** | **اسفند** | **فروردین** | **اردیبهشت** | **خرداد** |
| **استقرار kubeflow بر روی بستر کوبرنتیز به همراه ماژول‌های ML** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **تعریف دقیق صورت‌مسئله شاهکار** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **آماده‌سازی داده‌ها مسئله اول** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **توسعه مدل مسئله اول** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **طراحی آزمون** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **ایجاد پایپ لاین CI/CD یادگیری مدل مسئله** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **استنتاج مدل در داشبورد Kubeflow** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **مانیتورینگ و نظارت داشبورد Kubeflow** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Off-line feature store** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **On-line feature store** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **مستندسازی مسئله اول** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **تعریف دقیق صورت ‌مسئله دوم** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **آماده‌سازی داده‌ها مسئله دوم** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **توسعه مدل مسئله دوم** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **ایجاد پایپ لاین CI/CD یادگیری مدل مسئله** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **استنتاج مدل در داشبورد Kubeflow** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **مانیتورینگ و نظارت داشبورد Kubeflow** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Off-line feature store** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **On-line feature store** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **مستندسازی** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

# شکست کار پروژه

**جدول شکست کار پروژه**

| WBS | | شرح | | مدت‌زمان (ماه) | | پیش‌نیاز | | مسئول |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| فاز اول | | | | | | | | |
| ۱.۱ | تدوین سند نیازمندی | | ۲ | |  | | پیمانکار | |
| ۱.۲ | ارائه نیازمندی سخت‌افزار موردنیاز آزمایشگاه کارفرما | |  | | پیمانکار | |
| ۱.۳ | تحویل نمونه داده مسئله شاهکار | |  | | کارفرما | |
| ۱.۴ | ارائه سند معماری سامانه | |  | | پیمانکار | |
| ۱.۵ | تأمین سخت‌افزار بستر stage | | ۱.۱و ۱.۴ | | کارفرما | |
| ۱.۶ | تعریف دقیق صورت‌مسئله شاهکار | | ۱.۳ | | پیمانکار | |
| ۱.۷ | استقرار kubeflow بر روی بستر کوبرنتیز به همراه ماژول‌های ML | | ۱.۵ | | پیمانکار | |
| ۱.۸ | آماده‌سازی داده‌ها | | ۱.۳ | | پیمانکار | |
| ۱.۹ | توسعه مدل مسئله شاهکار | | ۱.۸ | | پیمانکار | |
| ۱.۱۰ | ارائه سند طراحی آزمون | |  | | پیمانکار | |
| فاز دوم | | | | | | | | |
| ۲.۱ | ایجاد پایپ لاین CI/CD یادگیری مدل مسئله شاهکار | | ۳ | | ۱.۸ | | پیمانکار | |
| ۲.۲ | استقرار ماژول Kserve | |  | |  | |
| ۲.۳ | استنتاج مدل مسئله شاهکار در داشبورد Kubeflow با استفاده از کامپوننت Kserve | | 2.۲ | | پیمانکار | |
| ۲.۴ | تعریف مسئله دوم ML | |  | | کارفرما | |
| ۲.۵ | تحویل داده مسئله دوم ML | |  | | کارفرما | |
| ۲.۶ | تعریف دقیق صورت‌مسئله دوم ML | | ۲.۴ | | پیمانکار | |
| فاز سوم | | | | | | | | |
| ۳.۱ | مانیتورینگ و نظارت داشبورد Kubeflow | | ۳ | | ۲.۱ | | پیمانکار | |
| ۳.۲ | آماده‌سازی داده‌ها | | ۲.۲ | | پیمانکار | |
| 3.۴ | استقرار ماژول موردنیاز feature store | | ۳.۲ | |  | |
| 3.۵ | Off-line feature store | | ۳.۴ | | پیمانکار | |
| ۳.۶ | توسعه مدل مسئله دوم ML | | ۲.۵ | | پیمانکار | |
| فاز چهارم | | | | | | | | |
| ۴.۱ | On-line feature store | | ۳ | | ۳.۲ | | پیمانکار | |
| ۴.۲ | ارائه سند نصب و استقرار مسئله اول | | ۴.۱ | | پیمانکار | |
| ۴.۳ | مستندات نحوه استفاده مسئله اول | | ۴.۲ | | پیمانکار | |
| ۴.۴ | اجرای آزمون‌های طرح آزمون | | ۴.۲ | | پیمانکار | |
| ۴.۵ | ایجاد پایپ لاین CI/CD یادگیری مدل مسئله دوم | | ۳.۵ | | پیمانکار | |
| ۴.۶ | استنتاج مدل مسئله دوم در داشبورد Kubeflow با استفاده از کامپوننت Kserve | | ۴.۳ | | پیمانکار | |
| ۴.۷ | انتقال مسئله اول به فضای Production | | ۴.۴ | | پیمانکار | |
| فاز پنجم | | | | | | | | |
| ۵.۱ | مانیتورینگ و نظارت داشبورد Kubeflow | | ۳ | | ۴.۴ | | پیمانکار | |
| ۵.۲ | Off-line feature store | | ۴.۴ | | پیمانکار | |
| ۵.۳ | On-line feature store | | ۴.۴ | | پیمانکار | |
| ۵.۴ | اجرای آزمون‌های طرح آزمون مسئله دوم | | ۵.۳ و ۴.۲ | | پیمانکار | |
| ۵.۵ | ارائه سند نصب و استقرار مسئله دوم | | ۵.۴ | | پیمانکار | |
| ۵.۶ | مستندات نحوه استفاده مسئله دوم | | ۵.۴ | | پیمانکار | |
| ۵.۷ | انتقال مسئله دوم به فضای Production | | ۵.۴ | | پیمانکار | |

