|  |  |  |
| --- | --- | --- |
| Description: Description: بازنشسته.jpg | ارتش جمهوری اسلامی ایران (فرماندهی جنگال و سایبر راهبردی آجا ) |  |

عنوان طرح:

طراحی و پیادهسازی سامانه هوشمند نظارت بر تصاویر دوربین های پیرامونی تحت شبکه

نام و نام خانوادگی مجری طرح:

حسین غلامی

استاد راهنما

نام کامل استاد راهنما

استادمشاور

نام کامل استاد مشاور

پاییز 1400

به نام خدا

صفحه تعهد نامه

چكيده

امنیت یکی از مفاهیم اولیه ای است که انسان از ابتدا با آن درگیر بود و همواره در صدد افزایش آن است. یکی از مواردی که در سال های اخیر برای افزایش امینیت رونق گرفته و پیشرفت کرده، ایجاد امنیت توسط سیستم­های نظارتی و دوربین هاست. دوربین ها نمیتوانند به صرف ذخیره اطلاعات امنیت را تامین کنند، و برای ایجاد امنیت نیازمند آن هستیم که شخصی بر دوربین ها نظارت دائمی داشته باشد که فرآیندی هزینه بر است و احتمال وجود خطای انسانی وجود دارد. در این پروژه هدف پیاده سازی سیستمی است که بتواند به کمک شخص ناظر آمده و فرایند مانیتور کردن دوربین ها را با کمک هوش مصنوعی و پردازش در یک سرویس دهنده تسهیل کند. در این پروژه سه برنامه کاربردی ارائه شده، به این نحوه که یک برنامه ای بر روی کامپیوتر سرویس دهنده ارائه شده که اطلاعات دوربین ها را دریافت و فرایند مدریت کاربران را انجام دهد، برنامه ای بر روی کامپیوتر سرویس دهنده دیگری اجرا شده که پردازش­های مربوط به هوش مصنوعی را انجام دهد، و برنامه ای برای کاربران ارائه شده که بتوانند فرایند دسترسی به اطلاعات دوربین ها را برعهده بگیرند. همچنین در این پروژه برای دوربین ها سطح دسترسی تعریف شده که کاربران بر اساس سطح دسترسی خود به دوربین مذکور دسترسی پیدا کنند. و البته اطلاعات دوربین ها در کامپیوتر سرویس دهنده ذخیره شده که کاربران بتوانند به اطلاعات ذخیره شده دسترسی پیدا کنند. برای این امر از سه سرویس RabbitMQ ، MinIO ، Redis ، PyQt استفاده شد. از RabbitMQ برای به جریان انداختن اطلاعات به صورت زمان واقعی ، از MinIO برای ذخیره سازی اطلاعات دوربین ها و دسترسی به آن ها در شبکه ، از Redis برای همگام سازی سه برنامه کاربردی با یکدیگر و از PyQt برای پیاده سازی ظاهر برنامه های کاربردی استفاده شد.

واژه‌های کلیدی:

VideoStraming,RabbitMQ,MinIO,Docker,PyQt

|  |  |
| --- | --- |
| فهرست مطالب | صفحه |

[1 فصل اول مقدمه 5](#_Toc90319291)

[1-1 پیشگفتار و اهمیت موضوع 6](#_Toc90319292)

[2-1 کار های پیشین انجام شده در این زمینه 7](#_Toc90319293)

[3-1 طرح مسئله 7](#_Toc90319294)

[4-1 ساختار پایان نامه 8](#_Toc90319295)

[2 فصل دوم سرویس های مورد استفاده 9](#_Toc90319298)

[1-2 مفهوم یکپارچهسازی 10](#_Toc90319299)

[2-2 مروری بر داکر 13](#_Toc90319300)

[3-2 مروری بر کارگزار RabbitMQ 17](#_Toc90319301)

[1-3-2 راهاندازی RabbitMQ با داکر 22](#_Toc90319302)

[4-2 مروری بر ذخیره کننده Redis 23](#_Toc90319303)

[1-4-2 راهاندازی Redis با داکر کامپوز 26](#_Toc90319304)

[5-2 پیاده‌سازی سیستم دریافت و ذخیره‌سازی 26](#_Toc90319305)

[6-2 جمع بندی 29](#_Toc90319306)

[3 فصل سوم طراحی و معماری برنامه های کاربردی 30](#_Toc90319307)

[1-3 گوجه فروشی حسین 31](#_Toc90319308)

[2-3 خیار فروشی محمد 31](#_Toc90319309)

[3-3 جمع بندی 31](#_Toc90319310)

[4 فصل چهارم بکارگیری و نحوه کارکرد سیستم برای کاربران 32](#_Toc90319311)

[1-4 راه اندازی و پیشنیاز ها 33](#_Toc90319312)

[2-4 جمع‌بندی فصل چهارم 33](#_Toc90319313)

[5 فصل هفتم نتیجه‌گیری و پیشنهادها 34](#_Toc90319314)

[1-5 نتیجهگیری 35](#_Toc90319316)

[2-5 پیشنهادها 36](#_Toc90319317)

[6 مراجع 37](#_Toc90319318)

|  |  |
| --- | --- |
| فهرست اشكال | صفحه |

[شکل ‏4–1-مثالی از یکپارچه‌سازی مبتنی بر فایل 11](#_Toc90319227)

[شکل ‏4–2-مثالی از یکپارچهسازی مبتنی بر پایگاهداده 12](#_Toc90319228)

[شکل ‏4–3-نمونه از یکپارچهسازی بر اساس ارتباط مستقیم میان برنامه‌ها 12](#_Toc90319229)

[شکل ‏4–4-نمونه‌ای از یکپارچهسازی با کمک یک کارگزار پیام 13](#_Toc90319230)

[شکل ‏4–5-موتور و هسته داکر که روند فرایند مدیریت برنامه را مشخص میکند. [22] 15](#_Toc90319231)

[شکل ‏4–6- نمایی از معماری داکر و فرایند چرخش دستورات و اطلاعات. [22] 17](#_Toc90319232)

[شکل ‏4–7-نمونه‌ای از یک فایل dockerfile 17](#_Toc90319233)

[شکل ‏4–8- نمونه‌ای از یک فایل docker-compose.yml 18](#_Toc90319234)

[شکل ‏4–9-معماری کلی نرم‌افزار RabbitMQ [23] 19](#_Toc90319235)

[شکل ‏4–10- داشبورد مدیریتی نرم‌افزار RabbitMQ 20](#_Toc90319236)

[شکل ‏4–11-انتقال پیام در نرم‌افزار RabbitMQ [23] 21](#_Toc90319237)

[شکل ‏4–12-نحوه استفاده از کلیدمسیریابی برای صف‌ها 22](#_Toc90319238)

[**شکل ‏4–13-اتصال دو مصرف‌کننده‌ی اطلاعات به یک صف** 22](#_Toc90319239)

[شکل ‏4–14-محتوای داکر فایل برای راه‌اندازی RabbitMQ 23](#_Toc90319240)

[شکل ‏4–15- بخش مرتبط با RabbitMQ در داکر کامپوز 23](#_Toc90319241)

[شکل ‏4–16- بررسی دستورات کار کردن با هشها در Redis 26](#_Toc90319242)

[شکل ‏4–17- بخش مرتبط با Redis در داکر کامپوز 27](#_Toc90319243)

[شکل ‏4–18- فرایند انتقال بسته‌ها از بخش توزیع دادگان به صفها 28](#_Toc90319244)

[شکل ‏4–19-کلاس مربوطه به برنامه انتقال‌دهنده اطلاعات RabbitMQ به Redis 29](#_Toc90319245)

[شکل ‏4–20-تابع فراخوانی اطلاعات هنگام دریافت اطلاعات از Redis و انتقال به RabbitMQ 30](#_Toc90319246)

|  |  |
| --- | --- |
| فهرست جداول | صفحه |

**No table of figures entries found.**

# فصل اول مقدمه

## پیشگفتار و اهمیت موضوع

امروزه امنیت مهمترین آرمان هر کشوري تلقی می شود و در صورت عدم تامین یا وجود نقص و تنگنا در ایجاد آن، حیات سالم شهروندان به چالش کشیده میشود. [1] طبعاً مکانها با تمام خصوصیات خود و نوع برداشت ذهنی عملکرد انسان در آنها، میتواند شکل­دهنده به فضایی با نوع امنیت مثبت یا منفی باشد. به بیانی واضحتر، بعضی فضاها بهدلیل خصوصیاتی که دارند، می توانند ظرفیت تولید جرائم بالایی را داشته باشند. همزمان و درعین حال که جرائم در مکانها صورت میگیرند، حاکی از این واقعیت است که، کنترل و نظارت دولتها بر برخی مکانها درحداقل سطح ممکن قرار دارد و بدیهی است که امنیت چه در بعد عینی و چه در بعد ذهنی، در آنها کاهش مییابد. کوچه­ها و خیابان­هایی که حداقل میزان عبور و مرور را توسط ساکنین، گشت نیروي انتظامی و ... دارند، بدلیل وجود حداقل نظارت، شاهد افزایش انواع جرائم (سرقت، قتل، قاچاق مواد مخدر و ...) میباشند و به طور کلی فضاهایی با حداقل نظارت را پدید می­آورند. بنابراین، باید از روشها و ابزارهایی استفاده کرد که امنیت اجتماعی مکانهاي مختلف را افزایش دهد و در نتیجه انگیزه مجرمان و خلافکاران در ارتکاب جرم کاهش پیدا کند. یکی از این روشها بکارگیري سیستم نظارت هوشمند در فضاهاي جرم­خیز است. معمولاً در شهرهاي بزرگ به دلیل گستردگی قلمرو و وسعت حوزة استحفاظی، برخی اماکن از تیررس مأموران انتظامی دور میماند و یا در زمانی که آنها مشغول پاسبانی از نقطهاي دیگر هستند، در جایی دیگر به دلیل نبود نظارت، جرم صورت میگیرد. پس باید یا تعداد افراد بیشتر شود که بنابه دلایلی (بودجه و شرایط سخت استخدام نیرو) این کار تقریباً غیرممکن است و یا باید ناظران الکترونیکی مراقب امنیت محل باشند. بنابراین، براي رسیدن به نظم اجتماعی، برنامه­ریزي امور اجتماعی و ساماندهی عملکردها میتواند در پیشگیري از جرائم غیرقابل کنترل ثمربخش باشد. امروزه با پیشرفت تکنولوژي و کوچک شدن زمان و کاهش فواصل، شرایط کاهش تخلفات اجتماعی با قانومند کردن شهروندان در سایه حاکمیت قانون بیش از گذشته فراهم شده است، اما باید دقت کرد که شیوه­هاي برنامه ریزي­هاي اجتماعی توسط مسئولین کشور با توجه به زمان و مکانهاي مختلف در این برنامه­ها حائز اهمیت است. شایان ذکر است، دوربین مداربسته با افزایش خطر دستگیري مجرمان موجب پیشگیري از جرم میشود. با توجه به اینکه مجرمان در هنگام ارتکاب جرم به خطر دستگیري بیشتر از شدت مجازات فکر می­کنند، چون دوربین مدار بسته این خطر را افزایش میدهد در پیشگیري از جرم بسیار مؤثر است [2]. استفاده از دوربین های نظارتی همانگونه که برای بکارگیری در محیط های شهری و ایجاد امنیت لازم است برای ارگان های نظامی از اهمیت بیشتری برخوردار است چرا که این محیط ها از اهمیت بیشتری برخوردار هستند و اگر امنیت این مکان ها مورد سوال واقع شود، احساس امنیت در جامعه از بین خواهد رفت. در عامه ضرب المثلی گفته میشود که "هر چه بگندد نمکش میزنند، وای به روزی که بگندد نمک" دقیقا همین ضرب المثل گویا همه چیز است. گفتیم که استفاده از دوربین های نظارتی لازم ولی استفاده از این دوربین ها برای تامین امنیت زمانی موثر خواهد بود که اشخاصی وجود داشته باشند و بر این دوربین ها نظارت دائمی داشته باشند، چرا در صورتی که این اطلاعات زیر نظر گرفته نشود. از وقوع حادثه جلوگیری نمیشود و تنها احساس امنیت کاذب به وجود می­آید. پس نیاز داریم که دوربین ها به صورت تمام وقت بررسی شوند که این کار باید توسط اشخاص انجام شود. و هر جایی که انسان وظیفه بررسی را خود به عهده میگیرد احتمال وجود خطای انسانی وجود دارد. برای کاهش خطای انسانی میتوان اقداماتی همچون کوتاه کردن شیفت کارمندان اتاق مانیتورینگ یا افزایش تعداد آن ها را انجام داد ولی همچنان احتمال وجود خطا وجود دارد. در ادامه راهی ارائه میشود که بتوان احتمال وجود خطا را کاهش داد.

در سال های اخیر با پیشرفت هوش مصنوعی و یادگیری ماشین، الگوریتم هایی ارائه شده اند که بتوان با کمک ان ها فرایندها را ساده تر نمود. (الگوریتم های از قبیل تشخیص انسان تشخیص پلاک تشخیص چهره و....)

شرکت ها در دنیا از این الگوریتم ها استفاده کرده و محصولات هوشمندی را روانه بازار کرده اند. که میتوان به الگوریتم های تشخیص چهره باز کردن قفل تلفن های همراه اشاره نمود. از این قبیل الگوریتم ها می­توان در بحث امنیت و سیستم های نظارتی نیز استفاده نمود و آن ها را به اتاق مانیتورینگ آورد تا احتمال وجود خطای انسانی را کاهش داد. برای مثال میتوان الگوریتم های تشخیص انسان را برای مناطقی که ورود اشخاص به نواحی ممنوع است به اجرا در آورد تا در صورتی که اشخاص غیرمجاز وارد آن مناطق شدند. هشداری به صدا درآید. یا در صورتی که پلاکی غیرمجاز وارد محدوده ای که اجازه ورود به را نداشت وارد شد، هشدار به صدا در آید و ...

همانطور که دیدیم برای یک محیط مشخص میتوان الگوریتم های متعددی برای نواحی خاصی بکار گرفته شود که نیازمند یک سیستم یکپارچه برای بررسی و کنترل است. پیاده سازی چنین سیستمی که بتواند هر نیازی که مطرح شود را برآورده کند. کار ساده ای نیست و نیاز است که همچنین سیستمی به صورت بومی پیاده سازی شود تا نیازهای روزمره و گوناگون ارگان ها و شرکت ها را برآورده کند. در ادامه شرکت هایی که چنین سرویس هایی را در دنیا ارائه میدهند بررسی می­کنیم.

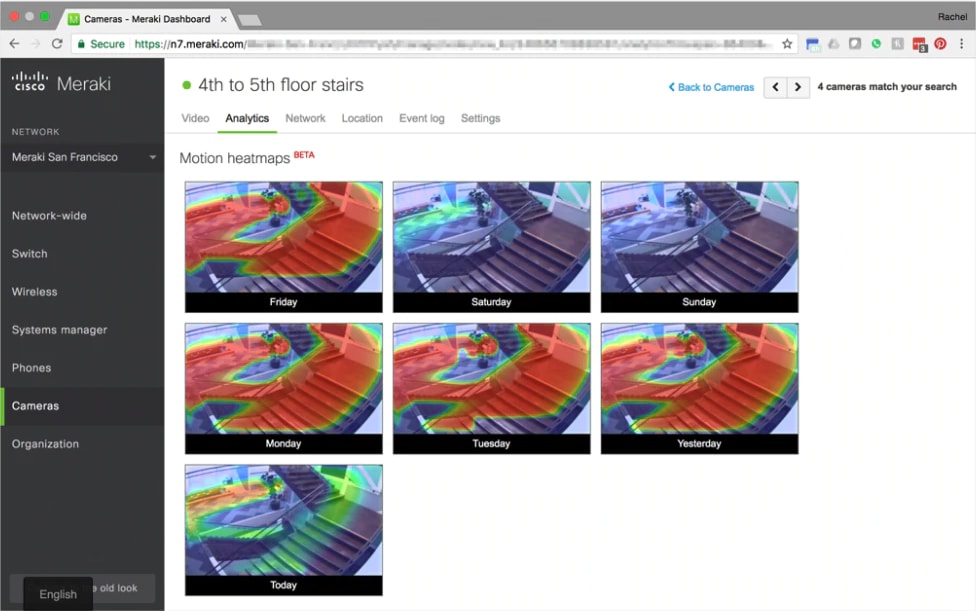
## کار های پیشین انجام شده در این زمینه

انتخاب دوربین های امنیتی تنها گام مهم در اجرای نظارت تصویری نیست. نرم افزارهای مدیریت تصویر در پشت دوربین باید از عملکردهایی که کاربر به دنبال آن است پشتیبانی کند. قبل از اینکه به بررسی نرم­افزارهای مدیریت ویدئویی بپردازیم، درک ارزش داشتن یک راه حل یکپارچه در مقایسه با دوربین های مجزا بسیار مهم است. چرا که نرم افزاری که قصد دارد اطلاعات تمامی دوربین ها دریافت کند، می­بایست تحت یک پروتکل استاندارد اطلاعات را از دوربین ها دریافت کند و سرویس هایی از این جنس به کاربران خودش ارائه دهد، جزئیات این مسئله را در بخش های بعدی خواهیم دید ولی اکنون به تعدادی از نرم افزارهای مدریت تصویر میپردازیم و هر کدام را به صورت مختصر بررسی میکنیم.

