راهنمای استفاده کارا

> برای آشنایی بیشتر با ابزار کارا و چگونگی پیدایش آن می‌توانید به سند (معرفی کارا) مراجعه کنید.
>
> برای استفاده از کارا برای ایجاد گزارشات روزانه می‌توانید به سند (گزارش روزانه کارا) مراجعه کنید.
>
> پس از نصب و راه اندازی کارا با استفاده از این سند (راهنمای نصب Kara-v1) می‌توان ابزار ها را مطابق ذیل استفاده کرد.

۱- شرح ابزار ها

۱-۱- آشنایی اولیه با ابزارهای موجود در پروژه

پروژه کارا تشکیل شده از **شش ابزار اجرایی یا عملیاتی** و **یک ابزار مدیریتی** است که برای هر یک از آنها دایرکتوری های خاصی ساخته شده است که درون آن فایل های مورد نیاز ابزار و سورس کد هر یک قرار دارد.

- **manager:** این ابزار وظیفه مدیریت مجموعه ای از ابزار های موجود را دارد و آنها را بسته به نیاز کاربر در حالت های مختلف به صورت خودکار اجرا می‌کند.
- **config_gen**: این ابزار وظیفه ساخت قالب یا تمپلیت های متفاوت با کانفیگ های مختلف را برای cosbench و swift به صورت خودکار را دارد.
- **mrbench**: این ابزار وظیفه اجرای تست های کارایی هیولا را دارد این تست ها با استفاده از ابزار cosbench بر روی هیولا اجرا می‌شوند.
- **status_reporter**: این ابزار وظیفه گزارشگیری از دیتابیس و نتایج تست و یا رخداد های موجود در هیولا در قالب فایل های csv و عکس را دارد.

- **monstaver**: این ابزار وظیفه بکاپ و ری استور داده های موجود در بازه های زمانی تست و یا رخداد از دیتابیس سامانه بینا را دارد.
- **analyzer**: این ابزار وظیفه تحلیل و تجمیع گزارش های گرفته شده از هیولا و مقایسه و خطایابی در کانفیگ های آن را دارد.
- **report_recorder**: ابزار وظیفه مستندسازی نتایج تست ها و رخدادها را در سامانه کاتب دارد.
- **results**: دایرکتوری محل ذخیره نتایج تست ها و گزارش های هیولا.
- **configure**: دایرکتوری ابزارهای فرعی (مورد استفاده فقط در موارد خاص)

۱-۲- ابزار مدیریتی manager

این ابزار مانند مرکز کنترل سیستم نظارت عمل می‌کند و تمام ابزار های ساخته شده را به ترتیب پایپ لاین و سناریو های مختلف اجرا و ورودی های مورد نیاز هرکدام را به آن می‌دهد. برای کاربری راحت تر و سریع تر بهتر است از این ابزار استفاده کنید تا این ابزار دیگر ابزار ها و نرم افزار ها را به صورت خودکار اجرا کند و احتمال خطای انسانی نیز کاهش یابد. برای استفاده از این ابزار فقط کافی است یک فایل سناریو که با توجه به نیاز کاربر به ابزار ها شخصی سازی شده به ورودی این ابزار داده شود.

در ادامه به شرح کامل این فایل پرداخته شده است:

فایل سناریو پیش فرض با فرمت yaml در دایرکتوری/manager/scenario_dir/ قرار دارد و می توان آن را بسته به نیاز ویرایش کرد و چندین نوع مختلف از آن ساخت. از ویژگی های مثبت استفاده از ابزار مدیریتی می‌توان به موارد ذیل اشاره کرد:

o_file.yaml

```
io:
    - Config_gen:
                {options}
    - Mrbench:
                {options}
    - Status-Reporter:
                {options}
    - Monstaver:
                {options}
    - Status_Analyzer:
                {options}
    - Report_Recorder:
                {options}
```

- مدیریت آسانتر ابزارها
- امکان ایجاد حالت های مختلف
- تعیین ورودی و خروجی هر ابزار
- مشخص کردن ترتیب اجرای ابزارها
- اجرای یک فرایند کامل به صورت خودکار
- مشخص کردن امکانات مورد نیاز در هر ابزار