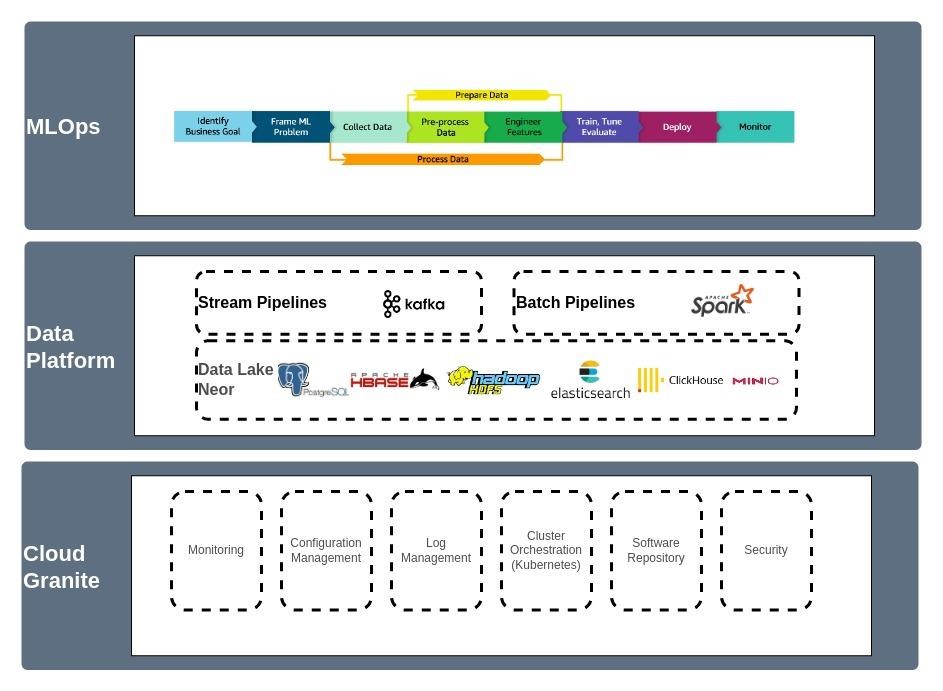
**پیوست نیازمندی فنی**

# مقدمه

هدف از تدوین این سند ایجاد دیدی شفاف و صریح بین کارفرما و پیمانکار در توسعه محصول زیرساخت MLOps می‌باشد. در این سند، ابتدا به بیان محدوده و دامنه مورد انتظار از این محصول خواهیم پرداخت، در ادامه و نیازمندی‌های مورد انتظار در زمینه توسعه و استقرا محصول MLOps شرح داده خواهد شد.

