باسمه‌تعالی

**سند معماری**

**توسعه بستر MLOps**

به سفارش

شرکت سامانه گستر سحاب پرداز

ارائه‌دهنده

**شرکت فناوری اطلاعات آرمان دید نو**

) آدین(

تابستان ۱۴۰۲

فهرست مطالب

[1- مقدمه.... 3](#_Toc143960600)

[2- معماری منطقی..... 4](#_Toc143960601)

[1-2- مؤلفه‌های اصلی 5](#_Toc143960602)

[2-2- جریان داده.. 7](#_Toc143960603)

[3- معماری فیزیکی.... 8](#_Toc143960604)

[1-3- Kubeflow 9](#_Toc143960605)

[2-3- خطوط لوله.. 9](#_Toc143960606)

[3-3- جمع‌آوری و پردازش داده 11](#_Toc143960607)

[4-3- ارائه ابزار، محیط و استانداردهای مناسب برای تحلیل داده. 12](#_Toc143960608)

[3-5- نوت‌بوک‌های Jupyter 13](#_Toc143960609)

[6-3- تنظیم هایپرپارامترها 13](#_Toc143960610)

[3-6-1- معماری Katib 14](#_Toc143960611)

[7-3- انباره ویژگی آنلاین. 15](#_Toc143960612)

[3-8- انباره ویژگی آفلاین....... 15](#_Toc143960613)

[3-9- آموزش، بهینه‌سازی و ارزیابی مدل 16](#_Toc143960614)

[10-3- استقرار و سرو مدل 16](#_Toc143960615)

[11-3- ‌مانیتورینگ، نظارت و مدیریت لاگ‌ها 18](#_Toc143960616)

[12-3- محل ذخیره‌سازی مدل 20](#_Toc143960617)

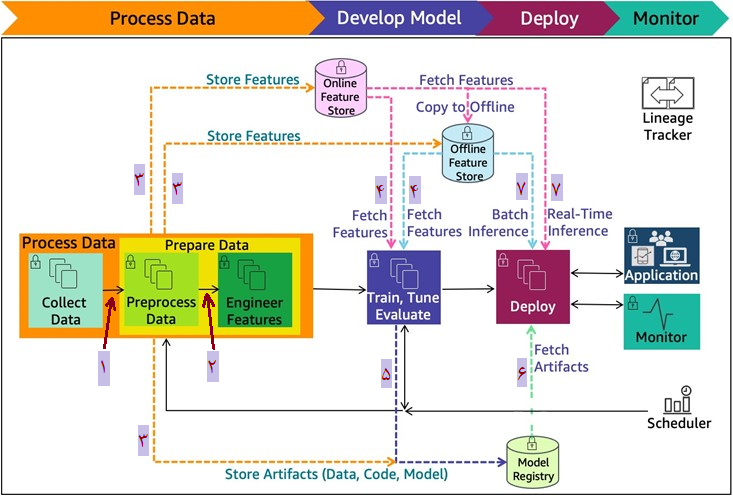
# مقدمه

سند حاضر به‌منظور ارائه معماری فنی در فاز اول پروژه «**توسعه بستر MLOps**» تهیه شده است. بدین منظور راهکار پیشنهادی شرکت آدین و جزئیات مربوط معماری منطقی و فناوری‌های به‌کاررفته در معماری فیزیکی تشریح شده است. به طور خلاصه در جدول زیر، نسخه استفاده شده برای تمامی فناوری‌های استفاده شده در معماری سامانه مشخص شده است.

|  |  |
| --- | --- |
| **فناوری** | **نسخه** |
| Metabase | Latest |
| Redis | Latest |
| Postgres | Latest |
| Python Library | Latest |
| MinIO | Latest |
| Nexus | Latest |
| Jenkins | Latest |
| Kubeflow | [v1.5.0](https://github.com/kubeflow/kubeflow/tree/v1.5.0/components/notebook-controller/config) |
| Training Operator | [v1.4.0](https://github.com/kubeflow/tf-operator/tree/v1.4.0/manifests) |
| Notebook Controller | [v1.5.0](https://github.com/kubeflow/kubeflow/tree/v1.5.0/components/notebook-controller/config) |
| Jupyter Web App | [v1.5.0](https://github.com/kubeflow/kubeflow/tree/v1.5.0/components/crud-web-apps/jupyter/manifests) |
| Katib | [v0.13.0](https://github.com/kubeflow/katib/tree/v0.13.0/manifests/v1beta1) |
| KFServing | [v0.6.1](https://github.com/kubeflow/kfserving/releases/tag/v0.6.1) |
| KServe | [v0.7.0](https://github.com/kserve/kserve/tree/v0.7.0) |
| Kubeflow Pipelines | [1.8.2](https://github.com/kubeflow/pipelines/tree/1.8.2/manifests/kustomize) |

# معماری منطقی

در این بخش به معرفی مؤلفه‌های اصلی در معماری یک بستر MLOps در سطح منطقی و فیزیکی با ذکر مصادیق فناوری‌ها و زیرساخت‌ها پرداخته شده؛ همچنین جریان داده اصلی و سناریو تعامل مؤلفه‌ها شرح داده شده است. شکل 1 یک نمای کلی از مؤلفه‌ها، وظایف و مراحل اصلی موجود در مفهوم و منطق MLOps را نمایش می‌دهد. معماری و شِما، پایه و اساس پیشنهاد توسعه مندرج در این سند است.



شکل . نمای کلی معماری منطقی بستر MLOps

## مؤلفه‌های اصلی

به‌طورکلی نمای اصلی در معماری منطقی MLOps شامل اجزا و مراحل زیر است:

**آماده‌سازی داده‌ها**: این مؤلفه جهت انجام اولین مرحله در چرخه حیات یادگیری ماشینی است و شامل جمع‌آوری، تمیز کردن و پیش‌پردازش داده‌هایی است که برای آموزش و ارزیابی مدل‌های ML استفاده می‌شود. هدف از این مرحله جمع‌آوری و آماده‌سازی داده‌ها در قالبی است که به‌راحتی توسط مدل ML مصرف شود و اطمینان حاصل شود که داده‌ها از کیفیت بالایی برخوردار هستند. آماده‌سازی داده‌ها ممکن است شامل ۳ مرحله ازجمله پردازش‌هایی مانند نرمال‌سازی داده‌ها، تشخیص نقاط پرت و مهندسی ویژگی‌ها باشد که بر اساس نیازمندی مسئله، ممکن است تمام موارد را شامل نباشد.