در فایل سناریو ترتیب اجرای ابزار ها مشخص شده که قابل تغییر است و در بخش هر ابزار ورودی ها و خروجی های مورد نیاز آن قابل تعریف است که هر کدام در ذیل توضیح داده شده اند:

> **نکته: قبل از تنظیم سناریو و اجرای آن نیاز است فایل کانفیگ هر یک از ابزار ها در مسیر /etc/kara/ تنظیم شود و اطلاعات مورد نیاز درون آنها قرار گیرد. (برای آشنایی با فایل های کانفیگ به بخش موجود برای هر ابزار در سند همین مراجعه کنید)**

- **Config_gen**: در بخش این ابزار لیستی از قالب های ورودی دریافت می شود این لیست می تواند شامل چندین قالب workload تست با شماره گذاری مشخص مانند نمونه زیر باشد و یا فایل های کانفیگ swift و در بخش بعدی آن مسیر خروجی برنامه مشخص شده.

> **نکته: در صورتی که در مسیر خروجی ابزار فایل هایی از قبل وجود داشته باشد ابزار ابتدا سوالی در مورد حذف یا نگهداشتن فایل های قبلی می‌پرسد.**

```
- Config_gen:
        conf_templates:    # list of cosbench and swift config file
            - /path/to/kara/config_gen/workloads.xml__1
            - /path/to/kara/config_gen/workloads.xml__2
            - /path/to/kara/config_gen/configs.conf
            - /path/to/kara/config_gen/object_swift.conf

        output_path: /path/to/kara/config_gen/out/
```

- **Mrbench**: در بخش این ابزار مسیر خروجی نهایی که درون آن یک دایرکتوری یکتا برای هر تست با نامی مشابه بازه زمانی آن تست ایجاد می‌شود و در خط های بعدی حالت اجرای status_reporter و monstaver هم زمان با اجرای هر تست نوع خروجی آنها مشخص شده که به چه صورت باشد که در مثال ذکر شده خروجی csv , image به همراه بکاپ دیتابیس و اطلاعات سخت افزاری و نرم افزاری در بکاپ وجود دارد و در گام بعدی دایرکتوری کانفیگ های swift و ring به صورت مجزا از یکدیگر مشخص شده است. در دایرکتوری conf_dir می‌توان چندین فایل کانفیگ قرار داد و همچنین برای هر یک از دایرکتوری های لیست رینگ نیز می‌توان چندین فایل قرار داد.

> **نکته: در صورتی که قبل از این ابزار config-gen اجرا شده باشد و دارای کانفیگ های swift باشد نیازی به استفاده از قسمت conf_dir نیست ولی فایل های رینگ باید در لیست موجود ذکر شوند.**

> **نکته: در صورتی که در مسیر خروجی ابزار فایل هایی از قبل وجود داشته باشد ابزار ابتدا سوالی در مورد حذف یا نگهداشتن فایل های قبلی می‌پرسد.**

> **نکته: اگر در هنگام اجرای این بخش از سناریو و ابزار mrbench به دلایلی فرآیند اجرای آن متوقف شود برنامه آخرین مرحله اجرا شده را ذخیره می‌کند و در اجرای بعدی از کاربر برای ادامه مراحل قبل یا شروع از ابتدا سوال می‌نماید.**

> **نکته: در زمان اجرای این ابزار در سناریو در صورت اجرای ابزار status_reporter و ساخت csv دو فایل تجمیع شده از تمام تست ها با نام های (merged , merged_info) نیز در مسیر خروجی و دایرکتوری analyzed ایجاد می‌شوند که در ابزار های بعدی برای ایجاد سند در کاتب کاربرد دارند.**

```
- Mrbench:
        output_path: /path/to/kara/results/

        # call status_reporetr and monstaver for each test
        status_reporter: csv,img    # values = none - csv - csv,img
        monstaver: backup,info    # values = none , backup,info - backup - info

        #conf_dir: /path/to/kara/config_gen/out/
        ring_dirs:    # list of directory include ring files
            - /path/to/kara/mrbench/ring/r1/
            - /path/to/kara/mrbench/ring/r2/
            - /path/to/kara/mrbench/ring/r3/
```

- **Status_reporter**: اگر نیاز باشد این ابزار به صورت مستقل کار کند باید از بخش مخصوص آن استفاده کرد در این بخش می‌توان برای ورودی فایلی که شامل لیستی ار بازه های تست مورد نیاز است و یا لیستی از زمان های مورد نیاز در فرمت های گوناگون نوشتاری (tehran timestamp) دریافت شده و در بخش بعدی مشخص شود که به تحلیل و ایجاد سند در کاتب و تصویر گراف از بازه های زمانی ذکر شده نیاز است یا خیر و بعد از آن مسیر خروجی نهایی مشخص شود. قسمت report_recorder در این ابزار بیشتر برای گزارشات روزانه کاربرد دارد و در صورتی که به آن نیاز نباشد کل این بخش باید کامنت شود.

