**گزارش مربوط به نحوه ذخیره سازی اطلاعات در سرور**

**مقدمه:**

بر اساس پروپوزال نیازمند آن هستیم که بتوانیم اطلاعات را در سرور ذخیره سازی نماییم و اطلاعات ذخیره شده را به کاربر نمایش دهیم برای انجام این کار نیاز داریم که اطلاعات دوربین ها را در سرور ذخیره کنیم. و با کمک یک سرویس اطلاعات ذخیره شده را به کاربر نمایش دهیم. به طور عمومی ذخیره سازی فیلم ها به صورت Object Storage انجام میشود به معنا که فیلم به صورت یک فایل ذخیره میشود و Object Storage روشی برای ذخیره‌سازی فایل‌ها است که با کمک آن می‌توانید انواع فایل اعم از عکس و فیلم و متن را ذخیره کنید. مزیت آبجکت استورج نسبت به فایل سیستم محلی، این است که شما ‌می‌توانید فایل‌های خود را در فضای ابری آپلود و دانلود کنید. همچنین این امکان وجود دارد که برای فایل‌های آپلود شده سطح دسترسی تعیین کنید و در صورت نیاز فایل‌های خود را به صورت عمومی با همه کاربران اپ یا وب‌سایت خود به اشتراک بگذارید. معروف ترین ابزاری در حال حاضر این سرویس را ارائه میکند minIO است.

**minIO چیست:**

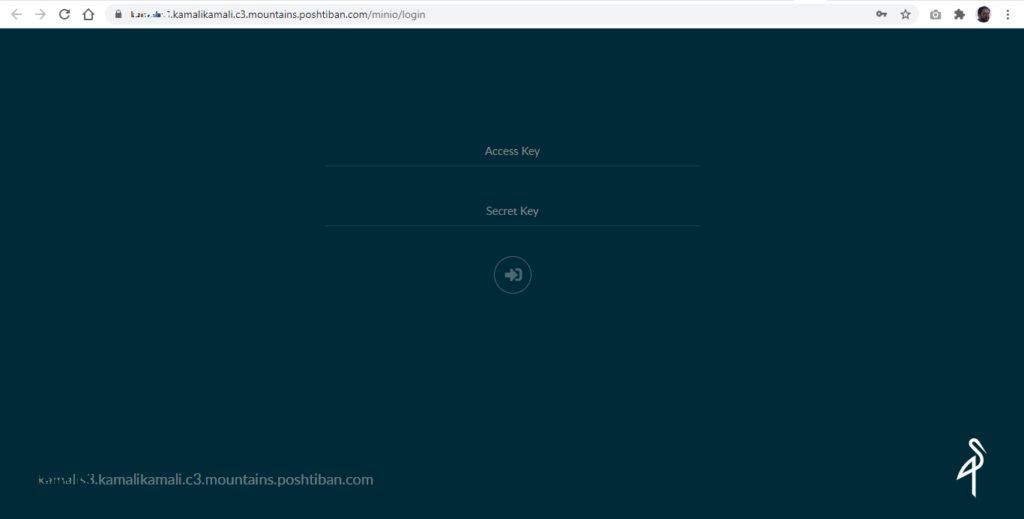
MinIO (به فارسی: مین آی او) یک نرم افزار متن باز، سمت سرور و سازگار با AWS S3 آمازون برای پیاده سازی Object Storage است. MinIO به صورت اختصاصی برای نگهداری داده در مقیاس بالا و ایجاد یک ابر خصوصی است که با زبان GO طراحی شده است. این ابزار با ساختار S3 سازگار است و S3 Compatible API در اختیار شما قرار می‌دهد.

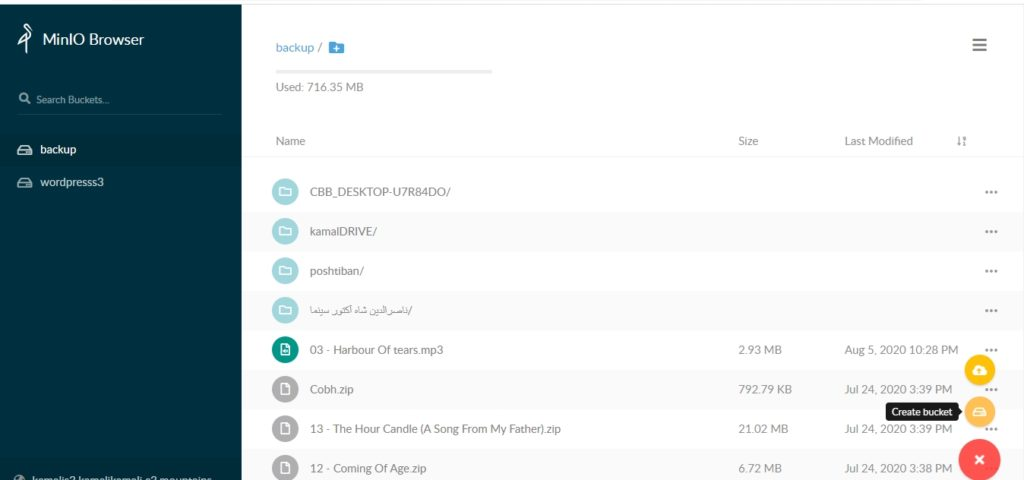
وقتی با داده‌های بی‌ساختار و استاتیک با حجم بالا مانند اطلاعات دوربین روبه‌رو باشید، بهترین روش ذخیره سازی دادها به صورت Object است و برای این که ذخیره سازی آبجکتی داشته باشید یکی از بهترین گزینه ها نصب MinIO در سمت سرور است. بدین ترتیب تصاویر، ویدیوها، بک‌آپ‌ها، لاگ‌ها و فایل‌های مختلف را با حجم چند کیلو بایت تا 5 ترابایت (تنها برای یک فایل) به راحتی و با سرعت ذخیره سازی می‌کنید.