### نرم افزار مدریت دوربین Cisco Meraki

Cisco Meraki طیف گسترده ای از راه حل های مرتبط با ویدئو را برای امنیت شما ارائه می دهد اما همه این راه‌حل‌ها یک وجه مشترک دارند، این نرم‌افزار دارای تجزیه و تحلیل ویدئویی قدرتمند با هوش مصنوعی است و از ویژگی های تشخیص حرکت و جستجوی حرکت را پشتیبانی می­کند. به طور مثال این نرم افزار این امکان را به شما می دهد که به گذشته مناطقی که عبور کرده اید مراجعه کنید و نقطه ی دقیق گم شدن کلید خود بشوید. این­ ویژگی ای است که قطعاً نمی توانید با نرم افزار رایگان مدریت تصویر دوربین به دست آورد.

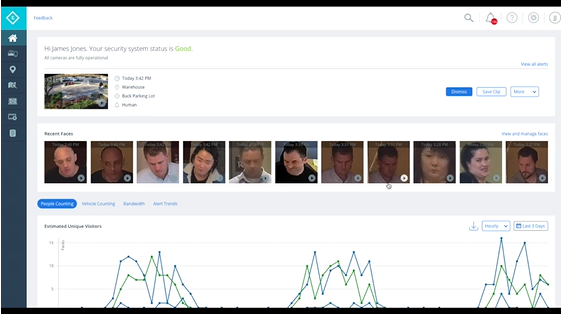
در این نرم افزار همه دوربین های شما به فضای ابری متصل خواهند شد و به هر دوربین با بیش از 500 گیگابایت حافظه تخصیص داده می­شود، این امکان میتواند باعث صرفه جویی در هزینه ها شود زیرا دیگر نیازی به NVR وجود نخواهد بود. Meraki همچنین با بسیاری از راه حل ها از جمله اپلکیشن موبایل ادغام شده تا یک راه حل امنیتی جامع برای دفتر یا استودیو شما ایجاد کند که شامل کنترل دسترسی و نظارت تصویری است. در تصویر شکل ‏1–1 نمونه ای از پنل این نرم افزار قابل مشاهده است که یک نمودار تجمیعی از مکان های مورد تحرک در طول هفته استخراج شده است. که قابل مشاهده است که در روز های شنبه و یکشنبه میزان تردد از راه پله ها به شدت کاهش یافته است.



شکل ‏1–1- تصویری نمونه از نرم افزار مدریت دوربین ‏ Cisco Meraki

### نرم افزار مدریت دوربین Rhombus

مشابه Meraki، Rhombus یک راه حل امنیتی جامع با دوربین های خود ارائه می دهد. نصب این راه حل ساده تر است و این نرم افزار دوربین ها را از طریق اینترنت به یکدیگر متصل می کند. Rhombus دارای یک داشبورد بسیار شکیل است که می‌توانید از طریق آن بر آنچه در فضای خود اتفاق می‌افتد نظارت کنید. از لحاظ میزان امکان تجزیه و تحلیل، Rhombus هشدارهای مفیدی را با فیلم های فیلتر شده برای حرکت، چهره ها، رویدادهای حسگر، رفتار غیر معمول و بسیاری موارد دیگر ارائه می دهد. این نرم افزار علاوه بر مزایای ابری (از جمله ذخیره‌سازی ویدیوها)، ویژگی‌های نوآورانه زیادی مانند تشخیص رفتار غیرعادی، تشخیص چهره، و تشخیص پلاک دارد. Rhombus، از مزیت ادغام با چندین راه حل، از جمله Kisi[[1]](#footnote-1)، برای ایجاد زیرساخت امنیتی کامل تری برای شرکت شما برخوردار است. این کار دفتر شما را به یک فضای امن با فناوری پیشرفته تبدیل می کند. در تصویر شکل ‏1–2 چهره هایی که توسط این نرم افزار در طول روز در یک دوربین ذخیره شده اند نمایش داده شده است.



شکل ‏1–2- تصویر

Blue Iris Software

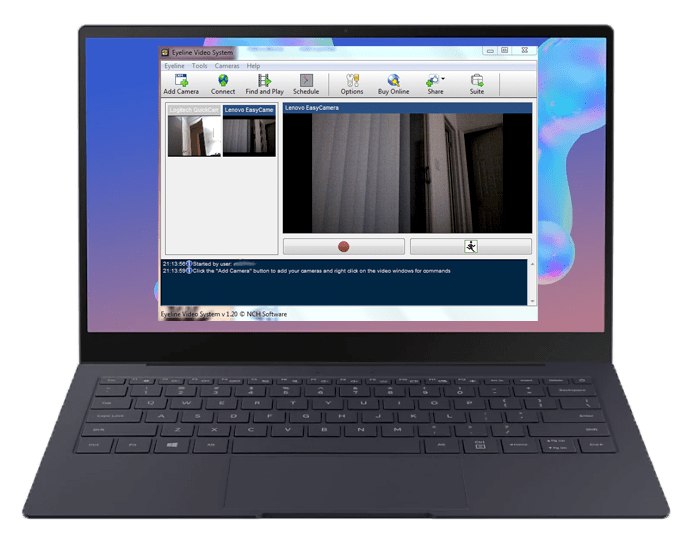
Blue Iris یک سیستم نظارت تصویری مبتنی بر ابر محبوب است که به هر کسی اجازه می‌دهد فید دوربین را از راه دور مشاهده و کنترل کند. این نرم افزار با مجموعه عظیمی از دوربین های بازار سازگار است، اما به صورت عمومی به این گونه شناخته شده که با دوربین های HD اجرا می شود. نسخه رایگان Blue Iris نیازی به لایسنس اضافی برای اضافه کردن چندین دوربین ندارد و به راحتی می توان همه دوربین ها را در یک سیستم واحد دریافت کرد اما برای فعال شدن بخش پردازش و ذخیره سازی اطلاعات در این نرم افزار به لایسنس نیاز دارد. این نرم افزار حجم زیادی از گزینه های ویرایش و پخش ویدیو را ارائه می دهد، اما برخی از مشتریان اشاره کرده اند که اغلب شکاف های زمانی و تاخیر وجود دارد. در تصویر شکل ‏1–3 میتون نمونه داشبورد اصلی این نرم افزار را مشاهده نمود. که تصاویر دوربین­ها در بخش های کوچکی مشخص تعیین میشوند.



شکل ‏1–3

### نرم افزار مدریت ویدیویی EyeLine

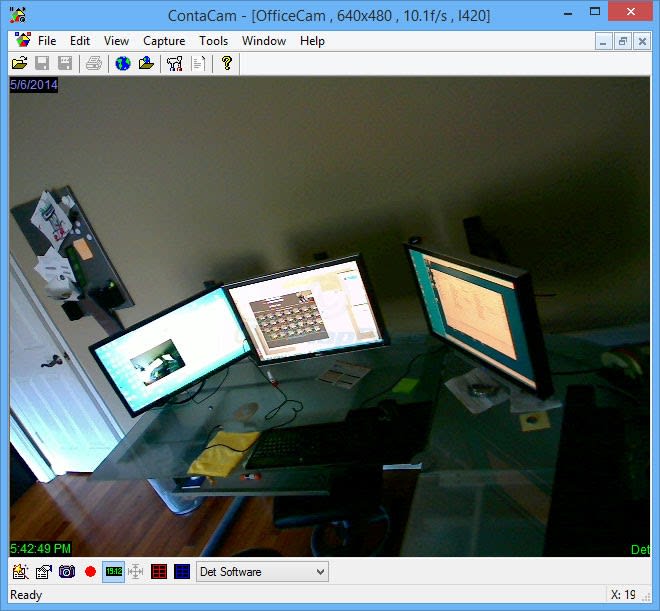
نرم افزار مدیریت ویدئو EyeLine یک گزینه جذاب برای مشتریانی است که به دنبال یک گزینه نرم افزار امنیتی هستند که از دوربین های زیادی پشتیبانی می کند. EyeLine می تواند بیش از 100 منبع دوربین را به طور همزمان نظارت و ضبط کند. این نرم افزار به شما این امکان را می دهد که فیلم را در زمان واقعی در محل یا آنلاین با ورود از راه دور تماشا کنید. مؤلفه تشخیص حرکت تنها با ضبط فیلم زمانی که اتفاقی در دوربین روی می‌دهد، فضا را ذخیره می‌کند. برخی از مشتریان با پشتیبانی از دوربین های خاص مشکل داشتند، بنابراین توصیه می شود از وب کم های متصل به USB استفاده کنید.



### 5-

ContaCam Video Management Software

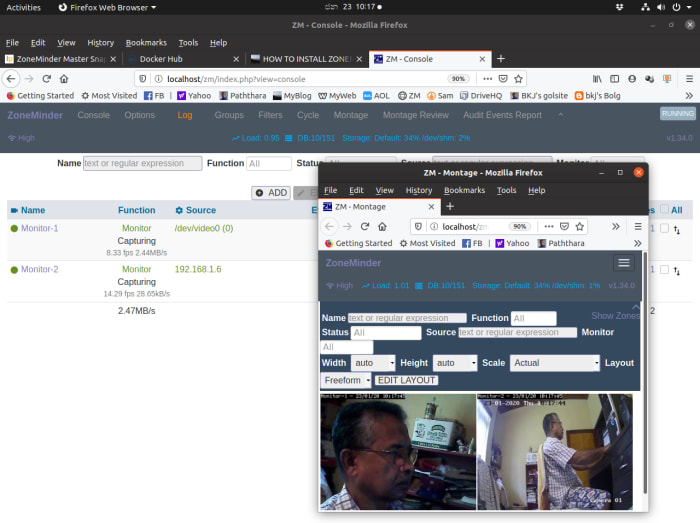
نرم‌افزار ContaCam خود را به‌اندازه کافی سریع، سبک و همه‌کاره می‌داند تا بتواند نیازهای امنیتی را برآورده کند. راه‌اندازی دوربین‌ها آسان است، و از کاربران می‌خواهد که به سادگی بین تشخیص حرکت یا ضبط مداوم تصمیم بگیرند، نام دوربین را وارد کنند و مدت زمان نگهداری فایل‌ها را پیکربندی کنند. این نرم افزار یک گزینه ایجاد ویدئو خلاصه روزانه را ارائه می دهد و به خوبی با فناوری تشخیص پلاک ادغام شده است. ContaCam به داشتن عملکرد پایدار شناخته شده است اما گاهی اوقات هنگام جفت شدن با دوربین های IP مشکلاتی دارد.



### 6-

ZoneMinder

ZoneMinder یک گزینه محبوب برای امنیت خانه و کسب و کار است. راه اندازی این نرم افزار بسیار آسان است و با طیف گسترده ای از دوربین های امنیتی کار می کند. مشتریان پاسخ‌های منفی کمی به این نرم‌افزار دارند، زیرا این نرم‌افزار چیزی بیش از یک نرم‌افزار واحد است - این یک مجموعه کامل از برنامه‌ها است. این برنامه ها با هم کار می کنند تا نظارت کسب و کار با کیفیت را ارائه دهند. ZoneMinder به خوبی طراحی شده است و یافتن سریع آنچه به دنبال آن است را برای هر فردی آسان می کند.



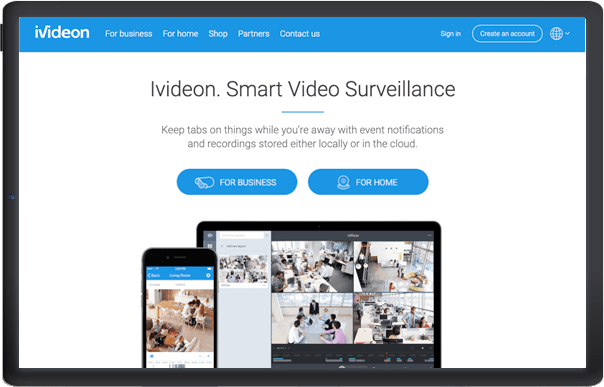
7-

### Ivideon Server

Ivideon به عنوان بهترین نرم افزار ضبط دوربین IP رایگان موجود در نظر گرفته می شود. این نرم افزار به طور مناسب و سریع به کاربران از هر گونه صدا یا اتفاق غیرعادی هشدار می دهد. الگوریتم‌های درون سرور حرکاتی را به ظرافت مانند سوسو زدن در وسایل روشنایی تشخیص می‌دهند که سیستم‌های نرم‌افزار نظارت معمولی معمولاً آن را انجام نمی‌دهند. نرم افزار پایه رایگان است، اما مشتریان برای استفاده از سرویس ابری باید مشترک شوند. علاوه بر این، نرم افزار تمایل دارد به خوبی عمل کند، اما برخی از مشتریان ادعا می کنند که در مقایسه با رقبا کمی ابتدایی است.

8-

Luxriot Evo

Luxriot Evo یک نرم‌افزار مدیریت ویدئو کاملا رایگان است که بر ارائه نرم‌افزارهای با عملکرد برتر و همچنین خدمات به مشتریان تمرکز دارد. این نرم افزار دارای تعدادی گزینه ویرایش از ویرایش صدا گرفته تا دقت فریم و جلوه های دیجیتال است و حتی دارای گزینه پخش مستقیم ویدیو به YouTube Live است. اگرچه این نرم‌افزار تقریباً با هر دوربین IP کار می‌کند، اما نمی‌تواند بیش از 10 کانال را پشتیبانی کند، بنابراین برای یک دفتر یا تجارت کوچک بهترین است. 

### جمع بندی

از میان نرم افزارهای فوق که بررسی شد، میتوان دریافت که نرم­افزارهایی که در حوزه معرفی شده اکثرا اطلاعات را در سرورهای خود ذخیره و پردازش میکنند که در نمیتوان از این ویژگی در مکان های امنیتی و نظامی یک کشور استفاده نمود. همچنین هیچ یک از نرم افزار های فوق اجازه پیاده سازی الگوریتم هوش مصنوعی شخصی سازی شده ارائه نمیدهند که بتوان آن را متناسب با نیاز تغییر و استفاده نمود.

## طرح مسئله و پیش فرض ها

در بخش های پیش نرم افزارهای مدریت دوربین های متفاوتی را بررسی نمودیم و دریافتیم که یکی از چالش های اصلی در جمع آوری اطلاعات دوربین ها این است که دوربین ها را شرکت های متفاوتی تولید میکنند و برای دریافت اطلاعات دوربین ها نیاز داریم که از نقطه ای مشخص یا از استانداردی واحد استفاده کنیم، در این بخش ابتدا استاندارد اصلی که شرکت های سازنده دوربین موظف اند پیاده سازی کنند بررسی میکنیم و سپس به طرح مسئله و راه حل پیشنهادی میپردازیم. در حقیقت تنها پیش فرض اصلی ما رجوع به نقطه مشترک دوربین ها یعنی استاندارد ONVIF است.

### استاندارد ONVIF

هر کسی که با دوربین های امنیتی کار کرده، حتما یک بار اسم ONVIF(Open Network Video Interface Forum) رو شنیده، ONVIF یک انجمن صنعتی است که رابط­های استاندارد، برای همکاری موثر، میان محصولات امنیت فیزیکی مبتنی بر شبکه را فراهم می­کند و ارتقا می­بخشد.

سنگ بناهای ONVIF عبارتند از:

• استاندارد سازی ارتباط میان محصولات امنیتی فیزیکی مبتنی بر شبکه

• قابلیت تعامل دستگاه های متفاوت بدون توجه به مارک تجاری

• در اختیار بودن برای همه شرکت ها و سازمان ها

ONVIF در سال 2008 توسط Axis Communications ، Bosch Security Systems و Sony Corporation تاسیس شد و دارای یک عضو قوی از هر شش قاره است. عضویت در ONVIF برای تولیدکنندگان ، توسعه دهندگان نرم افزار ، مشاوران ، مجتمع های سیستم ، کاربران نهایی و سایر گروه های علاقه مند که مایل به شرکت در فعالیت های ONVIF هستند ، رایگان است.

اعضای ONVIF، امروزه بیش از 18000 محصول سازگار با یکدیگر را بصورت جمعی ارائه می دهند. این امر منجر به نصب میلیون ها محصول سازگار ONVIF در بازار شده است. هر محصول باید حداقل یکی از پروفایل­ها راپشتیبانی کند:

• پروفایل A برای پیکربندی کنترل دسترسی­ها

• پروفایل C برای کنترل ورود خروج و مدیریت رویدادها

• پروفایل G برای ذخیره سازی و اطلاعات

• پروفایل Q برای نصب سریع

• پروفایل S برای پخش جریانی اصلی دوربین ها به صورت زنده

• پروفایل T برای پخش جریانی پیشرفته از اطلاعات دوربین ها به صورت زنده

مزایای ONVIF برای کاربران نهایی زیاد است چرا که ، به تهیه کنندگان ،اجازه انتخاب محصولات با انعطاف پذیری بیشتر را میدهد همچنین محصولاتی که توانایی قصد انتخاب آنها را دارند افزایش می­یابد که منجر به رقابتی شدن بازار و کاهش قیمت ها می­شود. ONVIF به شما امکان انتخاب و انتخاب محصولات بهینه برای نیازهای شما را می دهد بدون اینکه در یک برند خاص قفل شوید. به عنوان مثال ، یک نرم افزار مدیریت فیلم سازگار با ONVIF ، به شما امکان می دهد دستگاه های سازگار با ONVIF را از سازندگان مختلف سخت افزار ادغام کنید.حال سوال که مطرح است این است که دوربین های مورد استفاده در این پروژه چه ویژگی خاصی باید داشته باشند؟ هدف اصلی پیاده سازی این پروژه دریافت، آنالیز، ذخیره سازی و اجرای زنده جریان اطلاعات دوربین های نظارتی است پس دوربین هایی در این مجموعه استفاده می­شوند باید در پروفایل S و T از سری استاندار های ONVIF باشند. در ادامه این دو پروفایل را معرفی خواهند شد و ویژگی مشترکی که در اختیار ما قرار میدهند را بررسی می­گردد.