**انباره داده آنلاین** / **آفلاین ویژگی‌ها:** ویژگی‌ها پس از استخراج، وارد انباره‌های داده می‌شوند. انباره داده آفلاین، تاریخچه مقادیر ویژگی‌ها را نگهداری می‌کند و در مرحله یادگیری مدل مورداستفاده قرار می‌گیرد. همچنین برای استنتاج به‌صورت دسته‌ای نیز مورداستفاده قرار می‌گیرد. انباره داده آنلاین، ورودی ویژگی‌های استخراج‌شده از داده‌های خام ورودی را نگهداری می‌کند و برای تأمین انباره داده ویژگی‌های آفلاین مورداستفاده قرار می‌گیرد. همچنین بسته به نوع مدل، ممکن است در مرحله استقرار و سرو مدل نیز مورد بهره‌برداری قرار گیرد.

**مدیریت کدهای منبع:** تمامی کدهای استفاده شده در چرخه MLOps ازجمله کد منبع مدل‌های یادگیری ماشین، الگوریتم‌ها و خطوط لوله در یک مؤلفه با این نام ذخیره و نگهداری شده و بنا به نیاز در هر مرحله واکشی و استفاده خواهند شد. در شکل 1 این مؤلفه با نام Model Registry نمایش داده شده است.

**انتخاب، آموزش، تنظیم و ارزیابی مدل:** در این مرحله، دانشمندان و مهندسان داده، معماری، الگوریتم و هایپرپارامترهای مدل ML مناسب را بر اساس ویژگی‌های داده‌ها و مسئله‌ای که باید حل شود، انتخاب می‌کنند. سپس با استفاده از داده‌های آماده شده و با تنظیم پارامترهای مدل برای به‌حداقل‌رساندن خطا بین خروجی‌های پیش‌بینی‌شده و خروجی‌های واقعی، اقدام به اجرای فرایند آموزش و تنظیم مدل می‌نمایند در نهایت به 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 می‌پردازند.

**یکپارچه‌سازی مداوم / استقرار مستمر (CI/CD**) :این جزء ادغام و استقرار مدل‌های یادگیری ماشین را خودکار می‌کند. به‌عبارت‌دیگر، مدل‌های آموزش‌دیده را در یک محیط تولید مستقر می‌کند، جایی که می‌توان از آن‌ها برای پیش‌بینی استفاده کرد. این مرحله شامل در دسترس قرار دادن 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.

**نظارت 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. 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 ازجمله نسخه‌سازی، به‌روزرسانی[[1]](#footnote-1) و بازنشستگی مدل‌ها[[2]](#footnote-2) در صورت لزوم را مدیریت می‌کند. همچنین مدل‌های آموزش‌دیده و فراداده‌های مربوط به مدل‌ها مانند نسخه، دقت و داده‌های آموزشی را ذخیره می‌کند.

**نرم‌افزار نهایی:** این بخش که در شکل 1 با نام Application آمده است، محل استقرار کد یا API نهایی برای استفاده از مدل مستقر شده است. در واقع اینجا جایی است که درخواست کاربر را دریافت کرده و با استفاده از استنتاج به‌کمک مدل، جواب را برای کاربر ارسال می‌کند.

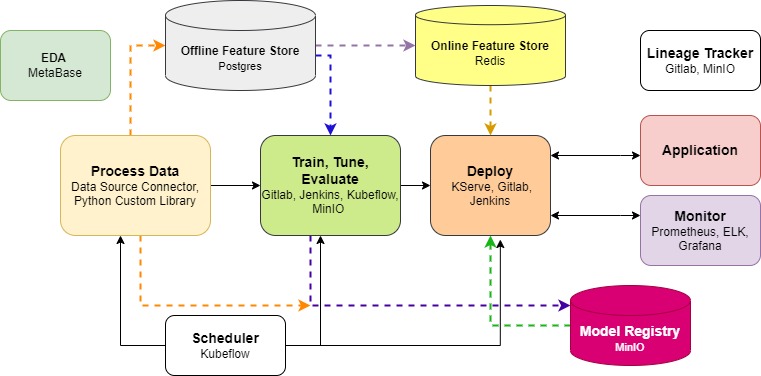
## جریان داده

جریان داده‌ای بین مؤلفه‌های اصلی معماری منطقی که در شکل 1 از روند کاری بستر MLOps ملاحظه می‌شود، در این بخش شرح داده خواهد شد. این جریان داده‌ای ناظر بر مراحل طی شده در MLOps از زمان ورود داده تا هنگام آموزش و استقرار مدل ML را شامل می‌شود. همان‌طور که در شکل 1 دیده‌ می‌شود جهت تسهیل در فهم توضیحات، کلیه مراحل جریان داده اصلی شماره‌گذاری شده‌اند. نحوه عملکرد MLOps در هر یک از مراحل به شرح زیر است:

1. ابتدا داده جمع‌آوری شده برای پیش‌پردازش ارسال می‌شود. این پیش‌پردازش می‌تواند در بستر اصلی MLOps، به‌صورت مستقل و یا در بستر داخلی Feature Store انتخابی مستقر شود.
2. داده پس از پیش‌پردازش، برای استخراج ویژگی‌ها موردبررسی قرار گرفته و ویژگی‌های موردنیاز از آن استخراج می‌شوند.
3. ویژگی‌های استخراج شده در دو انباره آنلاین و آفلاین ذخیره‌سازی می‌شوند. بسته به نوع پردازش داده (دسته‌ای یا جریانی)، ویژگی‌ها ممکن است ابتدا در انباره آنلاین ذخیره شده و بعد در انباره آفلاین کپی شوند (حالت جریانی)، یا از ابتدا مستقیماً در انباره آفلاین ذخیره شوند (حالت دسته‌ای). همچنین فراداده در مورد داده ورودی (مثلاً زمان دریافت ورودی، حجم داده، و...) در Model Registry ذخیره می‌شود.
4. در مرحله آموزش مدل، ویژگی‌ها از انباره‌ها خوانده شده و مورداستفاده قرار می‌گیرند. اینجا نیز باتوجه‌به دسته‌ای یا جریانی بودن پردازش، داده‌ها از انباره آفلاین یا آنلاین خوانده می‌شوند.
5. پس از پایان فرایند آموزش، مدل ساخته شده در محل ذخیره‌سازی مدل‌ها قرار می‌گیرد.
6. در مرحله استقرار، مدل موردنظر از محل ذخیره‌سازی مدل‌ها بارگذاری می‌شود.
7. هنگام استنتاج، بسته به نوع مدل (وابسته به داده، یا مستقل از داده) و همچنین نوع درخواست کاربر (تکی، یا دسته‌ای)، ممکن است نیاز به دسترسی به انباره داده آنلاین و یا آفلاین باشد و اطلاعات از انباره‌ها به محل استقرار مدل منتقل شوند.