> **نکته: در این صورت استفاده از گزینه analyze_csv یک نمونه csv دارای تحلیل نیز از گزارش ها ساخته می‌شود پس حتما قبل از اجرای آنها ابزار status_analyzer را بسته به متریک ها یا measurement های موجود در فایل های متریک status_reporter حتما کانفیگ کنید.**

```
    - Status-Reporter:
        time_list:
            - now-1d,now
            #- now-24h,now
            #- "2023-09-10 23:59:59,2023-09-11 23:59:59"
            #- ./time.txt
        image: True
        analyze_csv: True
        report_recorder:
            output_htmls_path: path/to/kara/report_recorder/output_htmls/
            cluster_name: "کارا"
            kateb_tags:
                - "تست"
                - "کارایی"
                - "گزارشها"
            kateb_list_page: "name of a page"  # append all pages title to this kateb page
        output_path: ./../results/
```

- **Monstaver**: اگر نیاز باشد این ابزار به صورت مستقل کار کند باید از بخش مخصوص آن استفاده کرد در این بخش می توان برای زمان بکاپ گیری فایلی شامل لیستی از بازه های زمانی مورد نیاز و یا لیستی از زمان ها با فرمت نوشتاری گوناگون (tehran timestamp) و همچنین دایرکتوری دلخواهی که نیاز است در کنار بکاپ ها باشد را مشخص کرد در قسمت بعد می‌توان با فعال کردن حالت batch_mode پس از اجرای تمام تست ها در ابزار mrbench عملیات بکاپگیری برای یک دسته تست انجام شود و در قسمت بعدی هم می‌توان نوع عملیات را مشخص کرد که بکاپ از دیتابیس باشد یا به همراه اطلاعات سخت افزاری و نرم افزاری سرور های هیولا و یا عملیات بازیابی.

> **نکته: در صورتی که قبل از monstaver از status_reporter استفاده شده باشد و بازه های زمانی آنها یکسان باشند می‌توان فقط یک لیست بازه زمانی برای status_reporter ایجاد کرد و ابزار monstaver نیز از همان استفاده کند برای جلوگیری از تکرار آنها در فایل سناریو.**

```
    - Monstaver:
            input_path: ./../results
            time_list:
                #- now-1h,now
                #- now-3d,now-2d
                #- "2023-09-11 00:00:00,2023-09-11 23:59:59"
                #- ./time.txt
            batch_mode: True # value = True for all backup modes and False for restore
            operation: backup # values = restore , backup,info - backup - info
```

- **Status_analyzer**: در بخش های این ابزار می‌توان عملیات های تجمیع و تحلیل را مشخص کرد برای تجمیع کردن فایل های csv گزارش های گرفته شده از دیتابیس (influxdb) سرویس مانیتورینگ هیولا یا هر نوع csv دیگری می‌توان همه فایل ها را در یک دایرکتوری مشترک انتقال داد و یا لیستی از آنها و مسیرشان نوشت تا فایل تجمیع شده آنها در خروجی ذکر شده ذخیره شود. برای بخش تحلیل نیز می‌توان همان فایل تجمیع شده یا هر نوع csv دیگری را برای ورودی آن انتخاب کرد و مشخص کرد پس از تحلیل ستون های اولیه در فایل ورودی از فایل نتیجه تحلیل حذف شوند یا خیر.

```
    - Status_Analyzer:
            output_path: /path/to/kara/results/analyzed/
            merge: True
            merge_csv: "./../result/*"  # list of csv file --> '/path/csv1,/path/csv3,/path/csv3,' or
/path/*

            analyze: True
            keep_source_columns: False    # keep original columns in source csv file
            analyze_csv: "/path/to/kara/results/analyzed/merged.csv"
```