# محدوده و دامنه محصول

در حال حاضر برخی از محصولات از جمله دو محصول کلود گرانیت و دیتا پلتفرم توسط تیم فنی شرکت سحاب توسعه داده شده یا درحال‌توسعه می‌باشند؛ بنابراین این انتظار وجود دارد که محصول MLOps از طرفی سازگاری کافی با این دو پلتفرم را داشته باشد و از طرفی نیاز است که از سرویس و خدمات آنها به‌خوبی بهره بگیرد. برای روشن‌شدن موضوع به‌اختصار به بیان ویژگی‌ها و دامنه این محصولات خواهیم پرداخت. در زیر شمای کلی از محصولات نمایش‌داده‌شده است.



زیرساخت رایانش ابری گرانیت

این پلتفرم وظیفه ایجاد بستر ابری را بر عهده دارد و برای این موضوع سرویس‌های مختلفی مانند مانیتورینگ، مدیریت لاگ و زیرساخت رایانش ابری (Kubernetes)را ارائه می‌کند؛ بنابراین محصول MLOps باید با این زیرساخت سازگار بوده و از سرویس‌های ارائه شده توسط آن استفاده نماید.

دیتا پلتفرم

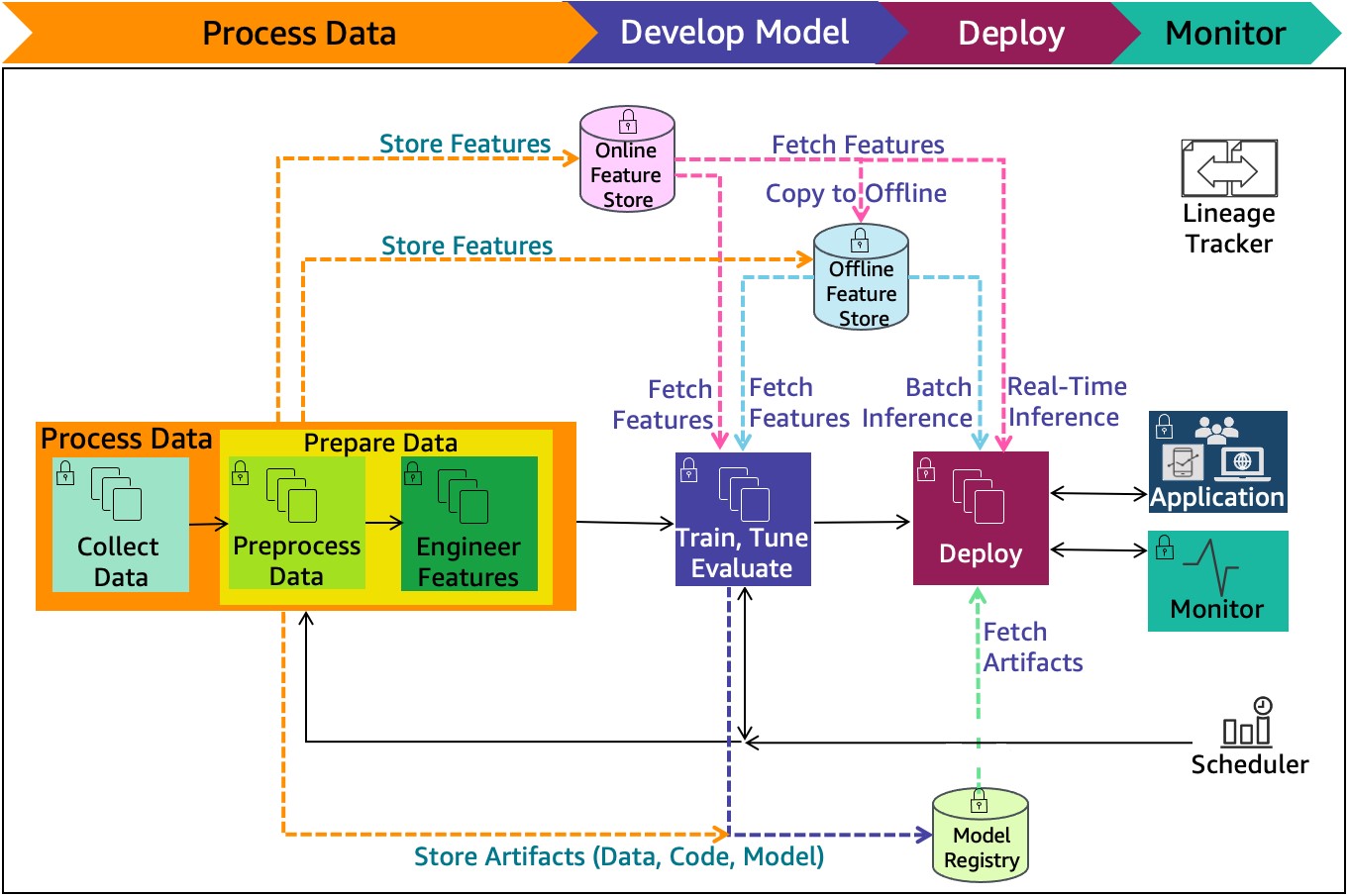
این بستر از سه بخش اصلی زیر تشکیل شده است:

* دریاچه داده: این بخش وظیفه ارائه بستری برای ذخیره‌سازی داده با ویژگی‌های مختلف و برای استفاده‌های متفاوت را بر عهده دارد.
* پردازش جریانی: این بستر امکان پردازش جریانی بر روی داده با امکان تعامل با دریاچه داده را در اختیار کاربر قرار می‌دهد
* پردازش دسته‌ای: این بستر امکان پردازش دسته‌ای بر روی داده ذخیره شده در دریاچه داده را بر عهده دارد.

سازگاری و تجمیع محصول زیرساخت MLOps با این پلتفرم اهمیت بسیار بالایی دارد و در توسعه آن دقت به این موضوع نقش کلیدی برای کارفرما دارد.

MLOps محصول

محصول MLOps ابزاری است که می‌تواند در جمع‌آوری داده، ذخیره‌سازی داده، ویژگی (Feature)و مدل، بهبود داده، مهندسی نیازمندی (Requirement Engineering)، مهندسی ویژگی (Feature Engineering)، مهندسی داده (DataEngineering)، مهندسی مدل (Model Engineering)، تست و اعتبارسنجی مدل، نصب و راه‌اندازی مدل، جریان CI/CD مدل، داده و کد، نظارت و هشداردهی (Monitoring & Triggering) و ارائه بستر پردازشی و ذخیره‌سازی متمرکز ایفای نقش کند و فرایند توسعه موتورهای تحلیلی را به‌مراتب ساده‌تر و بهینه‌تر کند. شکل زیر معماری کلی زیرساخت MLOps حاوی مؤلفه‌های اصلی و ارتباطات بین آنها است.



مؤلفه‌های این معماری در ادامه شرح داده شده است:

**پردازش داده (process data**) در این مؤلفه کتابخانه‌هایی در جهت استخراج داده از منابع مختلف، استخراج مشخصات آماری داده، تمیزسازی داده و ایجاد فیچرها ارائه می‌شود.

**انباره داده آنلاین**/**آفلاین فیچرها (online/offline store**) با ذخیره‌کردن فیچرها، محاسبات تکراری آنها در بخش‌های مختلف سازمان حذف می‌شود. انباره داده آنلاین در جهت دریافت سریع فیچرها برای استفاده در مرحله استنتاج به کار می‌رود. انباره داده آفلاین، تاریخچه مقادیر فیچرها را نگهداری می‌کند و در مرحله یادگیری مدل مورداستفاده قرار می‌گیرد. برای این مؤلفه از ابزارهای موجود در دریاچه داده استفاده می‌شود و در صورت نیاز به ابزاری که توسط این زیرساخت ارائه نمی‌شود برای انتخاب آن نیاز به هماهنگی با تیم کارفرما وجود دارد.