# معماری فیزیکی

در این بخش به بررسی معماری فیزیکی در بستر MLOps خواهیم پرداخت. منظور از معماری فیزیکی، نگاشت مؤلفه‌های معماری منطقی به فناوری‌های مشخص جهت توسعه نمونه انتخابی از بستر MLOps است. 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. برای معماری فیزیکی (مطابق با معماری منطقی ارائه شده در بخش قبل و نیز نیازها و مسائل مطرح شده توسط کارفرما)، نمونه‌ای مطابق شکل 2 پیشنهاد می‌شود که هر یک از فناوری‌های پیشنهادی قابل‌استفاده در هر مؤلفه 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. همچنین نسخه متناظر بند نیازمندی SOC در صورت وجود، در هر مؤلفه ذکر خواهد شد. شرح پیشنهاد اولیه فناوری‌ها در لایه فیزیکی همان‌گونه که در شکل 2 نمایش داده شده است، برای هر یک از مؤلفه‌های معماری منطقی در لایه فیزیکی به‌صورت زیر خواهد بود:



شکل 2. معماری فیزیکی برای MLOps

## Kubeflow

Kubeflow یک پلتفرم قدرتمند و منعطف برای مدیریت چرخه یادگیری ماشین و یک پلتفرم منبع‌باز برای توسعه، استقرار و مدیریت مدل‌های یادگیری ماشینی (ML) در کوبر 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.

## خطوط لوله

خطوط لولهبستری برای ساخت، استقرار و مدیریت گردش ‌کار یادگیری ماشینی سرتاسری است که در زیرساخت Kubeflow وجود دارد. این محیط، یک رابط بصری برای ایجاد و مدیریت گردش‌های کاری پیچیده و همچنین ابزارهایی برای آماده‌سازی داده‌ها، آموزش مدل و استقرار فراهم می‌کند. در سطح بالا، اجزای معماری و خط لوله Kubeflow مطابق با ساختار ارائه­شده در شکل 3، به شرح زیر است:

**Python SDK:** با استفاده از زبان اختصاصی دامنه Kubeflow Pipelines (DSL) اجزا اصلی را ایجاد کرده و خط لوله را مشخص می‌کند.

**کامپایلر DSL:** کامپایلر DSL کد پایتون خط لوله را به یک پیکربندی ثابت (YAML) تبدیل می‌کند.

**Pipeline Service:** در این بخش یک خط لوله اجرا شده از پیکربندی استاتیک ایجاد می‌شود.

**منابع کوبر:** سرویس خط لوله، API سرور کوبر را فراخوانی می‌کند تا منابع کوبر (CRD)لازم را برای اجرای خط لوله ایجاد کند.

**کنترل‌کننده‌های ارکستراسیون:** مجموعه‌ای از کنترل‌کننده‌ها کانتینرهای موردنیاز برای تکمیل خط لوله را اجرا می‌کنند. کانتینرها در Podهای کوبر روی ماشین‌های مجازی اجرا می‌شوند. برای مثال، کنترل‌کننده Argo Workflow گردش‌های کاری مبتنی‌بر Task را هماهنگ می‌کند.

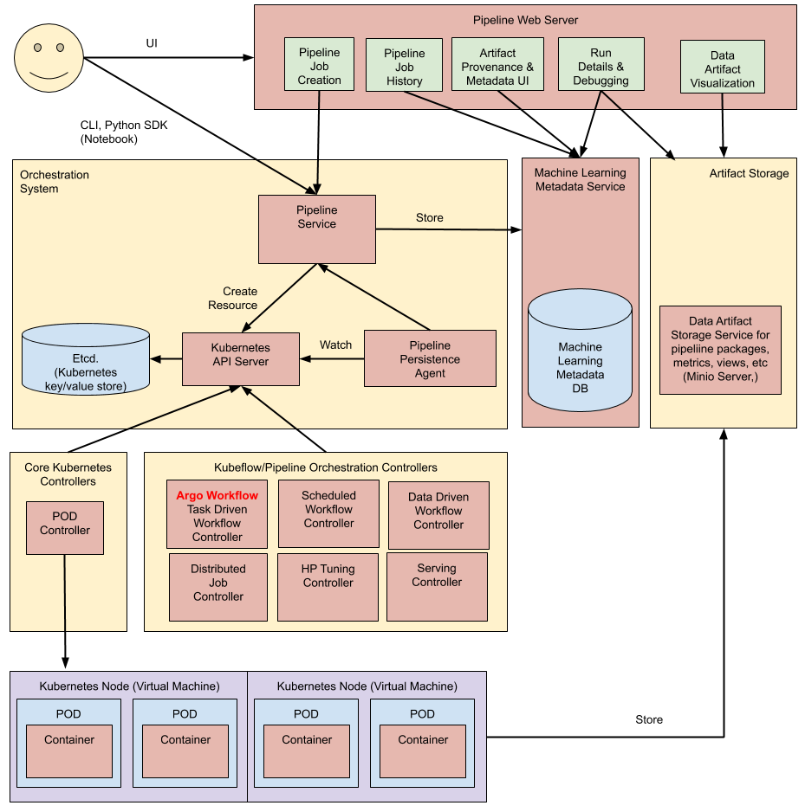
**ذخیره‌سازی آرتیتفکت‌ها:** Podها دو نوع داده را ذخیره می‌کنند:

فراداده: آزمایش‌ها، کارها، اجرای خط لوله و معیارهای داده‌های متریک به‌منظور مرتب‌سازی و فیلترکردن جمع‌آوری می‌شوند. Kubeflow Pipelines ابر داده‌ها را در پایگاه‌داده MySQL ذخیره می‌کند.