**Profile-S : برای پخش جریان اصلی دوربین ها به صورت زنده**

Profile S برای سیستم های ویدیویی مبتنی بر IP طراحی شده است. یک دستگاه Profile S (به عنوان مثال دوربین شبکه IP ) دستگاهی است که می تواند داده های ویدئویی را از طریق شبکه IP به یک سرویس گیرنده Profile S ارسال کند. سرویس گیرنده Profile S (به عنوان مثال ، یک نرم افزار مدیریت ویدیو) برنامه ای است که می تواند پخش ویدئو را از طریق یک شبکه IP از طریق دستگاه Profile S پیکربندی ، درخواست و کنترل کند. Profile S همچنین مشخصات ONVIF را برای کنترل PTZ ، ورودی صدا ، پخش چندمنظوره و رله برای دستگاه های سازگار و سرویس گیرنده هایی که از چنین ویژگی هایی پشتیبانی می کنند ، پوشش می دهد.

**Profile-T : برای پخش جریانی پیشرفته از اطلاعات دوربین ها به صورت زنده**

Profile T برای سیستم های ویدیویی مبتنی بر IP طراحی شده است. Profile T از ویژگی های پخش جریانی ویدئو مانند استفاده از قالب های رمزگذاری H.264 و H.265 ، تنظیمات تصویربرداری و رویدادهای زنگ مانند تشخیص حرکت و دستکاری پشتیبانی می کند. ویژگی های اجباری برای دستگاه ها همچنین شامل نمایش روی صفحه و پخش فراداده هستند. Profile T از مشخصات ONVIF برای استفاده در جریان HTTPS ، پیکربندی PTZ ، پیکربندی منطقه حرکت ، ورودی های دیجیتال و خروجی های رله، و همچنین صدای دو طرفه برای دستگاه های سازگار پشتیبانی می کنند.

همانطور که مشاهده میشود دروبین هایی با پروفایل T از امکانات بیشتری نسبت به دوربین هایی با پروفایل S برخوردار هستند و می­توانند اطلاعات خود را رمزنگاری شده و با کیفت بالاتری ارائه دهند وهمچنین امکان ایجاد ارتباط دو سویه در این پروفایل فراهم شده[[2]](#footnote-2)، ولی دلیل اصلی که نام این دو پروفایل را پخش زنده جریان نامیده اند، امکان پیاده­سازی پروتکل RSTP است که با کمک آن میتوان جریان اطلاعات دوربین ها را مدریت کرد. در ادامه بیشتر با این پروتکل آشنا می­شویم.

**پروتکل RTSP**

پروتکل جریان بخشی در زمان واقعی (RTSP[[3]](#footnote-3)) یک سیستم ارتباطی شبکه در لایه برنامه کاربردی است که با برقراری ارتباط مستقیم با سرویس دهنده، جریان داده های چندرسانه ای را به دستگاه دیگری منتقل می کند.

این پروتکل جریان چندرسانه­ای بین دستگاه­های سرویس دهنده و سرویس گیرنده را با استفاده از یک کنترل از راه دور شبکه برای همگام سازی جریانی، رسانه های مداوم مانند صدا و تصویر ، ایجاد و کنترل می کند. این پروتکل خود پیام های چندرسانه ای را منتقل نمی­کند بلکه با سرویس دهنده ای که اطلاعات چندرسانه ای(مانند تصویر دوربین) را پخش می کند ارتباط برقرار می کند. برای مثال وقتی کاربر، ویدیویی را که در حال پخش است قطع می­کند، RTSP درخواست کاربر را برای مکث فیلم به سرور پخش جریانی ویدئو منتقل می کند، و پخش ویدیو از سرویس دهنده متوقف می­شود.

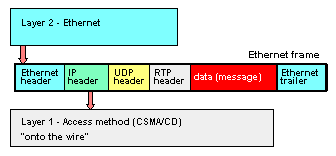
وقتی کاربر یا برنامه ای تلاش می کند ویدئو را از یک منبع از راه دور پخش کند، دستگاه سرویس گیرنده درخواست RTSP را سرور ارسال می­کند تا گزینه های موجود، مانند مکث ، پخش و ضبط را تعیین کند. پس سرویس دهنده لیستی از انواع درخواست هایی را که می تواند از طریق RTSP بپذیرد، برمی­گرداند. هنگامی که سرویس گیرنده می­داند چگونه درخواستش را ارسال کند، یک درخواست توصیف رسانه را به سرویس­دهنده پخش جریانی منتقل می کند و سرویس دهنده با توصیف رسانه پاسخ می دهد و هنگامی که سرویس گیرنده درخواست راه اندازی را ارسال می کند و سرور با اطلاعات مربوط به مکانیزم حمل و نقل بسته ها پاسخ می دهد. پس از اتمام مراحل نصب ، سرویس گیرنده با ارسال مکانیزم انتقال مشخص شده، درخواست راه اندازی، جریان پخش را با گفتن اینکه سرور برای ارسال شروع می­شود.

RTSP به عنوان راهی برای اجازه دادن به کاربران برای پخش صوتی و تصویری مستقیم از اینترنت و نه بارگیری فایل های رسانه ای در دستگاه های خود ، شروع به کار کرد. این پروتکل برای مصارف مختلف از جمله سایت های دوربین اینترنتی ، آموزش آنلاین استفاده شده است.

RTSP از مفاهیم مشابه HTTP پایه استفاده می کند ، که آن را به راحتی با شبکه های HTTP موجود سازگار می کند. این پروتکل همچنین امکان انعطاف پذیری زیادی را فراهم می کند تا جایی که سرویس گیرنده­ها می توانند ویژگی هایی را که می­خواهند استفاده کنند را درخواست کنند، تا بفهمند آیا سرور رسانه از آنها پشتیبانی می­کند یا خیر. به همین ترتیب ، هر کسی که رسانه دارد می تواند جریانهای رسانه ای را از چندین سرور ارائه دهد. این پروتکل همچنین برای انطباق با رشد رسانه ها در آینده طراحی شده است تا در صورت لزوم ، سازنده رسانه بتواند پروتکل را به بهبود دهد.

همانطور که پیشتر گفته شد RSTP یک پروتکل لایه کاربردی است که برای انتقال جریان صوتی ، تصویری و انیمیشن سه بعدی از طریق اینترنت استفاده می شود. این نرم افزار سرویس گیرنده کاربر را قادر می سازد تا کنترل از راه دور سرور را با عملکردهایی مانند مکث ، عقب و سریع جلو برد. RTSP توسط Netscape و Progressive Networks ساخته شده است و به طور گسترده ای همراه با پروتکل حمل و نقل بسته ی RTP استفاده می شود. RTSP برای کنترل جلسه[[4]](#footnote-4) RTP در لایه برنامه برنامه کاربردی استفاده می شود. با کمک این پروتکل، امکان اجرای توابعی مانند مکث ، عقب و جلو برد سریع، و... در نرم افزار سرویس گیرنده ، فراهم می­شود.

در شکل زیر نمونه ای از بسته مربوط به پروتکل حمل و نقل بیدرنگ (RTP) قابل مشاهده است، این پروتکل یک پروتکل بر بستر شبکه است که از انتقال صدا و فیلم در زمان واقعی پشتیبانی می کند. یک بسته RTP بر روی یک ارتباط UDP ، سوار می شود و شامل اطلاعات زمان سنجی و هماهنگ سازی در هدر آن برای مونتاژ مناسب در انتهای دریافت می باشد.[[5]](#footnote-5)



### راه حل پیشنهادی

با توجه به پیش فرض فوق که دوبین ها از پروتکل RTSP پشتیبانی کنند، میتوان برنامه ای نوشت که اطلاعات را از دوربین ها با پروتکل فوق دریافت کند. همچنین با دریافت اطلاعات دوربین ها میتوان آن ها را ذخیره نمود و آن ها را در شبکه به کاربر ارسال کرد. پس حداقل باید یک برنامه بر روی کامپیوتر میزبان وجود داشته باشد که اطلاعات را از دوربین ها دریافت کند و یک برنامه در سمت کابر اجرا شود که اطلاعات دوربین ها دریافت کند. اما راهکار برای پردازش چیست، اولین و کارآمدترین راهکاری که به ذهن می­آید این است که اطلاعات دریافت شده از دوربین ها علاوه بر کاربر، به یک برنامه کاربردی دیگر ارسال کرد که فرایند پردازش را انجام دهد و اطلاعات پردازش شده را به کاربر ارسال کند و کاربر تصاویر اصلی دوبین ها و اطلاعات پردازش شده تصاویر را دریافت کند. پس به سه برنامه کاربردی نیاز داریم که به شرح زیر است:

* برنامه کاربردی سرور: برنامه ای که اطلاعات دوربین ها را دریافت، ذخیره کرده و کاربر و برنامه سرور پردازشی ارسال میکند.
* برنامه کاربردی سرور پردازشی: برنامه ای که اطلاعات دوربین ها را دریافت، پردازش کرده و به کاربر ارسال میکند.
* برنامه کاربر: برنامه ای که اطلاعات خام و پردازش شده دوربین ها را دریافت به نمایش می­گذارد و همچنین به اطلاعاتی که برنامه سرور ذخیره کرده ، دسترسی پیدا میکند و آن ها را به نمایش میگذارد.

از آن جایی که این برنامه قرار است در یک محیط امنیتی اجرا شود و کاربران نباید به اطلاعات تصاویر هر نقطه ای دست رسی داشته باشند، پس این برنامه به یک ویژگی اساسی نیاز دارد و آن کنترل بندی مناطق و تعیین سطح دسترسی برای کاربران است که مدیر سیستم بتواند با کنترل و تعیین مناطق و تعیین سطح دسترسی کابران امنیت مورد رضایت را تایمن نماید.

چنین سیستمی نیازمند پیاده سازی یک معماری منسجم است که بتوان عملکردهای فوق را در شبکه و با تامین امنیت کافی پیاده سازی نماید. هدف اصلی این پروژه ارائه این معماری برای توسعه الگوریتم های هوش­مصنوعی است که از مدل­ها و الگوریتم های هوش مصنوعی که امروزه با سرعت زیادی در حال پیشرفت و توسعه هستند بتوان استفاده نمود و بتوان سیستم هایی هوشمند که قابلیت پیاده­سازی سناریوهای دلخواه را داشته باشند طراحی نمود.

## ساختار پایان نامه

در این رساله، که در پنج فصل تدوین شده، در ابتدای هر فصل ابتدا مروری بر مفاهیم کلی، نرم‌افزارهای به‌کاررفته، انجام شده است و در ادامه جزئیات در بخش­ها و طراحی آن را شرح داده خواهد شد.در فصل دوم سرویس های مورد استفاده در این پروژه، نحوه بکارگیری آن ها و ویژگی هایی که در اختیار ما میگذارند تشریح می­شود و در انتها نحوه چینش معماری با کمک سرویس های فوق بررسی می­شود.در فصل سوم به نحوه طراحی هر یک برنامه کاربردی سرور، سرور پردازشی و برنامه کاربردی کاربر، پرداخته میشود و سعی میشود جزئیات هر بخش را منتقل کرد و در فصل چهارم به نحوه بکارگیری و استفاده از سیستم فوق در یک شبکه پرداخته می­شود. در فصل پنجم نیز به نتیجه­گیری و پیشنهادهایی که در ادامه راه این پروژه می­تواند انجام شود، پرداخته می­­شود.



# فصل دوم سرویس های مورد استفاده

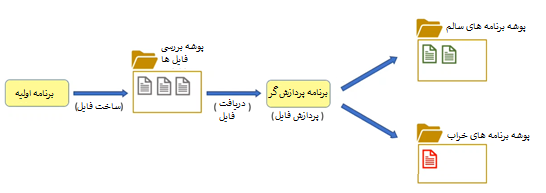
**پیاده­سازی سیستم دریافت و ذخیره­سازی اطلاعات**

در این بخش ابتدا به معرفی بستر اجرای سرویس های مورد استفاده در این پروژه میپردازیم و سعی می­کنیم با ابزارهای یکپارچه‌سازی بستر شبکه بتوانیم، شرایطی برای دریافت و انتقال و ذخیره‌سازی اطلاعات فراهم کنیم سپس معماری مورد استفاده را شرح خواهیم داد.

## مفهوم یکپارچه­سازی

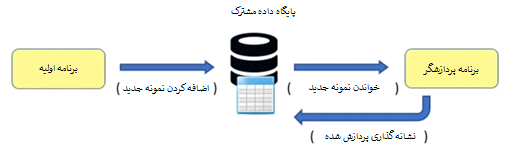
یک سیستم کامل، از چند برنامه تشکیل‌شده که وظایف متفاوتی را انجام می­دهند، در اغلب مواقع این برنامه‌ها نیاز دارند با یکدیگر تعامل داشته باشند و باهم ارتباط برقرار کنند. به ایجاد کردن ارتباط میان برنامه‌های متفاوت یکپارچه‌سازی[[6]](#footnote-6) می­گویند. شیوه‌های متفاوتی از یکپارچه­سازی وجود دارد.

مدل اول یکپارچه­سازی مبتنی بر فایل است. در این مدل، برنامه اولیه فایلی که باید پردازش شود را ایجاد می­کند و در پوشه‌ای مشخص قرار می­دهد، سپس برنامه‌ای دیگر فایل‌هایی که در این پوشه قرارگرفته‌اند را بررسی می­کند. برای مثال برنامه دوم و فایل­های خراب را از سالم جدا کرده و در پوشه‌هایی متفاوت قرار می­دهد. قابل‌مشاهده است که این دو برنامه می­توانند مستقل از هم کار کنند و حتی می­توانند با زبان‌های برنامه‌نویسی متفاوتی ایجادشده باشند. در شکل ‏2–1می­توان مثال فوق را مشاهده کرد.[[7]](#footnote-7)



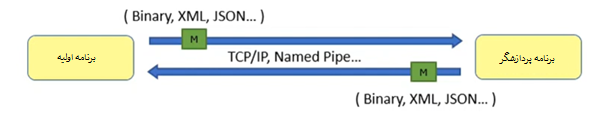
شکل ‏2–1-مثالی از یکپارچه‌سازی مبتنی بر فایل

مدل دوم یکپارچه­سازی­هایی مبتنی بر پایگاه­داده مشترک یا توزیع‌شده است. در این مدل ابتدا یکی از برنامه­ها اطلاعاتی را در پایگاه­داده تغییر می­دهد. سپس برنامه دیگری آن را برداشته و پردازش می­کند و مجدداً به پایگاه­داده برمی­گرداند و به آن برچسب پردازش‌شده می­زند. به‌طور مشابه، این دو برنامه می­توانند مستقل از یکدیگر فعالیت کنند. در شکل شکل ‏2–2 می­توان مدل فوق را بررسی نمود.[[8]](#footnote-8)



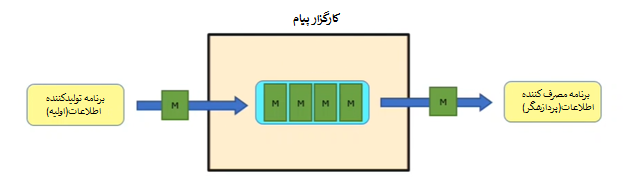
شکل ‏2–2-مثالی از یکپارچه­سازی مبتنی بر پایگاه­داده

مدل سوم یکپارچه­سازی­هایی مبتنی بر ارتباط مستقیم است، به‌نحوی‌که برنامه‌ها به‌صورت مستقیم با یکدیگر ارتباط tcp/ip برقرار کرده و بعد از برقراری ارتباط شروع به ارسال پیام به یکدیگر می­کنند. فرمت پیام‌ها می­تواند به گونه‌های متفاوتی ازجمله باینری یا مبتنی برنوشته مانند, xml, json ... باشد. در شکل ‏2–3 می­توان نمونه‌ای از این ارتباط را مشاهده کرد.[[9]](#footnote-9)



شکل ‏2–3-نمونه از یکپارچه­سازی بر اساس ارتباط مستقیم میان برنامه‌ها

مدل چهارم یکپارچه­سازی بر پایه پیام دادن نامتقارن با کمک یک کارگزار پیام[[10]](#footnote-10) است. برنامه اولیه یا تولیدکننده اطلاعات در یک‌سو با هر فرمتی که بخواهد می­تواند، پیام ارسال کند ولی این بار یک برنامه میانی پیام‌ها را دریافت می­کند (که اغلب به آن message broker یا message bus می­گویند) پیام‌ها را در یک لیست قرار داده که به آن Queue یا صف می­گویند و آن‌ها را به هدف، مقصد یا مصرف‌کننده منتقل می­کند. در شکل ‏2–4 می­توان نمونه‌ای از این شیوه انتقال اطلاعات را مشاهده کرد. این روش، یک روش محبوب است که در سیستم­های متفاوتی می­تواند مورداستفاده قرار بگیرد.