- **Report_recorder**: در بخش اول این ابزار می‌توان قالب html برای گزارش های سخت افزاری و نرم افزاری سرور هیولا را مشخص کرد و برای ایجاد گزارش از تست های گرفته شده از هیولا باید فایل های تجمیع شده از همه تست های گرفته شده را به عنوان ورودی این بخش داد و در کنار آن می‌توان مسیر دایرکتوری والد گزارش ها را داد تا تصاویر از آنها استخراج شود . در قسمت بعدی مسیر خروجی همه html های ساخته شده و دایرکتوری بکاپ یکی از تست های گرفته شده برای استخراج داده های سخت افزاری و نرم افزاری را مشخص کرد در بخش بعد در صورت نیاز می‌توان گزارش ها را در کاتب آپلود کرد و یا فقط فایل های ساخته شده آن ها را ابزار به صورت لوکال ذخیره کند و در آخر باید اسم دلخواه کلاستر هیولا و سناریو ایجاد شده را ذکر کرد تا تیتر سند های تولید شده بر اساس آنها نوشته شود. در قسمت آخر در صورت نیاز می‌توان اسم صفحه ای موجود در کاتب را که نیاز داریم اسم صفحات ساخته شده در آن لیست شود را بنویسیم.

> **نکته: در صورتی که این ابزار در یک سناریو همراه با monstaver یا mrbench اجرا شود دیگر نیازی به مشخص کردن آدرس فایل بکاپ در قسمت configs_dir نیست چون مسیر آن از ابزار قبلی دریافت می‌شود و همچنین اگر همراه با status_reporter و mrbench اجرا شود مسیر عکس های ساخته شده نیز خودکار از آن ابزار دریافت می‌شود و نیاز به استفاده از images_path در بخش این ابزار و در فایل سناریو نیست.**

```
    - Report_Recorder:
            create_html: True

            hardware_template: /path/to/kara/report_recorder/input_templates/hardware.html
            software_template: /path/to/kara/report_recorder/input_templates/software.html

            monster_test:
                report: True
                merged: /path/to/kara/manager/merged.csv
                merged_info: /path/to/kara/manager/merged_info.csv
                images_path: /path/to/kara/results/

            output_path: /path/to/kara/report_recorder/output_htmls/
            configs_dir: /tmp/influxdb-backup/backup_dir/

            upload_to_kateb: True
            cluster_name: kara
            scenario_name: performance
            kateb_list_page: "name of page"  # append all pages title to this kateb page
```

۱-۲-۱- مثال هایی از فایل سناریو:

۱-۲-۱-۱- سناریو کامل: اجرای ابزار ها از ایجاد فایل های کانفیگ swift و workload در cosbench تا اجرای تست و دریافت گزارش csv و تصویری گراف و بکاپ گیری از دیتابیس (influxdb) و اطلاعات سخت افزاری و نرم افزاری سرور های هیولا و تا تحلیل و تجمیع و ساخت سند در کاتب.

```
scenario:

    - Config_gen:
            conf_templates:    # list of cosbench and swift config file
                - /path/to/kara/config_gen/workloads.xml__1
                - /path/to/kara/config_gen/workloads.xml__2
                #- /path/to/kara/config_gen/configs.conf
                - /path/to/kara/config_gen/object_swift.conf

            output_path: /path/to/kara/config_gen/out/

    - Mrbench:
            output_path: /path/to/kara/results/

            # call status_reporetr and monstaver for each test
            status_reporter: csv,img    # values = none - csv - csv,img
            monstaver: backup,info    # values = none , backup,info - backup - info
```

```
            #conf_dir: /path/to/kara/config_gen/out/
            ring_dirs:    # list of directory include ring files
                - /path/to/kara/mrbench/ring/r1/
                #- /path/to/kara/mrbench/ring/r2/
                #- /path/to/kara/mrbench/ring/r3/

    - Status_Analyzer:
            #output_path: /path/to/kara/results/analyzed/
            #merge: True
            #merge_csv: "./../result/*"  # list of csv file --> '/path/csv1,/path/csv3,/path/csv3,' or
/path/*

            analyze: True
            keep_source_columns: True    # keep original columns in source csv file
            analyze_csv: "/path/to/kara/results/analyzed/merged.csv"

    - Report_Recorder:
            create_html: True

            hardware_template: /path/to/kara/report_recorder/input_templates/hardware.html
            software_template: /path/to/kara/report_recorder/input_templates/software.html

            monster_test:
                report: True
                merged: /path/to/kara/manager/merged.csv
                merged_info: /path/to/kara/manager/merged_info.csv
                images_path: /path/to/kara/results/

            output_path: /path/to/kara/report_recorder/output_htmls/
            #configs_dir: /tmp/influxdb-backup/backup_dir/

            upload_to_kateb: True
            cluster_name: kara
            scenario_name: complete_scenario
            kateb_list_page: "name of page"  # append all pages title to this kateb page

log:
   level: info  # values = debug - info - warning - error - critical
```