**توسعه مدل(train/tune/evaluate**) این مؤلفه بستری برای توسعه مدل از طریق آزمایش‌ها مختلف را فراهم می‌کند. این مؤلفه شامل نوت‌بوک‌ها و کتابخانه‌های مرسوم برای توسعه و ارزیابی مدل و کتابخانه‌هایی برای تنظیم پارامترها می‌باشد.

**رجیستری مدل(model registry**) رجیستری مدل یک مخزن برای ذخیره مدل‌های یادگیری ماشین و فراداده‌های مرتبط است. در این رجیستری، نسخه‌های مختلف داده و مدل قرار می‌گیرد.

**مانیتورینگ (monitoring**) این مؤلفه وظیفه نظارت بر مدل و تشخیص مشکلات را بر عهده دارد. مشکلات می‌تواند مرتبط باکیفیت داده، کیفیت مدل و دریفت باشد.

**استقرار (deploy**) این مؤلفه وظیفه استقرار مدل در محیط عملیاتی و سرو آن را بر عهده دارد. این مؤلفه عملیاتی مانند مقیاس‌پذیری متناسب با بار ورودی و استقرار قناری را پشتیبانی می‌کند.

**زمان‌بند (scheduler**) این مؤلفه می‌تواند فرایند یادگیری مجدد را در بازه‌های زمانی مشخص اجرا کند.

**ردیاب جامع(lineage tracker**) این مؤلفه بستری برای ثبت و ضبط اطلاعات تمام منابع در هر نقطه از زمان را ارائه می‌کند. این اطلاعات می‌تواند شامل نسخه کد، داده، فیچرها، مدل و نتایج مربوطه باشد.