آرتیتفکت‌ها: Kubeflow Pipelines آرتیتفکت‌ها را در یک بستر ذخیره‌سازی مانند سرور MinIO یا Cloud Storage ذخیره می‌کند. پایگاه‌داده MySQL و سرور MinIO هر دو توسط زیرسیستم Persistent Volume کوبر پشتیبانی می‌شوند.

**Persistence agent and ML metadata**: این بخش مجموعه کانتینرهای اجرا شده و همچنین ورودی و خروجی آن‌ها را ثبت می‌کند. ورودی/خروجی از پارامترهای کانتینر یا URIهای آرتیتفکت‌ها تشکیل شده است.

**Pipeline web server:** وب سرور خط لوله داده‌ها را از سرویس‌های مختلف همچون لیست خطوط لوله در حال اجرا، تاریخچه اجرای خط لوله، فهرست آرتیفکت‌های داده، اطلاعات اشکال‌زدایی در مورد اجرای خط لوله جداگانه، وضعیت اجرا در مورد اجرای خط لوله جداگانه. جمع‌آوری کرده و نمایش می‌دهد.



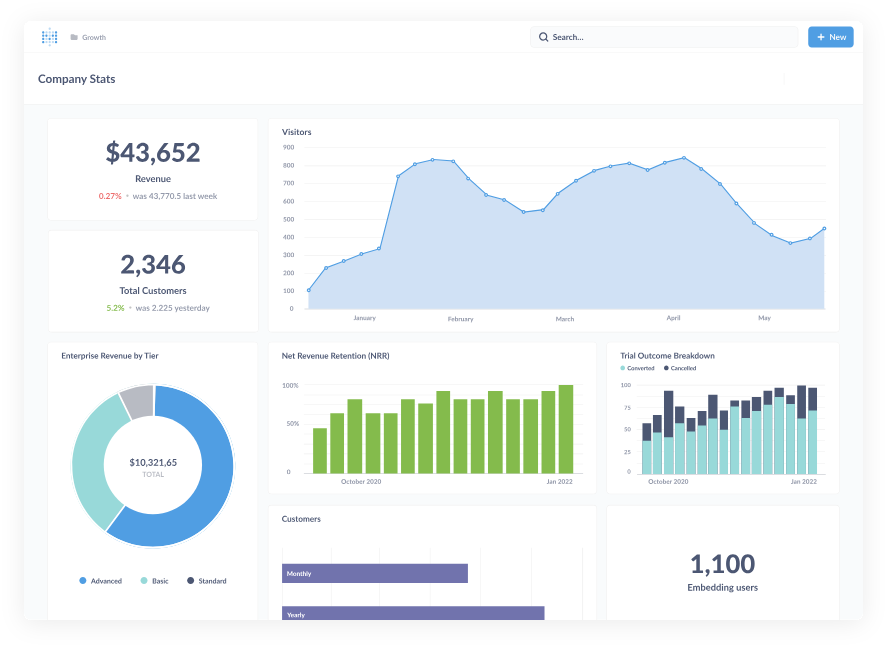
شکل . معماری جریان داده در Kubeflow

## جمع‌آوری و پردازش داده

باتوجه‌به آنکه داده‌های موردنیاز برای تحلیل در زیرساخت گرانیت ذخیره شده است، امکان اتصال و جمع‌آوری داده از زیرساخت گرانیت به‌نحوی‌که داده‌های موردنیاز در فرایند تحلیل و مدل‌سازی از مخازن ذخیره‌سازی داده جمع‌آوری شود فراهم می‌گردد. این بخش شامل جمع‌آوری داده، پیش‌پردازش داده و استخراج ویژگی‌ها بوده و هر نوع منبع داده‌ای قابل‌پذیرش است. این سامانه با ارائه کتابخانه‌های شخصی‌سازی‌شده پایتون، اتصال و خواندن داده از منابع داده‌ای کارفرما را مطابق استانداردهای موردنیاز فراهم می‌کند.

## ارائه ابزار، محیط و استانداردهای مناسب برای تحلیل داده

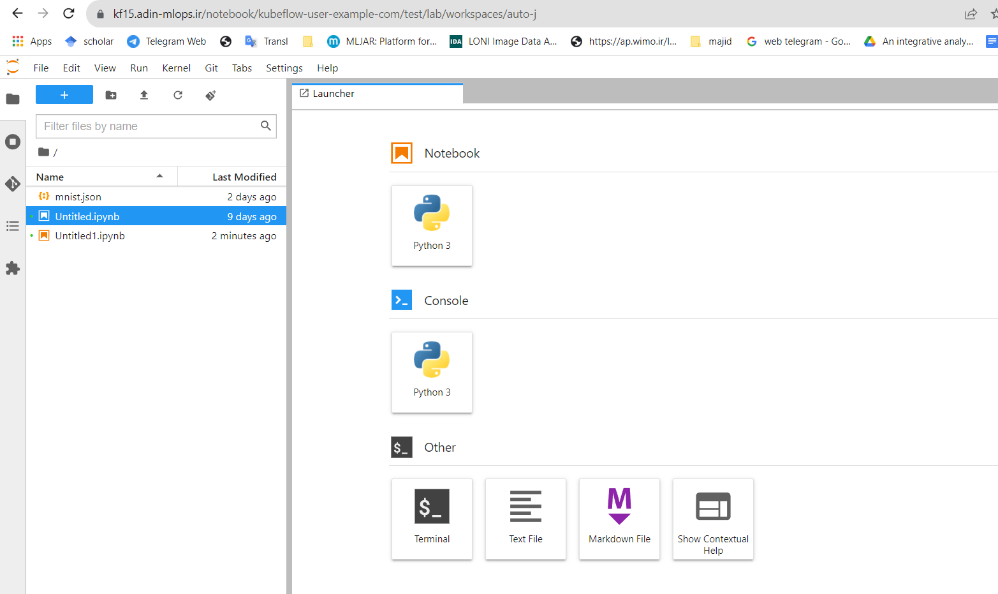
برای توسعه مدل نیاز است که ابزار و تکنولوژی در اختیار دانشمندان داده قرار داده شود که با استفاده از آن فرایند تحلیل و بررسی داده راستی آزمایی فرضیات و در ادامه استخراج و کشف بینش از داده را انجام دهند. Metabase یک ابزار متن‌باز برای ایجاد یک محیط کاربرپسند به‌منظور درک ارتباطات میان داده‌ها برای کاربران غیر فنی است. کاربران را قادر می‌سازد داشبورد، گزارش‌ها و نمودارهای تعاملی را بدون نیاز به مهارت‌های فنی گسترده ایجاد و به اشتراک بگذارند.