MinIO قابلیت استقرار بر روی یک دیسک(هارد) واحد یا کلاسترهای توزیع شده را دارد. همچنین می‌تواند به عنوان رابط کاربری در راه‌کارهای ذخیره سازی ابجکتی مثل Azure Blob storage و آبجکت استوریج Google Cloud Platform یا در محیط داکر برای ارائه آبجکت استوریج به عنوان میکرو سرویس، بکار گرفته شود.

در ساختار S3 برای اتصال به Object Storage به سه مقدار کلید دسترسی، کلید خصوصی و آدرس اندپوینت نیاز دارید. همچنین با توجه به این که در روش ذخیره سازی آبجکتی داده‌ها در یک ساختار flat ذخیره می شوند مفهومی به نام دایرکتوری یا پوشه وجود ندارد بنابراین برای دسته بندی فایل‌ها در آبجکت استوریج Bucket (سطل) ایجاد می‌کنید . میتوانید به تعداد مورد نیاز خودتان bucket بسازید. در MinIO فایل‌های ذخیره سازی شده در باکت می‌توانند مسیر (path) داشته باشند تا زمانی که فایلی در مسیر وجود داشته باشد میتوانید آن مسیر را به عنوان یک پوشه در نظر بگیرید ولی باید به یاد داشته باشید که در ذخیره سازی آبجکتی پوشه یا دایرکتوری وجود ندارد.

این اپلیکیشن همانند rabbitMq یک پنل وب دارد که در تصویر زیر میتوانید آن را مشاهده کنید:





بر دسترسی به امکانات این سرویس کتابخانه ای ارائه شده که میتوان به سادگی به این سرویس دسترسی داشت و آن استفاده نمود. در لینک زیر میتوان api های ارائه شده توسط Minio را بررسی کرد:

<https://docs.min.io/docs/python-client-quickstart-guide.html>

که با نصب کتاب خانه minio این کار شروع خواهد شد.

**نحوه پیاده سازی**

برای پیاده سازی این سرویس ابتدا باید docker-compose که پیشتر دیدیم را آپدیت نمود و سوریس minio را به آن اضافه نمود و از آن استفاده کرد، تصویر زیر نحوه اضافه کردن آن است.

  minio:

    image: "minio/minio"

    restart: always

    environment:

      MINIO\_ACCESS\_KEY: admin

      MINIO\_SECRET\_KEY: admin1234

    volumes:

      - "./Data:/data"

    ports:

      - "9000:9000"

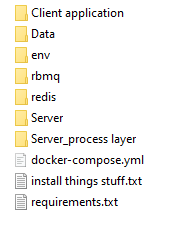
      - "9001:9001"

    command: server /data --console-address ":9001"

    networks:

      - app-tier

بدین ترتیب که ابتدا فایل Image آن را دریافت کرده و تگ ریست آن را در صورتی که سرور دچار اخلال شود فعال میکنیم که سرویس متوقف نشود. سپس برای کاربر admin آن رمز دیفالت admin1234 را تعیین میکنیم.

نکته قابل توجه نحوه bind کردن و محیط bind کردن اطلاعات است که در پوشه Data داخل پروژه را به بخش data داخل کانتینر است

این بدان معناست که اطلاعات دوربین ها را در پوشه Data ذخیره کنیم اطلاعات آن ها از طریق minIO در شبکه قابل دسترسی هستند.

پس در اپلیکشین دریافت اطلاعات دوربین ها که پیش تر دیدیم یک thread دیگر باز میکنیم و اطلاعات دوربین ها را هر یک دقیقه شروع به ذخیره در این پوشه بر اساس نام دوربین میکنیم، کد مرتبط با این بخش را میتوان در بخش زیر مشاهده نمود:

class file\_recorder(Thread):

    def \_\_init\_\_(self,file\_dump\_queue):

        Thread.\_\_init\_\_(self)

        self.queue=file\_dump\_queue

    def run(self):

        one\_min\_counter=0

        s\_flag=False

        fourcc = cv.VideoWriter\_fourcc(\*'XVID')

        if not os.path.isdir(RECORD\_PATH):

            os.mkdir(RECORD\_PATH)

        while(True):

            frame=self.queue.get(block=True)

            if one\_min\_counter==0:

                if s\_flag:

                    out.release()

                else:

                    s\_flag=True

                name=str(time.time()).split('.')[0]

                out = cv.VideoWriter(RECORD\_PATH+f'/{name}.avi', fourcc, RECORD\_FPS ,

                                     (frame[0][0][1]\*8, frame[0][0][0]\*8 )

                                     )

                one\_min\_counter=10

                # print(f'{EXCHANGE\_NAME}\_{name}.avi')

            if frame[0][0][2]==180:

                one\_min\_counter-=1

            out.write(frame)