شکل ‏2–4-نمونه‌ای از یکپارچه­سازی با کمک یک کارگزار پیام

این شیوه از یکپارچه‌سازی مزایای بسیاری دارد که به شرح زیر است:

* جدا بودن برنامه تولیدکننده از برنامه مصرف‌کننده:
  + برنامه تولیدکننده اطلاعات و برنامه مصرف‌کننده نیازی ندارد یکدیگر را بشناسند.
  + آدرس و تکنولوژی‌های مورداستفاده یکدیگر را نمی­دانند.
  + تنها چیزی که نیاز است نسبت به هم آگاه باشند، فرمت اطلاعات ارسالی و دریافتی است.
* ایجاد محیطی مطمئن، قابل‌اعتماد برای انجام پردازش‌ها و ارتباط میان برنامه­ها:
  + برنامه‌های تولیدکننده اطلاعات می­تواند به ارسال اطلاعات بپردازد، درحالی‌که هیچ برنامه‌ای در طرف دیگر، اطلاعات را مورداستفاده قرار نمی­دهد، به این معنا که اطلاعات در برنامه میانی به‌صورت موقت ذخیره می­شوند و در ادامه هرگاه که برنامه مصرف‌کننده اطلاعات، فعالیت خود را آغاز کرد، می­تواند از اطلاعات استفاده نماید.
  + درصورتی‌که برنامه مصرف‌کننده در پردازش شکست بخورد، می­تواند درخواست ارسال مجدد داشته باشد.
* فراهم کردن راهی برای پیاده­سازی معماری­هایی افقی در تعداد بالا
  + اگر برنامه مصرف‌کننده اطلاعات به‌تنهایی نتواند همه پیام‌ها را پردازش کند، می­توان چندین مصرف‌کننده اطلاعات را در کنار هم قرارداد تا فرایند سریع­تر انجام شود.

این شیوه بسیار پربازده‌تر از مدل‌های یکپارچه­سازی مبتنی بر پایگاه­داده و عموما برای جریان­داده مورد استفاده قرار میگیرد[[11]](#footnote-11).

## بستر اجرای سرویس ها (داکر)

به‌طورکلی داکر[[12]](#footnote-12) یک سکو برای توسعه، انتقال و اجرای برنامه‌ها است. با کمک داکر می­توانیم نرم­افزار­ها را مستقل از زیرساخت به‌سرعت پیاده‌سازی کنیم. همچنین با کمک این نرم‌افزار می­توان زیرساخت و سخت‌افزار مورداستفاده در پروژه‌ها را مدیریت کرد و فاصله زمانی میان پیاده‌سازی و رسیدن به مرحله محصول نهایی را به حداقل رساند.

داکر توانایی بسته‌بندی و اجرای برنامه­های ما را در محیط ایزوله‌ای به نام کانتینر[[13]](#footnote-13) فراهم می­کند. این ایزوله شدن شرایطی ایجاد می­کند که کانتینرها را بتوان به‌صورت هم‌زمان روی کامپیوتر میزبان اجرا کرد. این کانتینرها در مقابل ماشین‌های مجازی بسیار سبک هستند و می­توانند بدون نظارت بر روی کامپیوتر میزبان اجرا شوند. به‌طورکلی داکر ابزار و سکویی برای مدیریت کردن این کانتینرها است تا توسعه سریع و ساده­تر شود و بتوان پردازش قابل توسعه­ای، روی خوشه‌ای از کامپیوترها اجرا کرد. همچنین هنگامی‌که برنامه کامل شد و به مرحله محصول نهایی رسید، بدون در نظر گرفتن نوع زیرساخت، قابل‌انتقال به محیط‌های دیگر (مرکز داده‌ها، شرکت‌های خدمات دهنده ابری،...) باشد.

به‌طورکلی هسته اصلی داکر از سه بخش تشکیل‌شده که در شکل ‏2–5 قابل‌مشاهده است، بخش اول یک سرویس‌دهنده است که مدیریت اشیا ساخته‌شده توسط داکر را بر عهده دارد، اشیائی همچون فایل ایمیج (فشرده‌شده کانتینرها)، کانتینرها، شبکه‌ها و فضاهایی که تخصیص داده می­شوند. این برنامه فرایند محاسباتی[[14]](#footnote-14) نام‌گذاری می­شود. بخش دوم یک برنامه رابط است (REST API) که دستوراتی که ما می­دهیم را به فرایند محاسباتی منتقل می­کند. بخش سوم یک رابط و خط دستور (CLI)[[15]](#footnote-15) است که فرمان‌های ما در آن وارد می­شود. خط دستور از یک REST API استفاده می­کند تا وظیفه کنترل و ارتباط با فرایند محاسباتی را یا با استفاده از فرمان­هایی از پیش نوشته‌شده یا به‌صورت واردکردن در خط دستور انجام دهد.



شکل ‏2–5-موتور و هسته داکر که روند فرایند مدیریت برنامه را مشخص می­کند. [2]

به سه دلیل اصلی در این پروژه برای پیاده‌سازی سرویس ها از داکر استفاده شد که به شرح زیر است:

پیشرفت و توسعه مداوم برنامه‌ها؛

داکر با کار در محیط‌های استاندارد و با استفاده از کانتینرهای محلی این اجازه را به توسعه‌دهندگان برای پیاده­سازی برنامه‌های پیچیده می­دهد و چرخه عمر توسعه را ساده می­کند. چراکه بخش‌های مختلف برنامه می­توانند مستقل از هم با نسخه‌ها و حتی زبان­های برنامه‌نویسی متفاوت پیاده­سازی و اجرا شوند. به این عمل فرایند یکپارچه‌سازی پیوسته[[16]](#footnote-16)/ تحویل پیوسته[[17]](#footnote-17) می­گویند.

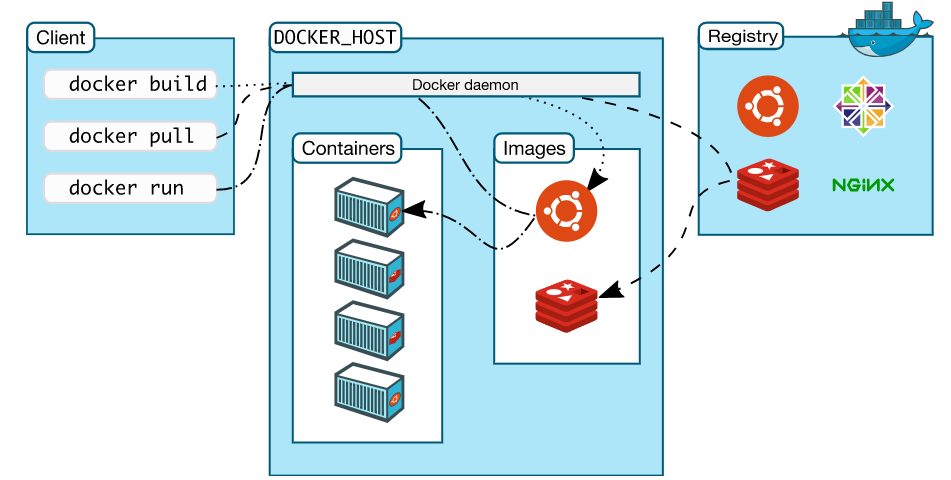
بستر مبتنی بر کانتینرهای داکر، ایجاد برنامه‌هایی قابل‌حمل[[18]](#footnote-18) را امکان‌پذیر می‌کند. کانتینرهای داکر می‌توانند بر روی کامپیوتر محلی توسعه‌دهنده، در ماشین‌های فیزیکی یا مجازی در یک مرکز داده، بر روی ارائه‌دهندگان خدمات ابری یا در مخلوطی از این محیط‌ها اجرا شوند که کمک می­کند برنامه به‌سادگی روی کامپیوتر توسعه‌دهنده آماده و سپس برای پیاده‌سازی عملی به سرورهایی باقدرت و پاسخ‌گویی بیشتر منتقل شود.

کانتینرهای داکر سبک و سریع هستند چراکه تنها برنامه‌هایی که موردنیاز است را در خود جای‌داده‌اند. این‌یک جایگزین مناسب و مقرون‌به‌صرفه به‌جای ماشین‌های مجازی است که نیازمند پیش‌نیازهای بیشتری برای پیاده‌سازی محیط‌های ایزوله هستند؛ بنابراین می­توان از ظرفیت محاسبه بیشتری برای رسیدن به اهداف خود استفاده کرد. داکر برای برنامه‌هایی با چگالی بالا (بخش‌های متفاوت) و عملکردهای کوچک و متوسط که نیاز به انجام کارهای بیشتر با منابع کمتری دارند، بسیار مناسب است.

داکر از یک معماری کاربر-سرویس­دهنده­ای استفاده می­کند. به این شکل که کاربر داکر، با فرایند محاسباتی در ارتباط است که وظیفه ساخت و اجرا و توزیع کانتینرها را بر عهده دارد. کاربر داکر و فرایند اصلی می­توانند در یک کامپیوتر باشند یا می­توانند از طریق REST API یا رابط­های شبکه با یکدیگر در ارتباط باشند. به‌طورکلی برنامه‌ها در محیطی به نام کانتینر اجرا می­شوند که نمونه‌ای قابل‌اجرا از فایلی به نام ایمیج است. برای ساخت فایل ایمیج یا باید خودمان آن فایل را از ابتدا ایجاد کنیم، یا از ایمیج های استانداری که در بخش رجیستری داکر (داکرهاب) قرارگرفته استفاده کنیم. عموماً برای سبک بودن محیط اجرای کار حتی درصورتی‌که بخواهیم فایل ایمیجی را خودمان طراحی کنیم از یک ایمیج مبتنی بر لینوکس ساده شروع کرده و هر چیزی که به آن نیاز داریم را به آن اضافه می­کنیم، ولی در اکثر مواقع برنامه‌های موردنیاز در داکرهاب وجود دارند که می­توان از آن بهره­برد.

برای مثال در شکل شکل ‏2–6 با دستور اول یک فایل ایمیج ساخته می­شود. با دستور دوم یک فایل ایمیج از بخش رجیستری (داکرهاب) به کامپیوتر میزبان منتقل می­شود و با دستور سوم، یکی از فایل­های ایمیج ساخته‌شده به‌صورت کانتینر درآمده و اجرا می­شود.

به این شکل کانتینرها ایجاد می­شوند. حال برای ارتباط کانتینرها با یکدیگر از مکانیزم های دیگری که پیش­تر در شکل ‏2–7 دیدیم استفاده می­شود. مکانیزم هایی همچون ایجاد شبکه یا اجازه دسترسی به حجم‌های فیزیکی که ارتباط میان کانتینرها را فراهم کند.



شکل ‏2–6- نمایی از معماری داکر و فرایند چرخش دستورات و اطلاعات. [2]

COPY requirements.txt ./

RUN pip install --no-cache-dir -r requirements.txt

COPY main.py ./

COPY util.py ./

CMD [ "python", "./main.py" ]

شکل ‏2–7-نمونه‌ای از یک فایل dockerfile

## داکر کامپوز[[19]](#footnote-19)

همان‌طور که در شکل ‏2–6 قابل‌مشاهده شد، برای اجرای هر مرحله می­بایست دستوری را اجرا کنیم تا مراحل پیش­رود و برنامه اجرا شود برای خودکار کردن این کار را می­توان از داکر کامپوز استفاده نمود. داکر کامپوز ابزاری است که می‌توان با استفاده از آن چند تا کانتینر را راه‌اندازی کرد و تمام موارد مربوط به راه‌اندازی آنها را در آن لحاظ کرد. داکر کامپوز با استفاده از یک YAML فایل تمام موارد مربوط به راه‌اندازی سرویس ما را دریافت و بر اساس آن سرویس که می‌تواند شامل چندین کانتینر باشد را با یک دستور راه‌اندازی می‌کند. از داکر کامپوز می‌توان در تمام محیط‌های کاری اعم از production, staging, development, testing و محیط‌های CI/CD استفاده کرد.

روابط بین کانتینرها را مشخص کرد. داکر کامپوز یک ابزار برای تعریف و اجرای برنامه‌های حاوی چند کانتینر است. این برنامه از یک فایل YAML برای پیکربندی این مجموعه استفاده می­کند. سپس با اجرای تنها یک دستور همه‌ی سرویس­های خود را فعال و آماده‌به‌کار می­کند. ساخت برنامه‌ها با داکر کامپوز از سه بخش تشکیل‌شده است.

برخی از مزایای استفاده از داکرکامپوز:

* با استفاده از آن می‌توان بر روی یک هاست چند تا محیط ایزوله با یکدیگر داشت.
* در زمان ایجاد تغییر کانتینر‌ها را مجدد ایجاد نماید.
* با استفاده از متغیرهای که تعریف می‌شود می‌توان سرویس‌ را در محیط‌های مختلف پیاده‌سازی کرد.
* اطلاعات کانتینرها‌ را حسب نیاز بر روی volumeهایی که ایجاد می‌کند قرار دهد.
* داکر کامپوز می‌تواند به خوبی با سرویس‌های CI/CD ادغام شود و تمام موارد مورد نیاز را به صورت خودکار انجام دهد.

معمولا این سناریو در راه‌اندازی سرویس‌ها انجام می‌شود. ابتدا داکرفایل برای ایجاد ایمیج‌ها آماده می‌شود. )که پیشتر در مورد ایجاد داکرفایل صبحت کردیم( سپس با توجه به نیاز موجود برای راه‌اندازی سرویس و ارتباطات و عملکرد آنها با هم یک کامپوز فایل آماده می‌گردد. سپس با استفاده از دستورات داکرکامپوز و با توجه به کامپوز فایل سرویس با تمام شرایط مد نظر را‌ه‌اندازی و در اختیار قرار می‌گیرد. می‌توان برای کل سرویس و یا هر جزئی‌ از آن نیز تست نوشت و تمام موارد را تست کرد. حال در ادامه مروری خیلی کوتاه اما مفید بر نگارش کامپوز فایل خواهیم داشت.

### دستورالعمل‌های کامپوزفایل

در این بخش در مورد دستور‌العمل‌های کامپوز فایل صحبت می‌کنیم. شایان ذکر است که نسخه‌ی ۳ کامپوزفایل بررسی خواهد شد که در زمان نگارش این داکیومنت جدیدترین نسخه‌ی آن می‌باشد. تنها دستورالعمل‌های services, networks و volumes به صورت مسیر پیش‌فرض در کامپوزفایل می‌باشند. همانطور که می‌دانید yaml به مکان دستورالعمل‌ها حساس می‌باشد از این رو در زمان ایجاد yaml file می‌بایست به موقعیت مکانی تمام دستورالعمل‌ها دقت کرد زیرا اگر در جای درستی قرار نگرفته باشند خطا داده و فایل شما اجرا نخواهد شد. در ضمن پسوند فایل می‌تواند yml. یا yaml. باشد و هر دو آن درست می‌باشد.

زمانی که یک کامپوزفایل اجرا می‌شود همانند این است که با دستورات مختلف ایمیج دریافت و یا ساخته شود و یا اینکه کانتینر با استفاده از تنظیماتی که قرار داده شده است ایجاد شود و یا اینکه شبکه با کانفیگ مشخص ایجاد شود. در واقع تمام آپشن‌ها و دستورالعمل‌هایی که در کامپوزفایل مورد استفاده قرار می‌گیرد همان دستورات داکر است که در حال استفاده می‌باشد.

1. دستورالعمل ‌build:

با استفاده از این دستورالعمل می‌توان تنظیمات مربوط به ساخت ایمیج در زمان اجرای سرویس را داد. می‌توان مسیر و اسم داکرفایل را مشخص کرد و آپشن‌های زمان build همانند arg را در آن قرار داد و یا اینکه ایمیج با چه نامی ساخته شود. دستورالعمل‌های مهم زیر مجموعه‌ی build عبارتند از:

دستورالعمل context: مسیری که داکرفایل در آنجا قرار دارد و یا آدرس ریپوی git آن را مشخص می‌کند.

دستورالعمل dockerfile: داکرفایل اگر با نام Dockerfile داخل مسیری که اشاره شد وجود ندارد و نام دیگری دارد با این آپشن مشخص می‌کند.

دستورالعمل args: دقیقا همان arg داخل داکرفایل است که می‌توان در زمان ساخت ایمیج آن را قرار داد.

دستورالعمل labels: در زمان ساخت ایمیج به آن لیبل داده می‌شود.

دستورالعمل command: دستورالعمل پیش‌فرض کانتینر را جایگذاری می‌کند.

1. دستورالعمل ‌container\_name:

اسم کانتینر را مشخص می‌کند. اگر اسم برای کانتینر انتخاب نشود ترکیبی از اسم سرویس به همراه دایرکتوری آن قرار می‌دهد.

1. دستورالعمل ‌depends\_on:

این دستورالعمل بسیار کاربردی می‌باشد و اگر بین سرویس‌های داخل کامپوز فایل اولویت زمانی در راه‌اندازی اهمیت داشته باشد با استفاده از این دستورالعمل می‌توان سرویسی را به سرویس دیگری وابسته کرد. در تصویر زیر برای اینکه سرویس web راه‌اندازی شود نیاز است تا ابتدا سرویس db و سرویس redis راه‌اندازی شود. تازمانی که هر دو سرویس راه‌اندازی نشوند سرویس web راه‌اندازی نخواهد شد. نمونه ای از اجرای این دستور عمل در شکل زیر قابل مشاهده است.



شکل فلان

1. دستورالعمل restart\_policy:

تنظیمات مربوط به restart کانتینر می‌باشد. این کانفیگ تنظیمات مربوط به restart را جایگزین می‌کند. در مثال زیر زمانی که موضوعی باعث fail کانتینر شود با ۵ ثانیه تاخیر ۳ بار تلاش می‌کند تا کانتینر را اجرا کند. نمونه از اجرای این دستورالعمل را میتوان در شکل فلان مشاهده کرد.