شکل . محیط Metabase

## نوت‌بوک‌های Jupyter

نوت‌بوک‌های Kubeflow Jupyter راهی برای اجرا و مدیریت نوت‌بوک‌های Jupyter در کوبر ارائه داده و قابلیت گردش‌های کاری یادگیری ماشین را در یک محیط توسعه و آزمایش فراهم می‌کند. همان‌گونه که در شکل 5 ملاحظه می­گردد، جهت تسهیل فرایند توسعه، امکان دسترسی هم­زمان به ترمینال و نوت‌بوک‌های Jupyter در محیط Kubeflow فراهم شده است.



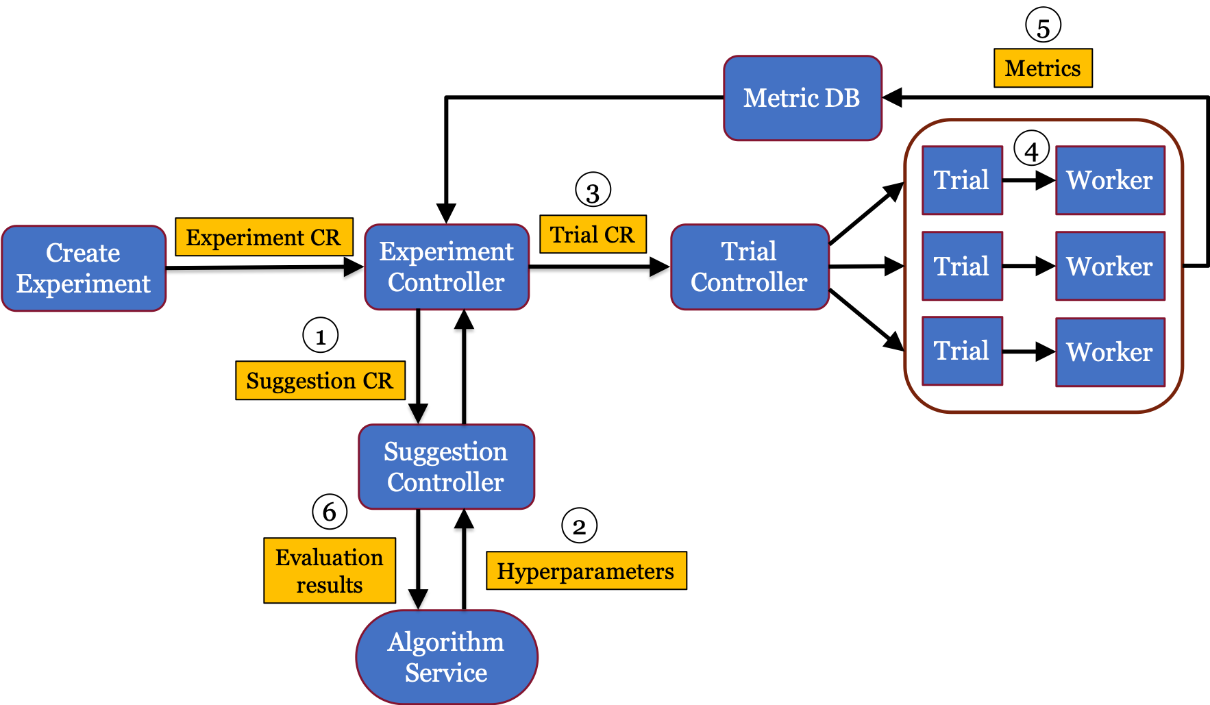
شکل 5. محیط Jupyter Notebook در MLOps

## تنظیم هایپرپارامترها

Katib چارچوبی است که فرایند یافتن هایپرپارامترهای بهینه را برای مدل‌های یادگیری ماشین خود به‌طور خودکار ممکن می‌سازد. تنظیم هایپرپارامتر شامل تنظیم پارامترهای مختلف الگوریتم یا مدل یادگیری ماشین برای بهینه‌سازی عملکرد و دستیابی به نتایج بهتر از نظر دقت و تعمیم در یادگیری ماشین است. بااین‌حال، تنظیم دستی هایپرپارامترها می‌تواند زمان‌بر بوده و نیاز به آزمون و خطای زیادی دارد. Katib این چالش را با ارائه قابلیت‌های تنظیم خودکار و توزیع شده هایپرپارامتر برطرف می‌کند. ویژگی‌های اصلی کلیدی Katib عبارت‌اند از ادغام با Kubeflow، ارائه الگوریتم و فضای جستجو، امکان تعیین معیارها برای ارزیابی و مقایسه پیکربندی هایپرپارامترهای مختلف، توقف زودهنگام، نمایش تجزیه‌وتحلیل و انتخاب توابع هدف. حداقل نسخه کوبر برای نصب Katib، نسخه ۱.۱۷ است.

### معماری Katib

Katib با استفاده از منابع کوبرنتیز جهت تنظیم مراحل زیر را انجام می‌دهد که در نمودار شکل ۶ مشخص شده است.



شکل . معماری Katib

اجزا اصلی معماری Katib شامل سه بخش اصلی از CDRهای تعریف‌شده در کوبر شامل Experiment، Suggestion و Trial است. Experiment به‌منظور بهینه‌سازی فضای جستجو و الگوریتم جستجو ایجاد شده و Suggestion مجموعه‌ای از هایپرپارامترهایی است که توسط الگوریتم جستجوی انتخابی کاربر تولید می‌شوند؛ بنابراین، Katib آزمایش‌هایی را برای ارزیابی آن‌ها ایجاد می‌کند و به‌وسیله هر Trial یک تکرار از فرایند تنظیم هایپرپارامترها انجام می‌گردد. ازآنجایی‌که هر Trial یک انتزاع از job worker است، از هر منبع کوبر مانند TFJob می‌توان برای انجام Training Job استفاده کرد. بنابراین، هنگامی که یک Experiment ایجاد شد، کنترل‌کننده Experiment یک شیء Suggestion را ایجاد می‌کند.