# نیازمندی‌ها فنی در توسعه و استقرار محصول

1. پروژه در دو محیط آزمایشگاه و عملیاتی مستقر خواهد شد. به‌منظور اینکه تیم پیمانکار بتواند استقلال موردنظر خود را داشته باشد، راه‌اندازی آزمایشگاه که شامل مولفههای زیرساختی است و نصب و راه‌اندازی پروژه در محیط آزمایشگاه کاملاً بر عهده تیم پیمانکار می‌باشد. در محیط عملیاتی، زیرساخت ابری پروژه شامل کوبرنیتیز، پایگاه‌داده‌ها و ابزارهای زیرساختی مانند ابزار مانیتورینگ و مدیریت لاگ توسط تیم کارفرما نصب و راه‌اندازی خواهد شد؛ بنابراین فرایند نصب و اعمال تمامی مولفههای محصول MLOps، پلاگین‌ها، تنظیمات باید قابل‌تکرار در محیط‌های مختلف بوده و بستری برای نصب مجدد پروژه در محیط دلخواه ارائه شود تا بتوان پروژه را روی محیط عملیاتی نیز راه‌اندازی نمود.
2. تمامی کدهای پروژه باید از ابتدا در یک codebase که توسط تیم کارفرما نیز قابل‌دسترسی است، قرار گیرند. تمامی مولفههای توسعه داده شده، تنظیمات صورت‌گرفته، مولفههای و پلاگین‌های توسعه داده شده، پایپ لاین‌های داده، و اسکریپت‌های نصب، راه‌اندازی، و نگهداری باید در codebase قرار گیرند.
3. تمامی مولفههای توسعه داده شده باید به‌صورت Docker قابل‌اجرا باشند .Dockerfileهای مربوطه و اسکریپت‌های Build باید در codebase قرار گیرد.
4. ابزارهای 3rd Party مورداستفاده باید با هماهنگی با تیم کارفرما انتخاب شوند. به‌عنوان‌مثال مدنظر است برای orchestration از ابزار Kubernetes و برای صف از ابزار Kafka استفاده شود.
5. مولفههای توسعه داده شده و تنظیمات صورت‌گرفته باید به‌نوعی باشند که بر روی زیرساخت ابری نیز قابل‌اعمال بوده و سازگار باشد.
6. لیست تغییرات بین هر دو نسخه اصلی ارائه شده و درصورتی‌که تغییر از یک نسخه به نسخه دیگر نیازمند migration می‌باشد راهنما و اسکریپت لازم برای این تغییر ارائه شود.
7. تیم پیمانکار موظف به ارائه مستندات و آموزش کافی در مورد جزئیات مؤلفه‌ها و همچنین نصب، پیکربندی، راه‌اندازی، و نگهداری محصول می‌باشد.
8. در صورت نیاز، نماینده فنی کارفرما باید بتواند جلسات هفتگی با تیم پیمانکار داشته باشد تا در جریان نحوه و کیفیت پیشبرد اهداف قرار گیرد و از طرفی آموزش‌های موردنیاز را دریافت کند.
9. در صورت رخداد مشکلی در نصب و راه‌اندازی و یا Bug در زمان اجرا در محیط عملیاتی، تیم پیمانکار موظف به حل مشکل می‌باشد.
10. اصول کیفیت کد شامل Clean Code،Unit Testing و Documentation باید در codebase رعایت شود.
11. کدها، مؤلفه‌ها، آرتیفکت‌ها، تنظیمات و اسکریپت‌های نصب و راه‌اندازی باید طی جلساتی به نماینده کارفرما تحویل داده شده و انتقال دانش موردنظر نیز انجام پذیرد. موارد تحویلی باید از جهت دارابودن سطح کیفیت کافی ذکر شده در بندهای بالا، مورد تأیید کارفرما قرار گیرد.
12. تمامی مؤلفه‌ها باید به‌صورت کوبرنتیزی (از طریق هلم) انتشار یابند. سرویس‌های غیر کوبرنتیزی موردنیاز از طریق مذاکره و تخصیص زمان قابل فراهم‌سازی است.
13. پادها امکان دسترسی به دیسک‌های لوکال و یا اجرا با privilege بالا را ندارند و برای بحث ذخیره‌کردن state باید از api سرویس‌های گرانیتی مانند پایگاه‌های داده استفاده کنند.
14. تمامی objectهای کوبرنتیزی پروژه باید در namespace مشخص شده قرار گیرند. استفاده از taint و affinity با مذاکره قابل‌انجام است.
15. به‌منظور اتصال به سرویس‌های موجود در گرانیت، آدرس و credentials سرویس‌ها به‌صورت متنی در اختیار سرویس‌گیرنده قرار می‌گیرد.
16. برای مدیریت پیش‌نیازهای مؤلفه‌ها بر روی بستر گرانیت مانند ایجاد یا تغییر پایگاه‌های داده و جداول، می‌توان از CRDها و init-container های گرانیتی استفاده نمود.
17. مؤلفه‌ها باید از بستر مانیتورینگ (Grafana/Prometheus) و مدیریت لاگ (EFK)گرانیت استفاده کرده ،و Observability موردنیاز برای نگهداشت سامانه با SLA مشخص شامل آلرت‌های disaster و high و داشبوردهای مناسب را ایجاد کنند.
18. SPOF در سطح فرایند یا داده نباید وجود داشته باشد؛ لذا تمامی مؤلفه‌ها باید HA باشند و داده‌ها بر روی بسترهای با replication ذخیره‌سازی شوند. همچنین درصورتی‌که نیاز به فرایندهای مشخصی برای Disaster Recovery همچون پشتیبان‌گیری از داده‌های حیاتی وجود دارد، این فرایندها باید مستندسازی یا خودکار شوند.

1. Model drift [↑](#footnote-ref-1)
2. <https://www.samsungsds.com/vn/ai-kubeflow/kubeflow.html> [↑](#footnote-ref-2)