۱-۲-۱-۲- سناریو دریافت گزارش روزانه از هیولا:

به دلیل پر استفاده بودن این سناریو سندی مجزا برای آن ایجاد شده است: (گزارش روزانه کارا)

```
        - Status-Reporter:
            time_list:
                - now-1d,now
                #- now-24h,now
                #- "2023-09-10 23:59:59,2023-09-11 23:59:59"
                #- ./time.txt
            image: True
            analyze_csv: True
            report_recorder:
                output_htmls_path: path/to/kara/report_recorder/output_htmls/
                cluster_name: "کارا"
                kateb_tags:
                    - "تست"
                    - "کارایی"
                    - "گزارشها"
                kateb_list_page: "name of a page"   # append all pages title to this kateb page
            output_path: ./../results/
```

۱-۲-۱-۳- سناریو اجرای تست کارایی هیولا:

در صورت ساخت فایل های تمپلیت cosbench و کانفیگ swift در زمان اجرای تست:

```
scenario:

    - Config_gen:
            conf_templates:    # list of cosbench and swift config file
```

```
            - /path/to/kara/config_gen/workloads.xml__1
            - /path/to/kara/config_gen/workloads.xml__2
            #- /path/to/kara/config_gen/configs.conf
            - /path/to/kara/config_gen/object_swift.conf

        output_path: /path/to/kara/config_gen/out/

- Mrbench:
        output_path: /path/to/kara/results/

        # call status_reporetr and monstaver for each test
        Status_Reporter: csv,img    # values = none - csv - csv,img
        monstaver: backup,info    # values = none , backup,info - backup - info

        #conf_dir: /path/to/kara/config_gen/out/
        ring_dirs:    # list of directory include ring files
            - /path/to/kara/mrbench/ring/r1/
            - /path/to/kara/mrbench/ring/r2/
            - /path/to/kara/mrbench/ring/r3/
```

در صورت استفاده از فایل های از قبل ساخته شده تمپلیت cosbench و کانفیگ swift در اجرای تست:

```
scenario:

- Mrbench:
        output_path: /path/to/kara/results/

        # call status_reporetr and monstaver for each test
        status_reporter: csv,img    # values = none - csv - csv,img
        monstaver: backup,info    # values = none , backup,info - backup - info

        conf_dir: /path/to/kara/config_gen/out/
        ring_dirs:    # list of directory include ring files
            - /path/to/kara/mrbench/ring/r1/
            - /path/to/kara/mrbench/ring/r2/
            - /path/to/kara/mrbench/ring/r3/
```

۱-۲-۱-۴- سناریو ساخت سند در کاتب:

در صورت وجود داشتن فایل بکاپ از اطلاعات سخت افزاری و نرم افزاری هیولا و فایل های تجمیع شده از گزارش های تست و رخداد:

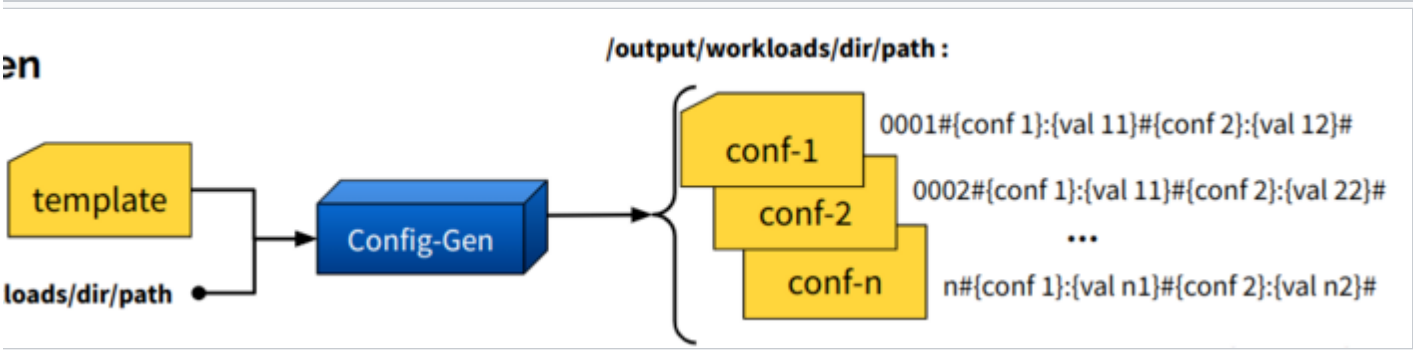
```
scenario:

- Report_Recorder:
        create_html: True

        hardware_template: /path/to/kara/report_recorder/input_templates/hardware.html
        software_template: /path/to/kara/report_recorder/input_templates/software.html

        monster_test:
            report: True
            merged: /path/to/kara/manager/merged.csv
            merged_info: /path/to/kara/manager/merged_info.csv
            images_path: /path/to/kara/results/

        output_path: /path/to/kara/report_recorder/output_htmls/
        configs_dir: /tmp/influxdb-backup/backup_dir/

        upload_to_kateb: True
        cluster_name: kara
        scenario_name: performance
        kateb_list_page: "name of page"  # append all pages title to this kateb page
```

۱-۲-۲- روش استفاده از ابزار:

```
python3 manager.py -sn scenario_dir/manager_scenario.yaml
```

۳-۱- ابزار config_gen

این ابزار وظیفه ایجاد کانفیگ های workload و تست های متفاوت را برای cosbench دارد. این برنامه در دایرکتوری config_gen وجود داشته و ورودی آن یک فایل با فرمت های استاندارد cosbench و کانفیگ های swift است و از این فایل کانفیگ با توجه به نوع نوشتن آن می تواند چندین فایل کانفیگ با مقادیر متفاوت ایجاد کند تا روند ساخت حالت های مختلف آسان تر و سریع تر شود.



روند کار config-gen

تغییر در مقادیر کانفیگ:

نمونه فایل اولیه قالب ورودی cosbench برای ساخت قالب های متفاوت از آن:

```
<?xml version="1.0" encoding="UTF-8" ?>
<workload name="swift-sample" description="sample benchmark for swift">
  <storage type="swift" />
  <auth type="swauth"
config="username=test:tester;password=testing;auth_url=http://0.0.0.0:8080/auth/v1.0"/>
  <workflow>
    <workstage name="init">
      <work type="init" workers="13" config="cprefix=?1L5s;containers=r(1,13)" />
    </workstage>
    <workstage name="main">
      <work name="main" workers="#1{8,16}concurrency#" runtime="60">
        <operation type="write" ratio="100" config="containers=r(1,13);cprefix=?
1L5s;objects=s(1,100);sizes=c(#2{128,256}objSize#)KB" />
      </work>
    </workstage>
    <workstage name="dispose">
      <work type="dispose" workers="13" config="containers=r(1,13);cprefix=?1L5s" />
    </workstage>
  </workflow>
</workload>
```

در ذیل بخشی از یک فایل قالب ورودی cosbench و اجرای workload نمایش داده شده است در آن بخش هایی وجود دارند که مشخص کننده متغییر های کانفیگ و تعداد عناصر آن است مانند تعداد size , worker و غیره. روند کار به این صورت است که از مقادیر داخل براکت {8,16} و {256,128} لیستی تهیه می‌شود و در هر بار ساخت فایل جدید یکی از آنها برای مقدار آن عنصر قرار می‌گیرند و به بیان دیگر به ازای هر کدام از این مقادیر حلقه ساخت فایل یکبار تکرار می‌شود . اگر در بخش های مختلف بعد از # اول یک عدد بیرون از براکت ها نوشته شد نمایانگر همروندی میان لیست های تشکیل شده است اگر اعداد یکسان بودند مقادیر داخل لیست ها مختلف متناظر با یکدیگر در فایل جدید قرار می‌گیرند اگر یکسان نبودند متفاوت و غیر همروند . در مرحله دیگر بعد از براکت ها متن یا اسمی نوشته شده که نمایانگر اسم آن قسمت از کانفیگ است که بعدا از آن برای نامگذاری فایل های کانفیگ استفاده می‌شود.
#concurrency{8,16}1# و **#objSize{256,128}2#**

```
<work name="main" workers="#1{8,16}concurrency#" runtime="120">
<operation type="write" ratio="100" config="containers=r(1,13);cprefix=?
1L5s;objects=s(1,10000);sizes=c(#2{256,128}objSize#)KB" />
```