1. کنترل‌کننده Suggestion یک سرویس الگوریتم AutoML را بر اساس شیء Suggestion ایجاد کرده و هنگامی که سرویس الگوریتم آماده است، کنترل‌کننده Suggestion، سرویس را برای دریافت هایپرپارامترهای جدید فراخوانی می‌کند و آن‌ها را به شیء Suggestion اضافه می‌کند.
2. کنترل‌کننده Experiment متوجه می‌شود که شیء Suggestion به‌روزرسانی شده است و برای هر مجموعه از هایپرپارامترها یک شیء Trial جدید ایجاد می‌کند.
3. کنترل‌کننده Experiment برای هر شیء Experiment یک job worker ایجاد می‌کند و وضعیت هر job را بررسی می‌کند.
4. هنگامی که job worker تکمیل شد، جمع‌آورنده معیارها را دریافت و آن‌ها را در پایگاه‌داده ذخیره می‌کند.
5. کنترل‌کننده Experiment نتایج معیارها را به سرویس الگوریتم ارسال می‌کند و هایپرپارامترها جدید را از شیء Suggestion دریافت می‌کند.

## انباره ویژگی آنلاین

ویژگی‌های ذخیره شده در انباره ویژگی آنلاین، عمدتاً در هنگام استنتاج مورداستفاده قرار می‌گیرند. ویژگی موردنیاز این بخش، سرعت بالای پاسخ‌گویی تکی 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. حتی 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.

## انباره ویژگی آفلاین

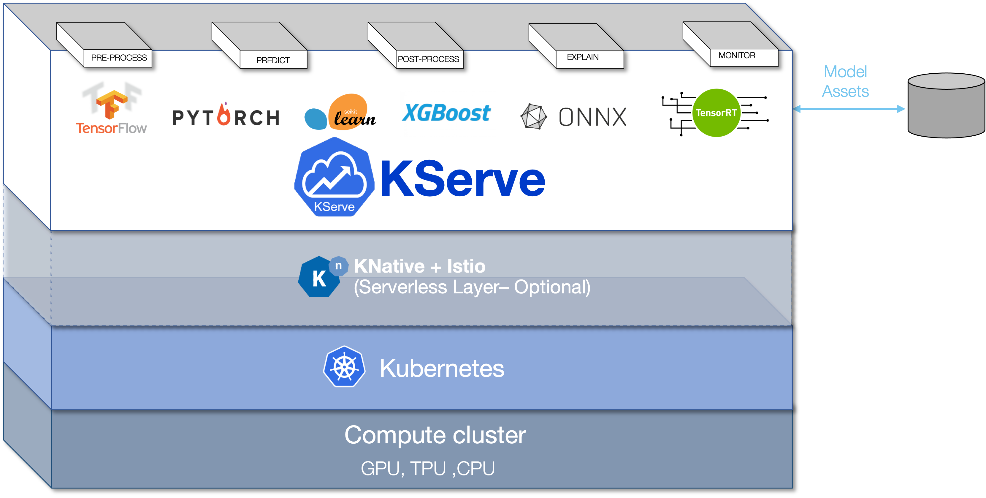
ویژگی‌های ذخیره شده در انباره ویژگی آفلاین، دو کاربرد دارند: برای آموزش مدل به‌صورت دسته‌ای، و همچنین برای استنتاج دسته‌ای. همچنین حجم داده بسیار زیادی باید در این انباره ویژگی ذخیره شود (توجه کنید که تاریخچه ویژگی‌های استخراج شده نیز در این انباره ذخیره می‌شوند). در نتیجه بهترین انتخاب برای این مؤلفه، استفاده از یک دریاچه داده (Data lake) است. نکته قابل‌ذکر این است که در صورت ارائه خدمات MLOps به‌صورت ابری، قطعاً حجم داده بیشتر و بیشتر خواهد شد و در نتیجه ناگزیر به استفاده از فناوری‌هایی مانند Clickhouse خواهیم بود. اما درصورتی‌که مشتری علاقه‌ای به‌قرار دادن داده خود بر روی خدمات ابری نداشته باشد (به‌علت محرمانگی، یا حجم بالای داده، یا هر علت دیگر) و نیاز به نصب یک نسخه از سامانه بر روی زیرساخت مشتری باشد، در این صورت کاملاً محتمل است که نیازی به پیچیدگی نصب و تنظیم و راه‌اندازی و پایش و نگهداری زیرساختی مانند Clickhouse نباشد و با استفاده از پایگاه‌های داده ساده‌تر مانند Postgres بتوان با هزینه بسیار پایین‌تر نیاز مشتری را مرتفع نمود. نکته دیگر اینکه فراداده مربوط به ویژگی‌های ذخیره شده را در Minio ذخیره می‌کنیم. در انباره ویژگی آفلاینباتوجه‌به حجم داده موردنیاز مشتری برای حجم کم یا متوسط، می‌توان از یک RDBMS مانند Postgres استفاده کرد. Postgres به‌علت سادگی نصب و کاربرد، برای ذخیره‌سازی حجم متوسط داده در Offline feature store استفاده می‌شود. این مؤلفه نیز از RAM بیشتر به‌عنوان cache استفاده می‌کند و پردازش معمول دارد. حجم دیسک نیز متناسب با حجم داده خواهد بود. در این مورد، برای بالاتر رفتن سرعت خواندن و نوشتن داده، از هارد SSD استفاده می‌کنیم.