شکل فلان

1. دستورالعمل ‌env\_file:

اضافه کردن متغیرهای محلی با استفاده از فایل که می‌تواند شامل یک متغیر و یا لیستی از آنها باشد. به مثال‌های زیر توجه کنید.

env\_file: .env

env\_file:

- ./common.env

- ./apps/web.env

1. دستورالعمل ‌environment:

از این دستورالعمل برای زمانی که بخواهیم یک یا چند متغیر را در کانتینر مقداردهی کنیم استفاده می‌کنیم. تفاوت در این است که تمام متغیرها داخل خود کامپوزفایل تعریف خواهد شد و فایل دیگه‌ای لود نخواهد شد. به هر دو صورت زیر از آن استفاده می‌شود.

environment:

- RACK\_ENV=development

- SHOW=true

- SESSION\_SECRET

1. دستورالعمل ‌expose:

برای expose کردن پورت داخل کانتیر استفاده می‌شود. این به معنای در دسترس قرار دادن آن پورت در داکرهاست نمی‌باشد. به آن عمل publish می‌گویند که دستورالعمل مخصوص خودش را دارد. معمولا از expose برای ارتباط داخلی بین کانتینرها و زمانی که آنها بین هم link می‌شوند استفاده می‌شود.

1. دستورالعمل ‌image:

ایمیج مورد استفاده را مشخص می‌کند و کانتینر از روی این ایمیج آماده می‌شود. البته می‌توان با استفاده از دستورالعمل build ایمیج مورد استفاده را از روی داکرفایل نیز ایجاد کرد.

1. دستورالعمل ports:

که با استفاده از آن می‌توان پورت‌ها رو از داخل کانتیر به هاست publish کرد. به صورت اختصار به صورت HOST:CONTAINER استفاده می‌شود. برای مثال پورت ۸۰ کانتیر به پورت ۸۰۸۰ هاست مپ می‌شود. می‌توان تنها پورت کانتیر را قرار داد که آنگاه سرویس داکر آن را بر روی یکی از رندم پورت‌های هاست مپ می‌کند. و می‌توان به صورت اختصاصی مشخص کرد که این پورت بر روی کدام پورت و ip سروس هاست bind شود و یا اینکه در چه پروتکلی قرار داشته باشد. در مثال‌های زیر تمام موارد آورده شده است.

ports:

- "3000"

- "3000-3005"

- "8000:8000"

- "9090-9091:8080-8081"

- "49100:22"

- "127.0.0.1:8001:8001"

- "127.0.0.1:5000-5010:5000-5010"

1. دستورالعمل ‌volumes:

به دو روش یکی قرار دادن بر روی مسیری از دایرکتوری هاست و دیگری ایجاد والیوم با استفاده از درایورهای مختلف می‌توان داده‌های کانتینر را بر روی هاست ذخیره کرد. در استفاده از والیوم نیز می‌توان به صورت مختصر یعنی HOST:CONTAINER استفاده کرد یا به صورت طولانی‌تر تمام هر قسمت را توضیح داد. در مثال زیر چند نمونه از استفاده‌ی مختصر آن آورده شده است.

volumes:

# Just specify a path and let the Engine create a volume

- /var/lib/mysql

# Specify an absolute path mapping

- /opt/data:/var/lib/mysql

# Path on the host, relative to the Compose file

- ./cache:/tmp/cache

# User-relative path

- ~/configs:/etc/configs/:ro

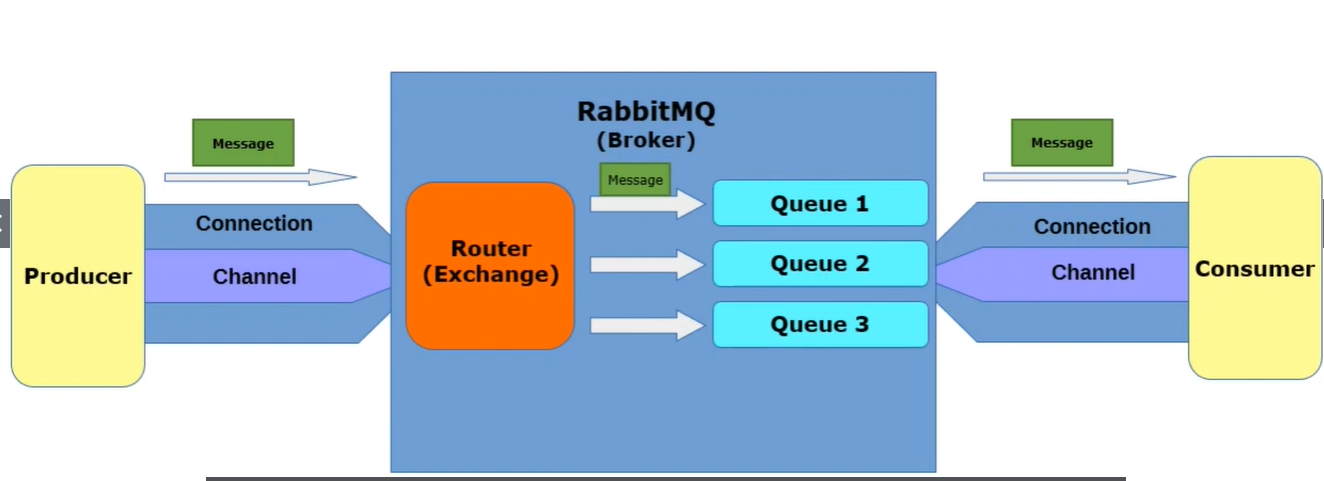
# Named volume

- datavolume:/var/lib/mysql

## سرویس RabbitMQ

RabbitMq یک کارگزار پیام است که می­توان با کمک آن سیستم‌های بزرگی را یکپارچه­سازی کرد. به این شکل که پیام را ابتدا از تولیدکنندگان اطلاعات دریافت می­کند و به مصرف‌کنندگان اطلاعات منتقل می­کند. این نرم‌افزار متن‌باز و با زبان برنامه‌نویسی Erlang نوشته‌شده است و از پروتکل­های بسیاری ازجمله AMQP، STOMP، MQTT، HTTP و WebSocket پشتیبانی می­کند. به همین جهت از این نرم‌افزار می­توان به‌عنوان پلی میان سخت‌افزار و محیط ابری یادکرد. این نرم‌افزار بر روی ویندوز، لینوکس و مک قابل‌استفاده است و همچنین به‌صورت کانتینر داکر در رجیستری داکر (داکرهاب) قرارگرفته است که استفاده از آن را بسیار ساده می­کند.

در شکل ‏4–9 معماری کلی این نرم‌افزار قابل‌مشاهده است، تولیدکنندگان اطلاعات[[20]](#footnote-20) و مصرف‌کنندگان اطلاعات[[21]](#footnote-21) در دو سمت این معماری و این نرم‌افزار در مرکز شکل قرارگرفته‌اند. آن‌ها به این نرم‌افزار متصل شده و به تبادل اطلاعات می­پردازند. در این معماری اجزای دیگری ازجمله صف[[22]](#footnote-22)، مرکز توزیع[[23]](#footnote-23) و کانال‌ها[[24]](#footnote-24) قابل‌مشاهده هستند.

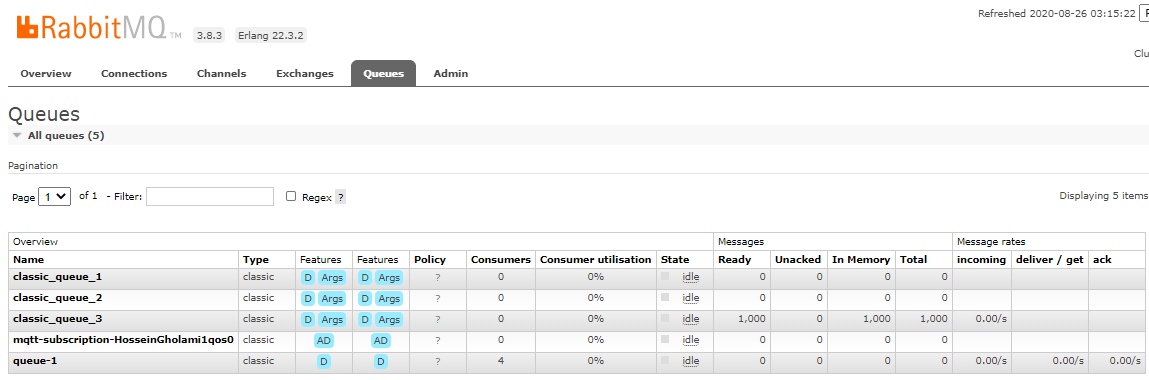


شکل ‏2–8-معماری کلی نرم‌افزار RabbitMQ [3]

که در ادامه با بررسی یک مثال به شرح وظایف هرکدام از این اجزا می­پردازیم. و در ادامه هر یک از بخش ها را به طور کامل شرح می­دهیم.

برنامه هایی را در نظر بگیرید که در هر ثانیه میخواهند یک پیام را به برنامه ای دیگر در شبکه منتقل کند، که ترتیب انتقال و عدم تاخیر و عدم حذف شدن اطلاعات در انتقال اطلاعات برای آن الویت دارد. پس واضح است که پیام ها باید در یک نقطه جمع شده و سپس به صف مربوطه خود منتقل شوند و برنامه دیگر اطلاعات را از انتها صف بردارد. با یک مثال ساده تر عملکرد کلی سیستم را شرح میدهم. کلینیک یک بیمارستان را در نظر بگیرید که تعدادی پزشک از رشته های متفاوت دارد، و اشخاص قصد دارند به این بیمارستان مراجعه کرده و هر کدام پیش پزشک مد نظر خود بروند. همه افراد ابتدا به پذیرش کلینیک مراجعه کرده و پذیرش پزشک مخصوص خود را میگیرند. سپس هر کدام به صف مربوطه پزشک خود میروند و هرگاه پزشک بخواهد می­تواند افرادی که در صف هستند را ویزیت کند. در این مثال تولید کننده اطلاعات، بیمارانی هستند که بیمارستان مراجعه میکنند. کانال ورودی ، بیمارستان است. کانال خروجی اتاق دکتر است. مرکز توزیع همان پذیرش بیمارستان است که افراد را به صف انتظار دکترها هدایت می­کند. و اتاق انتظار پزشک صف ها هستند.

RabbitMQ یک پنل داشبورد تحت وب برای بررسی شرایط صف‌ها، تبادل داده‌ها و کانال‌ها و ارتباطات می­دهد که در آن می­توان اطلاعات مفیدی هنگام به‌کارگیری در عمل به دست آورد. در شکل ‏4–10 صفحه مربوط به‌صف­های را می­توانید مشاهده کنید، برای مثال در این تصویر 4 مصرف‌کننده اطلاعات بر روی صف queue-1 قرارگرفته شده تا در صورت انتقال اطلاعات به این صف، پیام‌ها دریافت شوند. در صف classic\_queue\_3 هزار پیام در حافظه قرارگرفته و هیچ مصرف‌کننده اطلاعاتی به این صف متصل نیست.



شکل ‏2–9- داشبورد مدیریتی نرم‌افزار RabbitMQ

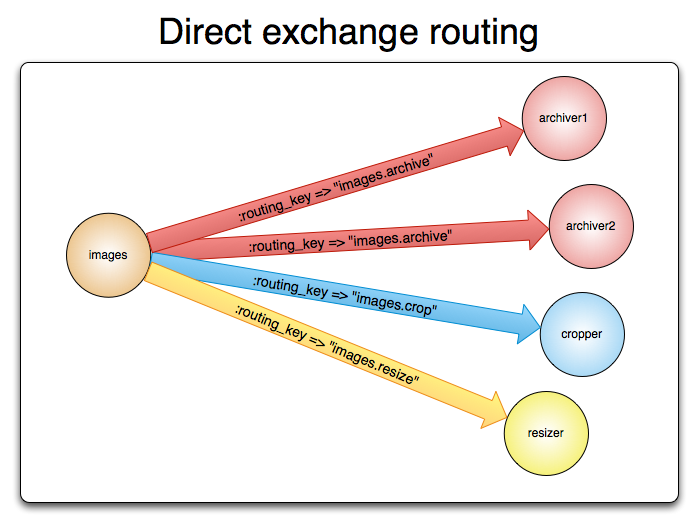
با توجه به توضیحات فوق و بررسی وظایف کلی هر بخش، به بررسی دقیق­تر هر یک از اجزا میرویم.

### مراکز توزیع پیام (Exchange ها)

مراکز توزیع پیام موجودیت هایی هستند که پیام ها به آنها ارسال می شود. مراکز توزیع پیام یک پیام را دریافت می کنند و آن را به صف ها یا هیچ هدایت می کنند. الگوریتم مسیریابی مورد استفاده بسته به نوع مراکز توزیع پیام و قوانینی به نام اتصالات[[25]](#footnote-25) استفاده می­شود. مراکز توزیع پیام از 4 نوع کلی تشکیل شده اند، مرکز توزیع مستقیم[[26]](#footnote-26)، مرکز توزیع خروج به همه[[27]](#footnote-27)، مرکز توزیع موضوعی[[28]](#footnote-28) و مرکز توزیع سری[[29]](#footnote-29) علاوه بر موارد فوق برای تعریف هر مرکز توزیع باید یک نام منحصر به فرد، پایداری[[30]](#footnote-30)(این که بعد از ریست شدن سرویس آیا این مرکز توضیع وجود داشته باشد یا حذف شود.) حذف خودکار(اگر صف های متصل شده به این سرویس اتصالشان قطع شود، مرکز توزیع حذف شود) و سایر پارامترهایی که میتوان برای کنترل بهتر داد(این موارد اختیاری هستند و شامل تعیین پارامترهایی برای کنترل آن در میزبان هایی دیگر تعریف میشوند.) در ادامه به انواع متفاوت این مراکز توزیع پرداخته میشود.

**مرکز توزیع مستقیم**

مرکز توزیع مستقیم پیام ها را بر اساس کلیدمسیریابی که در آن تعیین شده به صف های منتخب متصل به آن ارسال می­کند. این مراکز توزیع برای ارسال های تک به تک[[31]](#footnote-31) مناسب هستند.(اگر چه میتوان از آنها در ارسال های سراسری[[32]](#footnote-32) نیز استفاده نمود) در شکل فلان میتوان نمونه ای از استفاده از این مرکز توزیع پیام را مشاهده کرد.



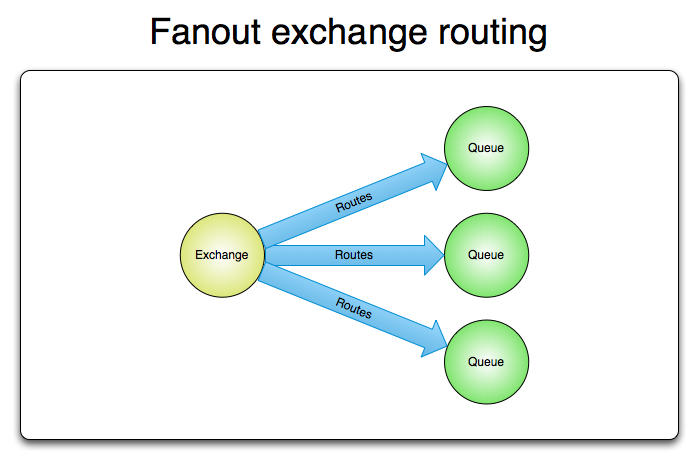
شکل ‏2–10

در این مدل صف های زیادی به مرکز توزیع پیام با کلید مسیریابی منحصر به فرد خودشان متصل شده اند و وقتی یک پیام جدید می­آید بر اساس کلیدمسیریابی که در آن قرار گرفته است به صف های مورد نظر خود ارسال می­شود. برای مثال در تصویر فوق مرکز توزیع image وجود دارد و صف های archive1 و archive2 و ... به این مرکز اتصال با کلید های مسیریابی image.archive1 و image.archive2 و... متصل شده اند. حال اگر یک پیام جدید بیاید، که کلید مسیریابی image.archive1 داشته باشد تنها به صف archive1 منتقل خواهد شد. این در حالی است که اگر بسته ای با کلید مسیریابی salam به این مرکز توزیع ارسال شود به هیچ یک از صف های فوق نخواهد رفت و پیام حذف خواهد شد.

مراکز توزیع مستقیم اغلب برای توزیع وظایف بین چندین برنامه (نمونه هایی از یک برنامه کاربردی) به صورت چرخشی استفاده می شود. هنگام انجام این کار، درک این نکته مهم است که پیام ها بین مصرف کنندگان و نه بین صف ها متعادل می شوند.

**مراکز توزیع خروج به همه**

در مراکز توزیع خروج به همه پیام‌ها را به تمام صف‌هایی که به آن متصل هستند، مستقل از کلید مسیریابی هدایت ‌خواهند شد و اگر N صف به یک مرکز توزیع خروج به همه متصل شده باشد شده باشد، زمانی که یک پیام جدید برای آن مبادله منتشر می شود، یک کپی از آن پیام به تمام N صف تحویل داده می شود. تبادلات Fanout برای مسیریابی پخش پیام ها ایده آل هستند، چرا که پیام را به تمامی صف ها ارسال میکنند. برای مثال; بازی‌های آنلاین چند نفره (MMO) می‌توانند از این ویژگی برای به‌روزرسانی‌های تابلوی امتیازات خود استفاده کنند. یا سایت‌های خبری ورزشی می‌توانند از مراکز توزیع خروج به همه برای کاربران خود در زمان واقعی استفاده کنند. یا سیستم های توزیع شده می توانند برای روز رسانی های وضعیت های مختلف و پیکربندی از این ویژگی استفاده کنند. چت های گروهی می توانند پیام ها را بین کاربران خود با استفاده از این مرکز توزیع، توزیع کنند. در تصویر زیر نمونه از این این مدل را میتوان مشاهده کرد.