برای نمونه این بخش از فایل کانفیگ ورودی در خروجی خود ۴ فایل کانفیگ جدید با کانفیگ های متفاوت را با این نوع نامگذاری ایجاد می کند: **concurrency:8#objSize:128##0001**

```
<work name="main" workers="8" runtime="120">
<operation type="write" ratio="100" config="containers=r(1,13);cprefix=?
1L5s;objects=s(1,10000);sizes=c(128)KB" />
```

0002#concurrency:8#objSize:256#

```
<work name="main" workers="8" runtime="120">
<operation type="write" ratio="100" config="containers=r(1,13);cprefix=?
1L5s;objects=s(1,10000);sizes=c(256)KB" />
```

0003#concurrency:16#objSize:128#

```
<work name="main" workers="16" runtime="120">
<operation type="write" ratio="100" config="containers=r(1,13);cprefix=?
1L5s;objects=s(1,10000);sizes=c(128)KB" />
```

0004#concurrency:16#objSize:256#

```
<work name="main" workers="16" runtime="120">
<operation type="write" ratio="100" config="containers=r(1,13);cprefix=?
1L5s;objects=s(1,10000);sizes=c(256)KB" />
```

تغییر در نامگذاری های object و container:

با استفاده از oprefix و cprefix می‌توان برای آبجکت ها و کانتینر ها اسامی قرار داد و آنها را دسته بندی کرد و یا مواردی یکتا از آنها ساخت، برای ساخت آنها می‌توان از کاراکتر های عددی و حروفی استفاده کرد، برای این کار می‌توان مقادیری را به جای اسم آن ها نوشت تا ابزار این کار را به صورت خودکار انجام دهد. برای مثال: ?1L5s یعنی ساخت مقداری تصادفی با شماره id 1 که اگر این شماره در جای دیگری استفاده شود همین مقدار نیز برای آن تکرار می‌شود. L یعنی طول متن اسم و عدد بعد از آن نمایانگر طول اسم و تعداد کاراکتر است و s به معنی نوع آن است که string بوده و اگر از d استفاده شود یعنی decimal و مقدار عددی.

```
<workstage name="init">
      <work type="init" workers="13" config="cprefix=?1L5s;containers=r(1,13)" />
</workstage>
```

خروجی کانفیگ ذکر شده برای cprefix ساخته شده:

```
<work name="main" workers="16" runtime="60">
        <operation type="write" ratio="100"
config="containers=r(1,13);cprefix=FP9I4;objects=s(1,10000);sizes=c(128)KB" />
      </work>
```

در مواردی که نیاز است فایل های کانفیگ swift تغییر پیدا کنند و نمونه های مختلف از آنها ساخته شود که می‌توان مانند مثال زیر عمل کرد.

نمونه فایل اولیه کانفیگ proxy-server در swift:

```
[DEFAULT]
bind_ip = 0.0.0.0
bind_port = 8080
workers = #1{4,8,16}#
user = swift

log_statsd_host = controller
log_statsd_port = 8125
log_statsd_default_sample_rate = 1.0
log_statsd_sample_rate_factor = 1.0
log_statsd_metric_prefix = swift
```

یکی از فایل ها ساخته شده از کانفیگ ذکر شده در بالا:

```
[DEFAULT]
bind_ip = 0.0.0.0
bind_port = 8080
workers = 4
user = swift

log_statsd_host = controller
log_statsd_port = 8125
log_statsd_default_sample_rate = 1.0
log_statsd_sample_rate_factor = 1.0
log_statsd_metric_prefix = swift
```

۱-۳-۱- روش استفاده از ابزار:

آرگومان های ورودی:

i- فایل قالب ورودی

o- مسیر دایرکتوری خروجی

```
python3 config-gen.py -i input.txt -o <output dir>
```