## آموزش، بهینه‌سازی و ارزیابی مدل

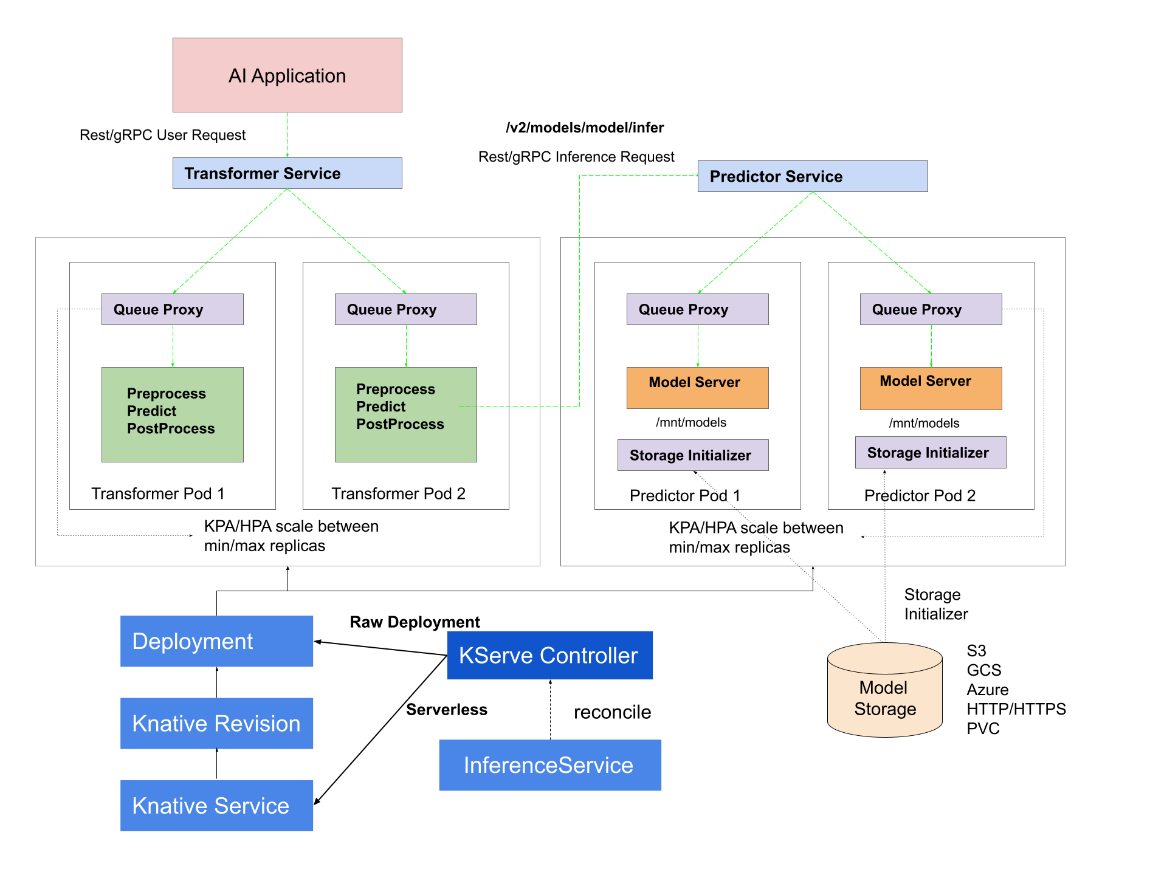
این بخش متشکل از یک خط لوله برای مراحل مختلف آموزش مدل است. این خط لوله توسط Jenkins فراخوانی شده و اجرای مراحل داخل آن تماماً توسط Kubeflow مدیریت می‌شود که برای ذخیره‌سازی داده بین مراحل خط لوله، از Minio استفاده می‌کند.

## استقرار و سرو مدل

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. استقرار مدل توسط Jenkinsدر بستر کوبر انجام می‌شود. در اینجا API استنتاج توسط Kubeflow و KServe ارائه خواهد شد. مدل‌ها و داده‌های موردنیاز، از مؤلفه‌های دیگر دریافت می‌شوند.



شکل . پشته فناوری Kserve



شکل . معماری Kserve

این بخش شامل یک خط لوله برای پردازش داده است که می‌تواند توسط Spark اجرا شود. نتایج مراحل مختلف در Minio ذخیره‌سازی می‌شود. نکته بعد اینکه پردازش داده در MLOps دو حالت دارد: دسته‌ای و جریانی. در حالت جریانی، خروجی این بخش در قرار می‌گیرد و توسط Online Feature Store خوانده شده و برای آموزش استفاده می‌شود. استقرار Kserve بر روی CPU/GPU را برای چارچوب‌های رایج ML Scikit-Learn، XGBoost، Tensorflow، PyTorch و همچنین زمان اجرا مدل سفارشی قابل اتصال را ارائه می‌دهد. برخی اجزای اصلی آن مطابق شکل ۸ عبارت‌اند از:

**کنترل‌کننده KServe:** مسئول ایجاد سرویس، منابع ورودی، ظرف سرور مدل و کانتینر عامل مدل برای ثبت  request/response loggin و pull کردن مدل است.

**Knative Serving Controller:** مسئول مدیریت بازنگری سرویس، ایجاد منابع مسیریابی شبکه یک کانتینر برای نمایش معیارهای ترافیک و اعمال محدودیت هم‌زمان است.

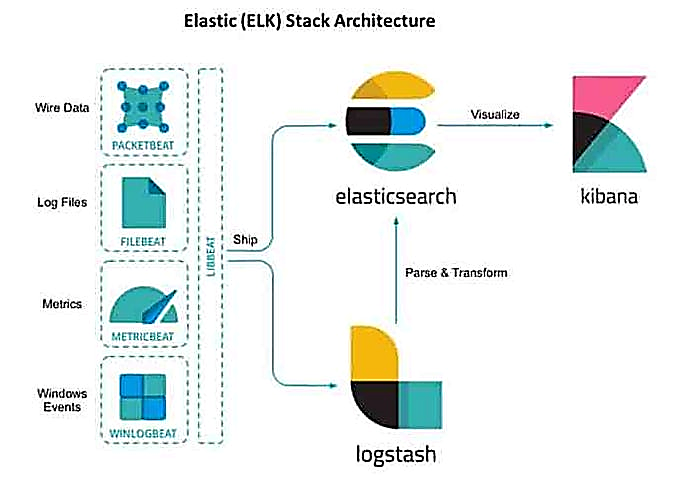
**Knative Activator:** Podهای کم‌حجم شده تا صفر را باز می‌گرداند و درخواست‌ها را Forward می‌کند.

**Knative Autoscaler (KPA):** جریان ترافیک به برنامه را نظارت می‌کند و کپی‌ها را بر اساس معیارهای پیکربندی‌شده کاهش یا افزایش می‌دهد.