شکل ‏2–11

**مراکز توزیع موضوعی**

مراکز توزیع موضوعی، یک پیام را به یک یا چند صف بر اساس تطابق بین یک کلید مسیریابی پیام و نوع اتصال صف بین یک یا چند صف مبادله می‌کند. نوع تبادل موضوع اغلب برای اجرای انواع الگوهای انتشار/اشتراک[[33]](#footnote-33) استفاده می‌شود. تبادل موضوع معمولا برای مسیریابی چندپخشی پیام ها استفاده می شود.

مبادلات موضوعی دارای مجموعه بسیار گسترده ای از موارد استفاده هستند. هر زمان که مشکلی شامل چندین مصرف کننده/برنامه کاربردی می شود که به طور انتخابی نوع پیام هایی را که می خواهند دریافت کنند انتخاب می کنند، استفاده از مبادلات موضوعی باید در نظر گرفته شود. برخی از کاربردهای آن شامل;

توزیع داده های مربوط به موقعیت جغرافیایی خاص، به عنوان مثال، نقاط فروش. پردازش کار پس‌زمینه توسط چندین کارگر انجام ‌شود که هر کدام قادر به انجام مجموعه‌ای از وظایف هستند. به روز رسانی قیمت سهام. به‌روزرسانی‌های خبری که شامل دسته‌بندی یا برچسب‌گذاری است (مثلاً فقط برای یک ورزش یا تیم خاص). ساختارها یا بسته‌بندی‌های نرم‌افزاری خاص معماری/سیستم‌عامل که در آن هر سازنده می‌تواند تنها یک معماری یا سیستم‌عامل را مدیریت کند.

**مراکز توزیع سری**

مراکز توزیع سری برای مسیریابی در چندین ویژگی طراحی شده است که به راحتی به عنوان سر پیام نسبت به یک کلید مسیریابی بیان می شوند. مبادلات سری ویژگی کلید مسیریابی را نادیده می گیرند. در عوض، ویژگی های مورد استفاده برای مسیریابی از ویژگی headers گرفته شده است. اگر مقدار هدر با مقدار مشخص شده در هنگام اتصال برابر باشد، پیامی مطابق در نظر گرفته می شود و به صف مربوطه ارسال خواهد شد.

می توان یک صف را با استفاده از بیش از یک سر برای تطبیق به یک تبادل سری متصل کرد. در این مورد، کارگزار به یک اطلاعات بیشتر از توسعه‌دهنده برنامه نیاز دارد، یعنی آیا باید پیام‌هایی را با هر یک از هدرها یا همه آنها در نظر بگیرد؟ این همان چیزی است که آرگومان الزام آور "x-match" برای آن است. زمانی که آرگومان "x-match" روی "any" تنظیم شود، فقط یک مقدار هدر منطبق کافی است. از طرف دیگر، تنظیم "x-match" بر روی "all" الزام می کند که همه مقادیر باید مطابقت داشته باشند.

این مراکز توزیع همانند مراکز توزیع مستقیم هستند و تنها تفاوت آن در نحوه ارسال بسته ها است که نوع تصمیم گیری به جای کلیدمسیریابی به سر بسته میسپارد.

### صف ها در RabbitMQ

صف‌ها در ربیت بسیار شبیه به صف‌ها در سایر سیستم‌های صف پیام و وظیفه[[34]](#footnote-34) هستند: آنها پیام‌هایی را ذخیره می‌کنند که توسط برنامه‌ها مصرف کننده اطلاعات، مصرف می‌شوند. صف ها برخی از ویژگی ها را با مراکز توزیع به اشتراک می گذارند، اما برخی ویژگی های اضافی نیز دارند که شامل موارد زیر هستند: نام منحصر به فرد، پایداری(اگر سرویس ریست شد، صف ها حذف نشوند.) انحصاری (فقط توسط یک مصرف کننده اطلاعات مصرف شوند) حذف خودکار (صفی که حداقل یک مصرف کننده اطلاعات داشته است، با لغو اشتراک آخرین مصرف کننده حذف می شود) آرگومان‌ها (اختیاری؛ استفاده شده توسط افزونه‌ها و ویژگی‌های خاص کارگزار مانند زمان نگهداری هر پیام، محدودیت طول صف و غیره)

قبل از اینکه بتوان از یک صف استفاده کرد، باید آن را تعریف کرد. تعریف یک صف باعث ایجاد آن می شود اگر قبلا وجود نداشته باشد. اگر صف از قبل وجود داشته باشد و نام آن مانند آنچه در لیست صف های موجود تعریف شده باشد، تعریف اثری نخواهد داشت و با خطای 406 مواجه خواهیم شد.

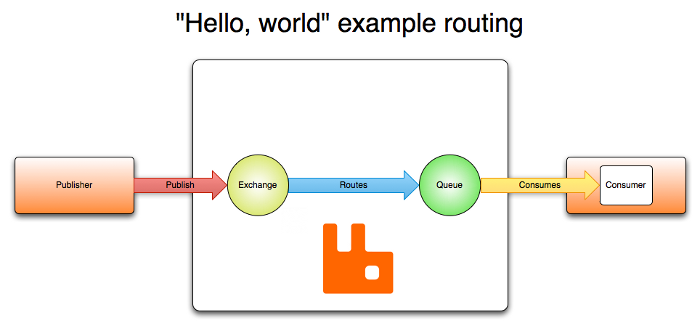
برنامه‌ها ممکن است نام‌های صف را انتخاب کنند یا از کارگزار بخواهند نامی برای آنها ایجاد کند. نام صف ممکن است تا 255 بایت نویسه UTF-8 باشد. ربیت می تواند یک نام صف منحصر به فرد از طرف برنامه ایجاد کند. برای استفاده از این ویژگی، یک رشته خالی را به عنوان آرگومان نام صف ارسال کنید.

صف ها را می توان به عنوان پایدار یا گذرا تعریف کرد. ربیت یک صف پایدار را روی دیسک ذخیره میکند این در حالی که اگر یک صف گذرا تعریف شود این اطلاعات در صورت امکان در حافظه ذخیره خواهند شد. که منجر به افزایش سرعت انتقال می­شود.

همانطور که پیشتر گفته شد، اتصالات، قوانینی هستند که مراکز توزیع بر اساس آن تصمیم میگیرند که یک بسته به کدام صف ارسال شود. برای دستور دادن به مرکز توزیع E برای مسیریابی پیام‌ها به صف Q، Q باید به E متصل شود. اتصالات ممکن است یک ویژگی کلید مسیریابی اختیاری داشته باشد که توسط برخی از انواع تبادل استفاده می‌شود. هدف کلید مسیریابی انتخاب پیام‌های خاصی است که به یک مرکز توزیع منتشر میکند تا به صف محدود شده مورد نظر خود هدایت شود. به عبارت دیگر، کلید مسیریابی مانند یک فیلتر عمل می کند. مثال زیر را در نظر بگیرید؛ فرض کنید که قصد سفر با هواپیما دارید و مقصد شما اصفهان است. مرکز توزیع مانند فرودگاه است. که میتوانید به فرودگاه بروید و اگر بلیت درستی نداشته باشید به مقصد نخواهید رسید ولی اگر بلیت درست(کلیدمسیریابی) صحیح داشته باشید به هواپیما منتثل خواهید شد(به صف منتقل خواهید شد) و با پرواز هواپیما به مقصد خواهید رسید(برنامه ای محتوا را از صف بردارد).

### تولید کننده و مصرف کننده اطلاعات

دیدیم که ربیت کنترل میان تولید کننده اطلاعات و مصرف کننده اطلاعات را فراهم میکند. مصرف کننده اطلاعات به صف متصل شده و در صورتی که پیامی داخل آن ها شود، آن پیام ها را دریافت می­کند. در عین حال تولید کننده اطلاعات پیام های خود را آماده میکند و با کلید مسیریابی مشخص به مراکز توزیع ارسال می­کند. این دو برنامه به طور کامل از یکدیگر مستقل هستند و میتوان هر کدام از آن ها را حتی با یک زبان هایی برنامه نویسی متفاوت نوشت که هر کدام در کامپیوتر هایی که در شبکه قرار گرفته اند به هم متصل شوند. عموما پروتکل ارتباطی با rabbitmq، AMQP است



شکل ‏2–12

AMQP (پروتکل صف پیشرفته پیام) یک پروتکل پیام رسانی است که برنامه های کاربردی کلاینت منطبق را قادر می سازد تا با کارگزاران میان افزار پیام رسان منطبق ارتباط برقرار کنند. AMQP یک پروتکل قابل برنامه ریزی است به این معنا که موجودیت های AMQP و طرح های مسیریابی در درجه اول توسط خود برنامه ها تعریف می شوند، نه توسط برنامه های ارسال کننده و دریافت کننده اطلاعات. بر این اساس، برای عملیاتی که صف‌ها و مبادلات را اعلام می‌کند، پیوندهای بین آنها را تعریف می شود، مراکز توزیع ور صف‌ها پیش بینی می‌شود. این به توسعه دهندگان برنامه آزادی زیادی می دهد.

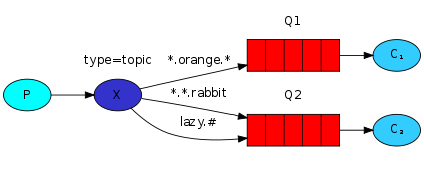
### نکاتی تکمیلی در مورد برخی ویژگی ها

هنگامی‌که هر صف تعریف می­شود، در خود بخشی با عنوان «کلید­ مسیریابی[[35]](#footnote-35)» دارد و باید به یک مرکز توزیع پیام متصل شود. صف‌ها با کمک کلید انتقال بسته‌هایی که وارد، یک مرکز انتقال می­شوند را می­توانند در خود جای دهند. برای مثال stock.usd.ny و quick.orange.rabit می­توانند هر دو نام تاپیک برای بسته‌ها باشند و صف‌هایی با کلید مسیریابی فوق می­توانند، این پیام‌ها را دریافت کنند. دراین‌بین برای RabbitMQ از سه کاراکتر ‘.’ و ‘#’ و’\*’ برای مدیریت کردن بهتر بسته‌ها هنگام تقسیم بسته‌ها میان صف‌ها استفاده می­کند.

کاراکتر ‘.’ به معنی ایجاد ساختار است و هنگامی‌که در کلید مسیریابی استفاده شود باید دقیقا تاپیک پیام و کلیدمسیریابی یکسان باشد. کاراکتر ‘\*’ به این معنی است که در بخش‌هایی که این کاراکتر قرارگرفته می­تواند تاپیک بسته با کلیدمسیریابی متفاوت باشد و کاراکتر ‘#’ به این معناست، درصورتی‌که تا قبل از این کاراکتر یکسان بود، بسته به صف منتقل شود. مثال زیر را در نظر بگیرید:

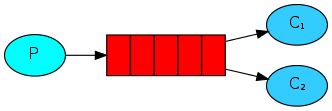
بسته‌ای با تاپیک quick.orange.rabbit به یک مرکز انتقال می­رسد و چهار صف با کلیدهای مسیریابی زیر به این مرکز انتقال متصل هستند.

حالت اول کلیدمسیریابی quick.orange است، این پیام دریافت نمی­شود، چراکه تعداد بخش‌های ایجادشده با ‘.’ یکسان نیست. حالت دوم کلیدمسیریابی \*.orange.\* است، این پیام دریافت می­شود، چراکه تعداد بخش‌های ایجادشده با نقطه یکسان و از دو بخش اول و آخر بسته صرف‌نظر می­شود. حالت سوم، کلیدمسیریابی quick.# است، این پیام دریافت می­شود چراکه تاپیک بسته با هر آنچه که قبل از ‘#’ است، یکسان می­باشد. این مکانیزم می­تواند برای دسته‌بندی پیام‌های ورودی بسیار مؤثر باشد. در شکل ‏4–12 می­توان مثال دیگری را مشاهده نمود.



شکل ‏2–13-نحوه استفاده از کلیدمسیریابی برای صف‌ها

در این ساختار همان‌طور که پیش­تر بررسی شد، می­توانیم از چندین مصرفت کننده اطلاعات به‌طور هم‌زمان استفاده کنیم تا بتوانیم حجم بسته‌های داخل صف را مدیریت کنیم. در این سیستم یک پارامتری تحت عنوان پیش­دستی[[36]](#footnote-36) تعریف می­شود، به این معنا که تعداد بسته‌های درخواستی از صف را کنترل می­کند که بسته‌ها به‌صورت چندتایی ارسال شوند. در این حالت، بسته‌ها به حالت گذرا تغییر وضعیت یافته تا هنگامی‌که سرویس­دهی این بسته‌ها تمام شد، یک تعداد بسته جدید تقاضا شود. این امر موجب می­شود تعداد پیام‌هایی که برای ارسال و دریافت بین سرور صف و سرویس‌دهنده منتقل می­شود، کاهش یابد و درنتیجه بازدهی بالاتر رود. برای مثال به شکل ‏4–13 توجه کنید.



**شکل ‏2–14-اتصال دو مصرف‌کننده‌ی اطلاعات به یک صف**

فرض کنید در این شکل دو مصرف‌کننده‌ی اطلاعات، به این صف متصل شده‌اند و در صف 50 بسته وجود دارد. C1 با پیش­دستی 5 به این صف متصل شده و C2 که کامپیوتری با امکانات سخت‌افزاری قوی­تری است، با پیش­دستی 10 به این صف متصل می­شود. حال در لحظه صفر C1 درخواست 5 بسته و C2 درخواست 10 بسته می­کند. در این حالت همچنان 50 بسته در صف وجود دارد، با این تفاوت که 15 بسته به حالت گذرا درآمده است. بعد از پردازش توسط C1 وC2 اعلان پایان کار داده و تقاضای بسته جدید کرده به این شکل بسته‌ها از صف حذف‌شده و بسته‌های جدید برای آن‌ها ارسال می­شود.

برای استفاده از RabbitMQ در پروژه میتوان از زبان­های برنامه نویسی متعددی استفاده نمود در این پروژه از زبان پایتون و کتابخانه Pika استفاده شد که توسط خود rabbitMq معرفی و پیشنهاد شده است. که در فصل های بعدی این کتابخانه را بهتر بررسی خواهیم نمود.

### راه­اندازی RabbitMQ با داکر

از پیش­تر در بخش مروری بر داکر به یاد داریم که می­توان ایمیج های استاندارد را از داکرهاب، بارگذاری و مورداستفاده قرارداد. RabbitMQ خود یکی از ایمیج های استاندارد است که به‌سادگی از آن استفاده می­شود. برای این کار می­توانیم در داخل داکر فایل از ایمیج استاندارد شروع کرده و پلاگین­هایی که قصد فعال‌سازی آن را داریم، با اجرای دستور مربوطه به‌کارگیریم تا ایمیج جدیدی با ویژگی‌هایی که مدنظر داریم ساخته شود. در شکل ‏4–14 محتوای داکر فایل را می‌توان مشاهده کرد.

FROM rabbitmq:3-management

COPY ./Conf/definitions.json /etc/rabbitmq/

COPY ./Conf/enabled\_plugins /etc/rabbitmq/

COPY ./Conf/rabbitmq.conf /etc/rabbitmq/

RUN rabbitmq-plugins enable rabbitmq\_shovel

RUN rabbitmq-plugins enable rabbitmq\_shovel\_management

RUN rabbitmq-plugins enable rabbitmq\_federation

RUN rabbitmq-plugins enable rabbitmq\_federation\_management

EXPOSE 5672

EXPOSE 15672

شکل ‏2–15-محتوای داکر فایل برای راه‌اندازی RabbitMQ

حال با به­کارگیری داکر کامپوز تنها کافی است دستور دهیم که ایمیج فوق ساخته شود و پورت مخصوص به ارتباط با کانتینر و کامپیوتر میزبان برقرار شود. همچنین می­توان فایل تنظیمات را تغییر داد و آن را در داخل کانتینر قرار داد. در شکل ‏4–15 می­توان دستوراتی که در فایل docker-compose.yaml برای بالا آمدن کانتینر RabbitMQ نوشته‌شده را مشاهده کرد.

services:

  queue:

    build: rbmq/.

    container\_name: rabbitmq

    ports:

      - 5672:5672

      - 15672:15672

    networks:

      - app-tier

شکل ‏2–16- بخش مرتبط با RabbitMQ در داکر کامپوز

## سرویس Redis

Redis یک ذخیره کننده ساختار اطلاعات، داخل حافظه‌ای متن‌باز است که می­تواند به‌عنوان پایگاه­داده، حافظه نهان و ابزاری برای یکپارچه‌سازی عمل کند. Redis از ساختارهای اطلاعات متفاوتی ازجمله؛ رشته­ها، هش­ها، لیست­ها، مجموعه­ها، مجموعه­های مرتب‌شده برای کاربردهای متفاوت و جریان­های اطلاعات (سری زمانی‌ها) پشتیبانی می­کند و توانایی اجرا بر روی خوشه­ای از کامپیوترها را دارا است.

در این نرم‌افزار بعضی از امکانات محاسباتی ازجمله اضافه کردن به رشته‌ها، تغییر در اعداد، تغییر در لیست‌ها، محاسبه اشتراکات دو مجموعه، بررسی المان‌های منحصربه‌فرد و ...را انجام داد. Redis به زبان **ANSI C** نوشته‌شده و بر روی همه‌ی سیستم‌عامل‌ها بدون هیچ‌گونه پیش‌نیاز قابل‌اجرا است. اگرچه این برنامه بر روی همه‌ی سیستم‌عامل‌ها توانایی اجرا دارد ولی خود برنامه، استفاده از سری سیستم‌عامل‌های Linux را پیشنهاد می­دهد.

درست است که Redis اطلاعات را در حافظه ذخیره می‌کند اما به این معنی نیست که پس از خاموش شدن و یا هر اتفاقی که باعث خالی شدن حافظه شود، داده‌های ما پاک می‌شوند. بلکه Redis برای نگه‌داری دائمی داده‌ها آن‌ها را با توجه به تنظیماتی که برای آن مشخص کرده‌ایم به دیسک اصلی سیستم منتقل کرده و بعد از پاک شدن حافظه مجدد می‌تواند آن‌ها را منتقل کند و کار را از سر بگیرد. این ویژگی باعث شده اصطلاحاً به آن **on-disk persistence** بگویند.

زمانی از ذخیره‌سازی موقت[[37]](#footnote-37) استفاده می‌شود که قصد داشته باشیم دسترسی به هارددیسک کمتر انجام شود. به‌عبارت‌دیگر در ذخیره­سازی موقت، اطلاعات در حافظه موقت ذخیره می‌شود که این فرآیند سرعت دسترسی به اطلاعات و بارگذاری آن‌ها را افزایش می‌دهد. از این طریق در کنار صرفه‌جویی در زمان و افزایش سرعت، دسترسی کمتری به منابع موردنیاز انجام می‌شود. این امر نیز به بهینه‌سازی بیشتر کمک می‌کند. به این نکته نیز باید اشاره کرد که در Redis اطلاعات در حافظه ذخیره می‌شوند، این امر باعث می‌شود دسترسی به آن‌ها با سرعت بسیار بیشتری انجام شود؛ اما این سکه روی دیگری نیز دارد و امکان ذخیره‌سازی دائمی اطلاعات را در Redis وجود نخواهد داشت. در این پروژه ازآنجا که پس از تجمیع اطلاعات نیازمند ارتباط میان برنامه ها را داریم از این سرویس استفاده شد.

### انواع ذخیره سازی اطلاعات در Redis

در Redis دو عنصر، کلید و مقدار[[38]](#footnote-38) داریم. عنصر مقدار می­تواند انواع مختلفی داشته باشد. اجازه دهید به برخی این مقادیر و کاربردهای آن‌ها نگاهی بیندازیم.

**رشته‌ها:** اگر مقدار از نوع رشته بود، می­توان عملیات درج، بهنگام سازی، حذف، دریافت را از آن کلید انجام داد. مثلاً با دستور زیر می­توان یک کلید ساخت و یک‌رشته در آن درج کرد:

|  |  |
| --- | --- |
| یک کلید به اسم user\_1 ساخته‌شده و  book1 به‌عنوان مقدار برای این کلید در نظر گرفته می­شود. | SET user\_1 book1 |
| مقدار مورد (book1) نظر برای کلید user\_1 را برگردانده می­شود. | GET user\_1 |

**لیست:** با استفاده از لیست می­توان یک آرایه دلخواه داشت که بتوان در این آرایه، عنصری را اضافه یا کم کرد و عملیات مختلف دیگر را انجام داد.

**دسته‌ها**[[39]](#footnote-39)**:** اگر مقدار موجود از نوع دسته‌ها باشد درواقع یک لیست وجود دارد که هیچ‌کدام از عناصر آن تکراری نیستند.

**دسته‌های منظم شده**[[40]](#footnote-40)**:** همانند مقدارهایی از نوع دسته است با این تفاوت که هر عنصر از مجموعه دارای وزن است و این وزن می­تواند به‌صورت مرتب نگهداری شود. (مثلاً از وزن کم به زیاد)

**هش­ها[[41]](#footnote-41):** اگر با انواع JSON آشنایی داشته باشید درک مقادیر هش‌ها ساده­تر است. این مقادیر می­توانند اشیایی مانند JSON را در خود جای دهند. بدین شکل که ابتدا کلیدی برای ذخیره‌سازی اطلاعات دریافت کرده و سپس به ذخیره‌سازی اطلاعات به‌صورت زوج مرتب‌هایی از کلید و مقدار می­پردازد.

علاوه بر موارد فوق، Redis می­تواند مقادیر دیگری را نیز ذخیره و بازیابی کند. یکی از انواع آن‌ها، HyperLogLog است. فرض کنید می­خواهید تعداد تکرار یک عنصر خاص از یک لیست را به دست آورید. اگر این لیست بسیار بزرگ باشد، این کار به‌راحتی انجام نمی­پذیرد. مقادیر HyperLogLog می‌تواند با دقت بسیار بالا (اما نه ۱۰۰ درصد) تعداد تکرار یک عنصر خاص را حدس بزند. این کار با استفاده از الگوریتم‌های خاصی امکان‌پذیر است. انواع دیگری مانند ذخیره‌سازی عناصری از جنس موقعیت‌های مکانی و یا Bitmap های نیز در Redis وجود دارند که کاربردهای خاص خود را دارند.

### مکانیزم لیست کنترل دسترسی

مکانیزم لیست کنترل دسترسی یا ACL[[42]](#footnote-42) قابلیتی است که به اتصالات خاصی اجازه می دهد از نظر دستورات قابل اجرا و کلیدهای قابل دسترسی محدود شوند. روش کار به این صورت است که پس از اتصال، یک کاربر باید با ارائه یک نام کاربری و یک رمز عبور معتبر احراز هویت کند. اگر مرحله احراز هویت موفقیت آمیز بود، اتصال با یک کاربری معین و محدودیت هایی که کاربر دارد مرتبط می شود. Redis را می توان به گونه ای پیکربندی کرد که اتصالات جدید که قبلاً با یک کاربر "پیش فرض" احراز هویت شده اند دسترسی پیدا کنند (این پیکربندی پیش فرض است)، بنابراین پیکربندی کاربر پیش فرض، به عنوان یک اثر جانبی، توانایی ارائه تنها زیرمجموعه خاصی از عملکردها را برای اتصالات دارد.

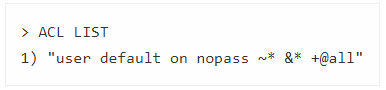
قبل از استفاده از ACL، ممکن است بخواهید از خود بپرسید هدفی که می خواهید با اجرای این لایه حفاظتی به دست آورید چیست. به طور معمول دو هدف اصلی وجود دارد که ACL به خوبی انجام می دهد:

* شما می‌خواهید با محدود کردن دسترسی به دستورات و کلیدها، امنیت را بهبود ببخشید تا کلاینت‌های غیرقابل اعتماد دسترسی نداشته باشند و کلاینت‌های مورد اعتماد فقط حداقل سطح دسترسی را به پایگاه داده برای انجام کار مورد نیاز داشته باشند. به عنوان مثال، برخی از مشتریان خاص ممکن است فقط بتوانند دستورات فقط خواندنی را اجرا کنند.
* شما می‌خواهید ایمنی عملیاتی را بهبود ببخشید، به طوری که به دلیل خطاهای نرم‌افزاری یا اشتباهات دستی، دسترسی فرآیندها یا انسان‌ها به Redis مجاز نباشد به داده‌ها یا پیکربندی آسیب برساند. به عنوان مثال، هیچ دلیلی وجود ندارد که شخصی کاربری عادی دارد بتواند دستور FLUSHALL، که کل دیتابیس را پاک می­کند، را فراخوانی کند.

یکی دیگر از کاربردهای معمولی ACL مربوط به نمونه های Redis مدیریت شده است. Redis اغلب به عنوان یک سرویس مدیریت شده توسط تیم های داخلی شرکت ارائه می شود که زیرساخت های Redis را برای سایر مشتریان داخلی که دارند مدیریت می کنند، یا در یک راه اندازی نرم افزار به عنوان یک سرویس توسط ارائه دهندگان ابر ارائه می شود. در هر دو چنین تنظیماتی، ما می خواهیم مطمئن باشیم که دستورات پیکربندی برای مشتریان مستثنی شده است.

ACL ها با استفاده از یک DSL[[43]](#footnote-43) (زبان خاص دامنه) تعریف می شوند که توصیف می کند یک کاربر مشخص می تواند چه کاری انجام دهد یا نه. چنین قوانینی همیشه از اولین تا آخرین، از چپ به راست اجرا می شوند، زیرا گاهی اوقات ترتیب قوانین برای درک اینکه کاربر واقعاً قادر به انجام چه کاری است مهم است. به طور پیش فرض یک کاربر تعریف شده است که به آن پیش فرض می گویند. ما می‌توانیم از دستور ACL LIST برای بررسی ACLهای فعال فعلی استفاده کنیم و تأیید کنیم که پیکربندی یک نمونه Redis تازه شروع شده با پیکربندی پیش‌فرض چیست.

دستور شکل فلان(زیر) لیستی از کاربران را با همان قالبی که در فایل های پیکربندی Redis استفاده می شود قابل مشاهده است که به معنای دریافت ACL های فعلی تنظیم شده برای کاربران را گزارش می دهد. دوکلمه اول در هر سطر عبارت "user" و سپس نام کاربری است. کلمات بعدی قوانین ACL هستند که چیزهای مختلفی را توصیف می کنند. ما با جزئیات نشان خواهیم داد که قوانین چگونه کار می کنند، اما در حال حاضر کافی است بگوییم که کاربر پیش فرض برای دسترسی بدون نیاز به رمز عبور، برای دسترسی به هر کلید ممکن (~\*) پیکربندی شده است. کانال Pub/Sub (&\*)، و قادر به فراخوانی هر دستور ممکن (+@all) باشید. همچنین، در این مورد خاص کاربر پیش‌فرض، داشتن قانون nopass به این معنی است که اتصالات جدید به طور خودکار با کاربر پیش‌فرض بدون نیاز به تماس صریح AUTH احراز هویت می‌شوند.



شکل ‏2–17

در زیر لیستی از قوانین معتبر ACL آمده است. برخی از قوانین فقط کلمات واحدی هستند که به منظور فعال کردن یا حذف، یا انجام یک تغییر معین در کاربر ACL استفاده می‌شوند. قوانین دیگر پیشوندهای char هستند که با نام دستورات یا دسته‌ها، یا الگوهای کلیدی و غیره ترکیب می‌شوند.

**فعال و غیر مجاز کاربران:**

on: کاربر را فعال می­کند: امکان احراز هویت به عنوان این کاربر وجود دارد.

off: کاربر را غیرفعال می­کند: دیگر امکان احراز هویت با این کاربر وجود ندارد، اما اتصالات قبلاً تأیید شده همچنان کار خواهند کرد.

**دستورات مجاز و غیر مجاز:**

+<command>: دستور را به لیست دستوراتی که کاربر می تواند فراخوانی کند اضافه می­کند. قابل استفاده با | برای اجازه دادن به دستورات فرعی (به عنوان مثال "+config|get").

-<command>: دستور را از لیست دستوراتی که کاربر می تواند فراخوانی کند حذف می­کند. قابل استفاده با | برای مسدود کردن دستورات فرعی (به عنوان مثال "-config|set").

+@<category>: همه دستورات را در این دسته اضافه می­کند تا توسط کاربر فراخوانی شود، با دسته بندی های معتبر مانند @admin، @set، @sortedset، ... و غیره، لیست کامل را با فراخوانی ACL CAT مشاهده کنید. فرمان دسته ویژه @all به معنای تمام دستورات است، هم دستوراتی که در حال حاضر در سرور وجود دارد و هم آنهایی که در آینده از طریق ماژول ها بارگذاری شوند.

-@<category>: +@<category> را می پسندد اما دستورات را از لیست دستوراتی که مشتری می تواند فراخوانی کند حذف می کند.

+<command>|first-arg: یک آرگومان اول خاص از یک دستور غیرفعال را مجاز کنید. توجه داشته باشید که این فرم به عنوان منفی مانند -SELECT|1 مجاز نیست، بلکه فقط افزودنی است که با "+" شروع می شود.

همه دستورات: نام مستعار برای +@all. توجه داشته باشید که به معنای توانایی اجرای تمام دستورات آینده بارگذاری شده از طریق سیستم ماژول ها است.

nocommands: نام مستعار برای -@all.

کاربران را می توان به دو روش اصلی ایجاد و تغییر داد:

* با استفاده از ارسال دستور در خط فرمان ACL SETUSER.
* تغییر پیکربندی سرور، جایی که کاربران می‌توانند تعریف شوند را هنگام فعال سازی سرویس تغییر داد و سرور را دوباره راه‌اندازی شود.

برای استفاده از ردیس در برنامه ها و کنترل آن ردیس از زبان های متفاوتی پشتیبانی میکند ما در این پروژه از کتابخانه redis-py استفاده نمودیم که در فصل های بعدی این کتابخانه را بررسی خواهیم نمود.

### راه­اندازی Redis با داکر کامپوز

از پیش‌تر در بخش مروری بر داکر به یاد داریم که می­توان ایمیج های استاندارد را از داکرهاب، بارگذاری و مورداستفاده قرارداد، Redis یکی از ایمیج های استاندارد است که می­توان به‌سادگی از آن استفاده نمود، برای استفاده از این ایمیج با داکر کامپوز، تنها کافی است، پورت مخصوص به ارتباط با کانتینر آن را به کامپیوتر میزبان متصل کرده و فضایی در کامپیوتر میزبان برای ذخیره‌سازی اطلاعات برای آن اختصاص دهیم تا با شروع به کار مجدد کانتینر، اطلاعات ذخیره‌شده در آن از بین نرود، همچنین می­توان فایل تنظیمات مربوط به Redis را تغییر داد و در کانتینر قرارداد و دستور داد تا برنامه Redis بر اساس تنظیماتی که در کانتینر قرارگرفته، اجرا شود. در شکل ‏4–17 می­توان دستوراتی که در فایل docker-compose.yaml برای بالا آمدن کانتینر Redis نوشته‌شده را مشاهده کرد.

  cache:

    image: redis:latest

    container\_name: redis

    ports:

        - 6379:6379

    volumes:

        - ./redis/config/redis.conf:/redis.conf

        - ./redis/Data:/data

    command: [ "redis-server", "/redis.conf" ]

    networks:

      - app-tier

شکل ‏2–18- بخش مرتبط با Redis در داکر کامپوز

## سرویس MinIO

سلام سلام خاله بزغاله، الک سلام خاله بزغاله!

## جمع بندی

در بخش‌های فوق هر یک از نرم‌افزارهای به‌کاررفته در این پروژه را بررسی کردیم و ویژگی هایی که در اختیار ما قرار میدهند را دریافتیم ، دیدیم که میتوان با RabbitMQ بسته ها درون صف هایی قرارداد و جریان داده ایجاد کرد. همچنین دیدیم که با ردیس میتوان کاربران کنترل نمود و پارامترهایی را در آن ذخیره کرد و در پایان نیز دیدیم که میتوان اطلاعات ذخیره شده در سرور را در اختیار کاربران قرارداد حال با توجه به توضیحات فوق در فصل بعد این ابزارها را در کنار میچینیم و معماری پیشنهادی را تشریح میکنیم.

# فصل سوم طراحی و معماری برنامه های کاربردی

**طراحی و معماری برنامه های کاربردی**

در این فصل پیش سرویس ها و نحوه اجرای آن ها را بررسی نمودیم حال وقت آن رسیده که ابتدا معماری که ابزارهای فوق در کنار یکدیگر قعال خواهند بود را معرفی کنیم و سپس به سراغ طراحی هر یک از برنامه های کاربردی که کاربران میخواهند از آن استفاده کنند بپردازیم.

## طراحی معماری و چالش ها

بر اساس پروپوزال نیازمندی و هدف این پروژه آن است که بتوانیم اطلاعات دوربین ها دریافت کنیم سپس اطلاعات را پردازش شده را در اختیار کاربرنهایی قراردهیم. کاربر باید بتواند به اطلاعات ذخیره شده دوربین ها دسترسی داشته باشد. و اطلاعات دوربین ها را در زمان واقعی به همراه اطلاعات پردازش شده دریافت کند. برای رسیدن به هدف فوق معماری زیر در سه برنامه کاربردی ارائه شد. که در یک شبکه داخلی ، یک برنامه بر روی کامپیوتر سرور، یک برنامه بر روی سرور پرازشی و یک برنامه بر روی کلاینت اجرا شود.[[44]](#footnote-44)



شکل ‏3–1

در این معماری داکر را بر روی کامپیوتر سرور به اجرا خواهیم آورد پس سرویس ها در این کامپیوتر اجرا خواهند شد که شامل سرویس RabbitMQ، Redis، MinIO هستند. بهتر است ابتدا به اصلی ترین چالش این پروژه که پردازش در زمان واقعی است بپردازیم ولی پیش از آن که در چالش پردازش در زمان واقعی عمیق شویم بهتر است. راهکار انتقال تصویر در زمان واقعی را بررسی کنیم.