## ‌مانیتورینگ، نظارت و مدیریت لاگ‌ها

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. مانیتورینگ به شناسایی مشکلات احتمالی سیستم، مانند جابه‌جایی مدل کمک می‌کند و پاسخ سریع برای حفظ عملکرد سیستم را ممکن می‌سازد. مانیتورینگ امکان می‌دهد آزمایش‌های مختلف را ردیابی و مقایسه کرده و عملکرد مدل‌های خود را در طول زمان پیگیری 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. فناوری مورداستفاده ELK یک پشته فناوری است که از سه ابزار منبع‌باز Elasticsearch، Logstash و Kibana تشکیل شده است. این ابزارها جهت جمع‌آوری، تجزیه‌وتحلیل، جستجو و نمایش داده‌های حجیم در زمان واقعی مورداستفاده قرار می‌گیرند و توسط ELK برای مدیریت گزارش، نظارت به کار گرفته می‌شوند. پشته ELK همچنین می‌تواند با ابزارها و پلاگین‌های اضافی گسترش‌یافته و سفارشی شود تا نیازهای خاص را برآورده کند. Elasticsearch یک موتور جستجو و تجزیه‌وتحلیل توزیع شده و مقیاس‌پذیر است که امکان می‌دهد داده‌ها را در ابعاد مختلف جستجو، فیلتر و تجمیع کنید. از سوی دیگر، Logstash یک ابزار خط لوله پردازش داده است که داده‌ها را از منابع مختلف قبل از ارسال به Elasticsearch برای نمایه‌سازی جمع‌آوری، پردازش و تبدیل می‌کند. این سامانه از طیف وسیعی از منابع ورودی مانند گزارش‌ها، معیارها و رویدادها پشتیبانی می‌کند. همچنین از نرمال‌سازی و تبدیل داده‌ها برای تسهیل تجزیه‌وتحلیل داده‌ها پشتیبانی می‌کند. Kibana نیز یک ابزار قدرتمند نمایش و کاوش داده است که در ارتباط با Elasticsearch کارکرده و یک رابط وب کاربرپسند برای تعامل با داده‌های نمایه شده در Elasticsearch فراهم می‌کند. با Kibana می‌توان داشبوردهای تعاملی، نمایش‌ها و نمودارها را برای کاوش و تجزیه‌وتحلیل داده‌ها ایجاد کنید. Prometheus یک جعبه‌ابزار نظارت و هشدار منبع‌باز است که جهت جمع‌آوری و تجزیه‌وتحلیل داده‌های سری زمانی به شیوه‌ای مقیاس‌پذیر و کارآمد است. مفهوم اصلی Prometheus جمع‌آوری و ذخیره‌سازی داده‌های سری زمانی است که معمولاً معیارهایی مانند استفاده از CPU، مصرف حافظه، تأخیر شبکه و موارد دیگر را نشان می‌دهد. علاوه بر این، Prometheus دارای یک سیستم هشدار قدرتمند است که به کاربران اجازه می‌دهد قوانین و آستانه‌هایی را برای تولید هشدارها بر اساس شرایط خاص تنظیم کنند. این هشدارها را می‌توان از طریق کانال‌های مختلفی مانند ایمیل، Slack، PagerDuty و غیره ارسال کرد. همچنین با Grafana، ابزار محبوب نمایش داده‌ها، به‌خوبی ادغام می‌شود. به‌طورکلی، Prometheus یک راه‌حل انعطاف‌پذیر و مقیاس‌پذیر برای نظارت و هشدار در سیستم‌های توزیع شده مدرن ارائه می‌دهد.

Grafana یک ابزار نمایش و نظارت بر داده‌های منبع‌باز است که داشبوردهای قابل‌تنظیم برای نمایش و تجزیه‌وتحلیل داده‌ها از منابع مختلف را ایجاد کنند. Grafana از ادغام با چندین منبع داده ازجمله پایگاه‌های داده محبوب مانند MySQL، PostgreSQL، MinIO، Elasticsearch و Prometheus پشتیبانی می‌کند. این ابزار یک رابط یکپارچه برای اتصال و جمع‌آوری داده‌ها از چندین منبع فراهم کرده و نظارت و تجزیه‌وتحلیل داده‌ها را در پلتفرم‌های مختلف آسان‌تر می‌کند. با استفاده از Grafana، کاربران می‌توانند داشبوردهای بصری جذاب با گزینه‌های نمایش همچون نمودارها، نمودارها، جداول را ایجاد کرده و طیف وسیعی از گزینه‌های سفارشی‌سازی را ارائه می‌دهد. همچنین، این ابزار ویژگی‌هایی مانند هشدار و اعلان را ارائه می‌دهد که در آن کاربران می‌توانند قوانینی را برای ایجاد هشدار بر اساس شرایط مشخص تنظیم کنند؛ بنابراین تا زمانی که معیارهای از پیش تعیین شده از آستانه‌ای خاص فراتر می‌روند یا از آن کمتر می‌شوند، اعلان فوری می‌دهد و به نظارت فعال و حل مشکلات پلتفرم MLops کمک می‌کند.



شکل -معماری ELK

## محل ذخیره‌سازی مدل

این بخش محل نگهداری نسخه‌های مختلف فراداده‌ها، مدل‌ها و کدها است، و هنگام استقرار، مدل از آن بارگذاری می‌شود. برای ذخیره‌سازی نسخه‌های مختلف مدل از Minio استفاده می‌شود، و تمام Artifactها مانند imageهای داکر و پکیج‌های پایتون در Nexus ذخیره می‌شوند که مخصوص ذخیره و بازیابی انواع Artifactها است. برای نگهداری و استفاده از کدهای Gitlab گزینه مناسبی به نظر می‌رسد. همچنین برای ذخیره‌سازی فراداده مربوط به داده ورودی، از Minio استفاده می‌شود.

1. به‌روزرسانی مدل بر اساس شرایط مشخص (مانند زمان‌بندی قبلی یا بر اساس پارامترهای مورد پایش) انجام می‌شود. در صورت بروز هر یک از شرایط تعیین شده، عملیات آموزش مدل مجدداً اجرا شده و نسخه جدید مدل مستقر می‌شود. [↑](#footnote-ref-1)
2. 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. [↑](#footnote-ref-2)