### چالش انتقال تصویر در زمان واقعی و تعیین سطح دسترسی

در فصل قبل دیدیم که میتوان توسط RabbitMQ مراکز توزیع داده تعریف کرد که اطلاعات را به آن ارسال نمود و صف هایی ایجاد کرد که بتوان به این مراکز متصل شده و اطلاعات را دریافت کنند. در این پروژه قصد داریم که چندین کاربر بر اساس سطح دسترسی خود بتوانند اطلاعات دوربین ها دریافت کنند. برای اینکه بتوان اطلاعات یک دوربین را بین چندین کاربر تقسیم نمود، نوع مرکز انتقال داده را باید از نوع خروج به همه تعیین نمود. بدین معنا که اگر صفی به این مرکز توزیع داده متصل شد، یک کپی از اطلاعاتی که داخل این مرکز توزیع قرار دارد به تمامی صف ها منتقل شود و همه کاربران بتوانند به اطلاعات دوربین ها دسترسی داشته باشند. ولی مسئله اینجاست که کدام کاربران اجازه دسترسی به اطلاعات این دوربین ها را دارند. یک راهکار این است که این مراکز توزیع اطلاعات هر کدام در یک ماشین مجازی که خود RabbitMQ در اختیار ما قرار میدهد قرار بگیرند تا امنیت اطلاعات تامین شود و فرایند احراز هویت برای دسترسی به اطلاعات دوربین ها توسط خود RabbitMQ انجام شود یا راهکار دیگر این است که همه مراکز توزیع داده در یک ماشین قراربگیرند و سطح دسترسی افراد در Redis تعیین شود و همچنین نام مراکز توزیع داده نیز در آن قرارگیرد و کاربران در Redis احراز هویت شوند و براساس احراز هویت خود نام دوربین هایی که میتوانند به آن دسترسی داشته باشند در اختیار آنها قرارگیرد که تنها به توانند به دوربین های مجاز خود متصل شوند. در مدل اول سطح امکانات سخت افزاری که توسط کاربر در سرور تعریف میشود بیشتر است. ولی روش دوم که از نظر سخت افزاری روش دیگری زیرساخت سخت افزاری بسیار مناسب است. در این پروژه از روش دوم برای انجام این کار استفاده شد که برای کاربران در redis سطح دسترسی تعریف میشود و همچنین برای آنها در ACL نام کاربری و رمز عبور تعریف میشود که بتوانند بعد از احراز هویت بر اساس سطح دسترسی خود به دوربین های مجاز خود دسترسی داشته باشند.

### چالش پردازش در زمان واقعی

در بخش قبل دیدیم که اطلاعات دوربین ها در مراکز توزیع اطلاعات RabbitMQ با نرخی مشخص وارد می­شوند یعنی با ایجاد یک صف میتوان اطلاعات دوربین ها را دریافت کرد. اجازه دهید مسئله را با یک مثال تشریح کنم، فرض کنید که اطلاعات نرخ 30 فریم بر ثانیه وارد مرکز اطلاعات میشوند. و ما قصد داریم که همه فریم های این تصویر را پردازش کنیم. این بدان معناست که مدل هوش مصنوعی ما باید هر فریم را در حداکثر 30/1 ثانیه یا ساده تر بگویم، در 33 میلی ثانیه پردازش کند. در صورتی که زمان پردازش از عدد فوق بیشتر شود نمیتواند از صفی که اطلاعات به آن وارد میشوند نمونه بردارد و صف پر خواهد شد و سیستم دچار اخلال میشود. پس راهکاری باید برای حل این مسئله ارائه شود. راهکار ساده این است که نیازی نیست همه فریم ها پردازش شوند. و از n فریم که از برداشته میشود تنها کافی است 1 فریم پردازش شود. بگذارید مثال فوق را برای n=15 تکرار کنم. اطلاعات با نرخ 30 فریم بر ثانیه وارد می­شوند و ما قصد داریم از هر 15 فریم تنها یک فریم را پردازش کنیم پس در یک ثانیه تنها 2 فریم پردازش خواهند شد ولی در عوض مدل هوش مصنوعی ما زمانی به اندازه 30/15 خواهد داشت که به بیان ساده تر 500 میلی ثانیه خواهد داشت. پس در صورتی که یک پارامتر با این ویژگی وارد مسئله کنیم. میتوان در عمل مدل های هوش مصنوعی متفاوتی را بکارگرفت.

پس برای راهکاری که برای چالش پردازش بکارگرفته شد این است که، پارامتری را در اختیار کاربر قرار خواهیم داد که بتواند متناسب با مدل هوش مصنوعی خود این پارامتر را بهینه کند.



### چالش ذخیره سازی اطلاعات

همون جا که اطلاعات رو میفرستیم به ربیت باید یه ترد جدا باز کنیم اطلاعات رو ذخیره هم بکنیم.

رم پر میشه باید چانک چانک بگیریم. ما یه دقیقه ای گرفتیم.

### چالش هماهنگ سازی برنامه ها

در این برنامه یه کاری میکنی اون یکی باید خبر دار بشود.

### چالش دریافت اطلاعات از دوربین ها

با opencv از رفتیم جلو.

به ازای هر دوربینی که قصد داریم اطلاعات آن را به جریان داده تبدیل کنیم، در RabbitMQ یک مرکز توزیع اطلاعات تعریف می­کنیم و برنامه ای مینویسیم که اطلاعات دوربین را دریافت به آن منتقل کند. نوع این مرکز توزیع اطلاعات باید از نوع ارسال به همه باشد. چرا که قصد داریم چندین کاربر بتوانند اطلاعات این دوربین را دریافت کنند. پس کابران تنها با ایجاد یک صف که این مرکز اطلاعات متصل هستند، میتوانند اطلاعات دوربین ها را دریافت کنند.

از آنجایی که قصد داریم اطلاعات ذخیره شده دوربین ها را به کاربر نمایش دهیم، باید برنامه ای نوشته شود که اطلاعات دوربین ها را نیز ذخیره کند و اطلاعات ذخیره شده را نیز توسط MinIO میتوانیم به کاربر نمایش دهیم، بدین شکل اطلاعات دوربین ها را به صورت منظم در یک دایرکتوری مشخص قرارداده و باکت های MinIO را به آن دایرکتوری متصل میکنیم. همچنین هنگامی که دوربینی جدید یا تغییری در آن ایجاد میکنیم در redis ذخیره میکنیم تا سایر برنامه های کاربردی از تغییرات موجود مطلع شوند. همچنین دسترسی کاربران را نیز در redis تعیین میکنیم تا فرایند احراز هویت کاربران انجام شود.

## گوجه فروشی حسین

شسشس

## خیار فروشی محمد

شسکخیدشس

## جمع بندی

کمسبیتدسمنتب

# فصل چهارم بکارگیری و نحوه کارکرد سیستم برای کاربران

**بکارگیری سیستم**

پیاده‌سازی کرده تا اتفاقی که هنگام عملی شدن پروژه رخ می­دهد را شبیه‌سازی کنیم.

## راه اندازی و پیشنیاز ها

سسشیی

## جمع‌بندی فصل چهارم

در این بخش به بررسی و آزمودن بخش‌های مختلف پروژه پرداختیم، ابتدا اهمیت هرکدام از حسگرها را با بررسی فرایند آموزش بر روی اطلاعات تنها یک حسگر انجام دادیم و دریافتیم که حسگر شتابسنج اهمیت بیشتری دارد. سپس با انجام آزمایشی دیگر بر روی اطلاعات هر دو حسگر، فرآیند آموزش را بدون به‌کارگیری پنجره‌بندی اتفاقی انجام دادیم و اثر آن را در تشخیص هر رخداد مشاهده نمودیم. در پایان نیز با کمک یک برنامه شبیه‌سازی، نحوه عملکرد سیستم دریافت و ذخیره‌سازی اطلاعات را در ابرمحاسباتی بررسی کردیم؛ و بسته‌هایی که یک راننده نمونه به این سیستم ارسال می­کند را ارزیابی کردیم.

# فصل پنجم نتیجه‌گیری و پیشنهادها



## نتیجه­گیری

در این رساله ما یک سیستم کامل برای پیاده‌سازی در شرکت‌های بیمه‌ای، استفاده در پلیس راهنمایی و رانندگی و شرکت­های حمل‌ونقل ارائه کردیم که با به‌کارگیری آن بتوان میزان ایمن بودن رانندگی افراد را سنجید و در یک شرکت بیمه‌ای، برای تعیین حق بیمه، در یک سیستم عادلانه متناسب با نوع رانندگی افراد برای آن‌ها مبلغ، تعیین نمود تا بتوان رانندگان را با انجام این روش به رانندگی ایمن تشویق کرده تا جاده‌ها و خیابان‌هایی ایمن‌تری ایجاد شود و آمار تصادفات شهری و تلفات جاده‌ای را کاهش یابد. برای این کار ابتدا مدلی با به‌کارگیری حسگرهای شتابسنج و ژیروسکوپ طراحی شد که میزان رخدادهای خطرناک رانندگی با دقتی قابل‌قبول تشخیص دهند. رخدادهایی همچون گردش‌به‌راست خطرناک باسقت 91 درصد (87 درصد[[45]](#footnote-45))، گردش‌به‌چپ خطرناک با دقت 95 درصد (83 درصد)، شتابگیری خطرناک با دقت 78 درصد (69 درصد)، ترمزگیری خطرناک با دقت 91 درصد (73 درصد)، تعویض لاین خطرناک به سمت راست با دقت 84 درصد (69 درصد)، تعویض لاین خطرناک به سمت چپ با دقت 83 درصد (73 درصد) به‌دست‌آمده است. سپس یک سیستم دریافت و ذخیره‌سازی اطلاعات طراحی شد که بتواند بسته‌های تولیدی توسط خودروها را در یک سامانه ابری ذخیره‌سازی نماید. برای این کار از نرم‌افزار RabbitMQ به‌عنوان مبادله گر اطلاعات و Redis به‌عنوان پایگاه داده موقت استفاده شد، همچنین برنامه با زبان پایتون نوشته شد که اطلاعات را از صف‌های RabbitMQ خوانده و به Redis منتقل کند. کل این مجموعه بر بستر کانتینرهای داکر طراحی شد تا توانایی انتقال به هر ارائه‌دهنده سرویس ابری را داشته باشد. در پایان نیز یک مدل مبتنی بر اطلاعات آماری طراحی شد که به رانندگان بر اساس زمان رانندگی و تعداد رویدادهای خطرناک تولیدشده امتیازی اطلاق کند. تا بتوان رانندگان را با یکدیگر بر اساس این امتیاز مقایسه کرد تا رانندگان ایمن را از رانندگان خطرناک متمایز کرده و حق بیمه‌ای متناسب با نوع رانندگی آن‌ها در نظر گرفت.

## پیشنهادها

در این بخش پژوهش‌هایی را که می­توان در ادامه راه این پژوهش انجام شود ذکرشده:

1-بهینه‌سازی با دقت بالاتر پارامترهای پنجره‌بندی اتفاقی برای تشخیص رخدادهای خطرناک

در این پروژه برای تشخیص رویدادهای خطرناک، از پنجره­بندی اتفاقی استفاده شد که با کمک پارامترهای آن، چندین مدل آموزش داده‌شده‌اند و مدلی که دقت بیشتری بر روی دادگان تست در اختیار ما قرار می­داد، به‌عنوان بهترین مدل انتخاب شد، ولی شیوه‌ای که برای تغییر این پارامترها بکار گرفته شد، برای پارامتر احتمال نگاه رو به عقب بافاصله‌ی 10 درصد و برای پارامتر میانگین حرکت پنجره روبه‌جلو عدد 5 درصد در نظر گرفته شد. به‌عنوان کارهای آینده می­توان گستره بررسی را افزایش داد تا مدل‌های بیشتری موردبررسی قرار گیرند و نتایج بهتری به‌دست­آید.

2-استفاده از مجموعه دادگان دیگری برای امتیازدهی رانندگان بر اساس میزان اهمیت رخداد خطرناک

در فرایند نمره­دهی به رانندگان، هنگامی‌که میزان احتمال خطرناک بودن رخدادها را محاسبه کردیم، این اعداد را به‌صورت مستقیم با یکدیگر جمع نمودیم و میزان خطرناک بودن، رخداد ترمزگیری خطرناک و گردش‌به‌چپ خطرناک را یکسان در نظر گرفتیم. درحالی‌که می­توان با کمک مجموعه دادگانی دیگر در زمینه تصادفات میزان اهمیت هر یک از رخداد را به‌صورت دقیق اندازه گرفت و در فرایند نمره دهی به رانندگان بهره برد.

3-استفاده از اطلاعات GPS و تشخیص رویدادهایی مربوط به‌سرعت غیرمجاز

برای فرایند تشخیص رخدادهای خطرناک، می­توان با اضافه کردن حسگر GPS سرعت حرکت خودرو را محاسبه کرد و رخدادهای مربوط به‌سرعت غیرمجاز در محل را اندازه‌گیری نمود. همچنین با کمک این اطلاعات می­توان محل‌های پرخطر را تشخیص داد.

4-طراحی یک داشبورد برای نمایش اطلاعات و ارزیابی عملکرد رانندگان

در این پروژه داشبوردی برای نمایش اطلاعات پردازش‌شده، طراحی نشد، ولی یکی از بخش­هایی که این پروژه نیاز دارد تا به مرحله عمل برسد، طراحی و پیاده‌سازی یک سرویس‌دهنده تحت وب است که اطلاعات پردازش‌شده و امتیازی که رانندگان کسب نموده‌اند را به آن­ها اطلاع‌رسانی کند.

# مراجع

|  |  |
| --- | --- |
| [1] | https://www.eghtesadnews.com/fa/tiny/news-273274, تعداد خودرو ها در سطح شهر, تهران: اقتصاد نیوز, 1397. |
| [2] | Docker overview, https://docs.docker.com/, 2021. |
| [3] | https://www.rabbitmq.com/documentation.html, RabbitMQ documentation 3.8.4, 2021. |

**Abstract**

Dangerous driving is one of the most important causes of car accidents. If drivers know that their behavior is being recorded while driving, they will be driving safer. Also, if it is possible to implement a system that encourages people to drive safely by being aware of the type of driving behaviors of individuals, this will lead to a reduction in traffic accidents. This project aims to implement a method for detecting driving events (including dangerous right and left lanes, dangerous lane changes left and right, dangerous braking, and acceleration) with the help of sensor information so that driver information Receive and store in a supercomputer and use a solution to calculate how safe people are driving. More precisely, using a new windowing method on time series, a decision tree on window similarities (using the Fast-DTW algorithm) was trained to detect driving events, then to receive and Data storage RabbitMQ was used as a message exchanger, and Redis was used as a temporary database. Finally, a statistical method was used to score and compare people's driving. The dangerousness of driving introduces them as a benchmark for comparison. With the help of the results presented in this study, an insurance plan can be implemented so that drivers who show safer driving pay fewer insurance premiums so that this plan can be used as an incentive for Improved driving culture was considered.

**Key Words:** Internet of Things, Machine learning, Driver Profiling, Driving culture

1. یک شرکتی است دستگاه هایی در زمینه کنترل ورود و خروج و قفل و باز کردن درب ها در اختیار استفاده کنندگان از این سرویس قرار میدهد. [↑](#footnote-ref-1)
2. فایل اسناد عملی هر دو پروفایل ضمیمه قرار گرفت. [↑](#footnote-ref-2)
3. Real-Time Streaming Protocol [↑](#footnote-ref-3)
4. Session [↑](#footnote-ref-4)
5. فایل اسناد عملی هر پروتکل فوق در ضمیمه قرار گرفت. [↑](#footnote-ref-5)
6. integration [↑](#footnote-ref-6)
7. در این پروژه سرویس MinIO به این شکل عمل می­کند. [↑](#footnote-ref-7)
8. در این پروژه سرویس redis بهاین شکل عمل می­کند. [↑](#footnote-ref-8)
9. در این پروژه دریافت اطلاعات از دوربین ها به نحوه پیاده سازی شده است. [↑](#footnote-ref-9)
10. Message Broker [↑](#footnote-ref-10)
11. در این پروژه سرویس RabbitMq به این شکل عمل می­کند. [↑](#footnote-ref-11)
12. Docker [↑](#footnote-ref-12)
13. container [↑](#footnote-ref-13)
14. Daemon process [↑](#footnote-ref-14)
15. Command line [↑](#footnote-ref-15)
16. CI(continuous integration) [↑](#footnote-ref-16)
17. CD(continuous delivery) [↑](#footnote-ref-17)
18. Portable [↑](#footnote-ref-18)
19. docker-compose [↑](#footnote-ref-19)
20. Producer [↑](#footnote-ref-20)
21. Consumer [↑](#footnote-ref-21)
22. Queue [↑](#footnote-ref-22)
23. Router(Exchange) [↑](#footnote-ref-23)
24. Channels [↑](#footnote-ref-24)
25. Binding [↑](#footnote-ref-25)
26. Direct Exchange [↑](#footnote-ref-26)
27. Fanout Exchange [↑](#footnote-ref-27)
28. Topic Exchange [↑](#footnote-ref-28)
29. Header Exchange [↑](#footnote-ref-29)
30. Durability [↑](#footnote-ref-30)
31. Unicast [↑](#footnote-ref-31)
32. Broadcast [↑](#footnote-ref-32)
33. publish/subscribe [↑](#footnote-ref-33)
34. Task [↑](#footnote-ref-34)
35. Routing key [↑](#footnote-ref-35)
36. Prefetch [↑](#footnote-ref-36)
37. Caching [↑](#footnote-ref-37)
38. Value [↑](#footnote-ref-38)
39. Sets [↑](#footnote-ref-39)
40. Sorted Sets [↑](#footnote-ref-40)
41. Hashes [↑](#footnote-ref-41)
42. Access Control List [↑](#footnote-ref-42)
43. domain specific language [↑](#footnote-ref-43)
44. میتواند همه برنامه ها بر روی یک کامپیوتر اجرا شوند ولی بهتر است که کامپیوتر سرور از ram بیشتری برخوردار باشد و کامپیوتر سرور پردازشی از کارت گرافیک قوی برخوردار باشد. و کامپیوتر کلاینت میتواند یک کامپیوتر ساده باشد. [↑](#footnote-ref-44)
45. اعداد دقت اعلام شده دقت precision و اعداد داخل پردانتز معیار دقت f1-score است. [↑](#footnote-ref-45)