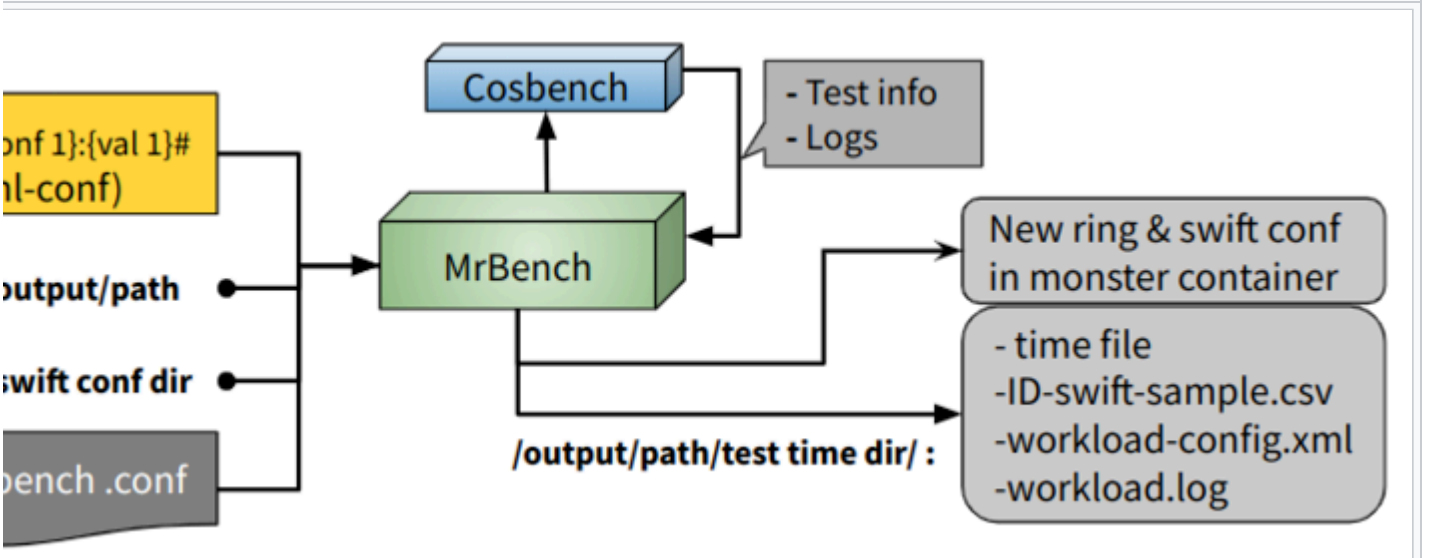
۱-۴- ابزار mrbench

این ابزار وظیفه اجرای کانفیگ های workload و ارسال آنها به cosbench را دارد و پس از اجرای تست و workload برای هر یک از آنها یک دایرکتوری یکتا بر اساس بازه زمانی اجرای آن در یک دایرکتوری والد ایجاد کرده و نتایج و اطلاعات به دست آمده از cosbench را در آن ذخیره می‌کند. این ابزار توانایی تغییر فایل های گانفیگ swift و ring را نیز دارا است و می‌تواند فایل هایی با فرمتهای .gz و .builder و .conf را برای این کار از یک دایرکتوری که شامل فایل هایی با این فرمت ها است دریافت کند. در مسیر /etc/kara فایل کانفیگ mrbench.conf وجود دارد که اطلاعات سرورهای هیولا و کانتینر های در آن موجود است و از این کانفیگ ها برای اتصال از طریق ssh و تغییر فایل های ring , swift استفاده می شود و در آخر پس از تغییرات در فایل های ring و swift برای اینکه تنظیمات جدید اعمال شود کانتینر های هیولا را restart می‌کند.

> **نکته: در اولین بار اجرای ابزار در صورتی که برای user ذکر شده در فایل کانفیگ ssh_key تعریف نشده باشد آنها را ایجاد و در سرور های هویلا کپی می‌کند.فقط درصورتی که ابزار فوق موفق به ساخت و کپی کلید ssh نشد دستورات زیر را به ازای تمام سرور های هویلا اجرا کنید.**
>
> ```
> su user
> ssh-keygen (make sure new key just for new user and his .ssh directory)
> ssh-copy-id -p <port> <user>@ip
> ```

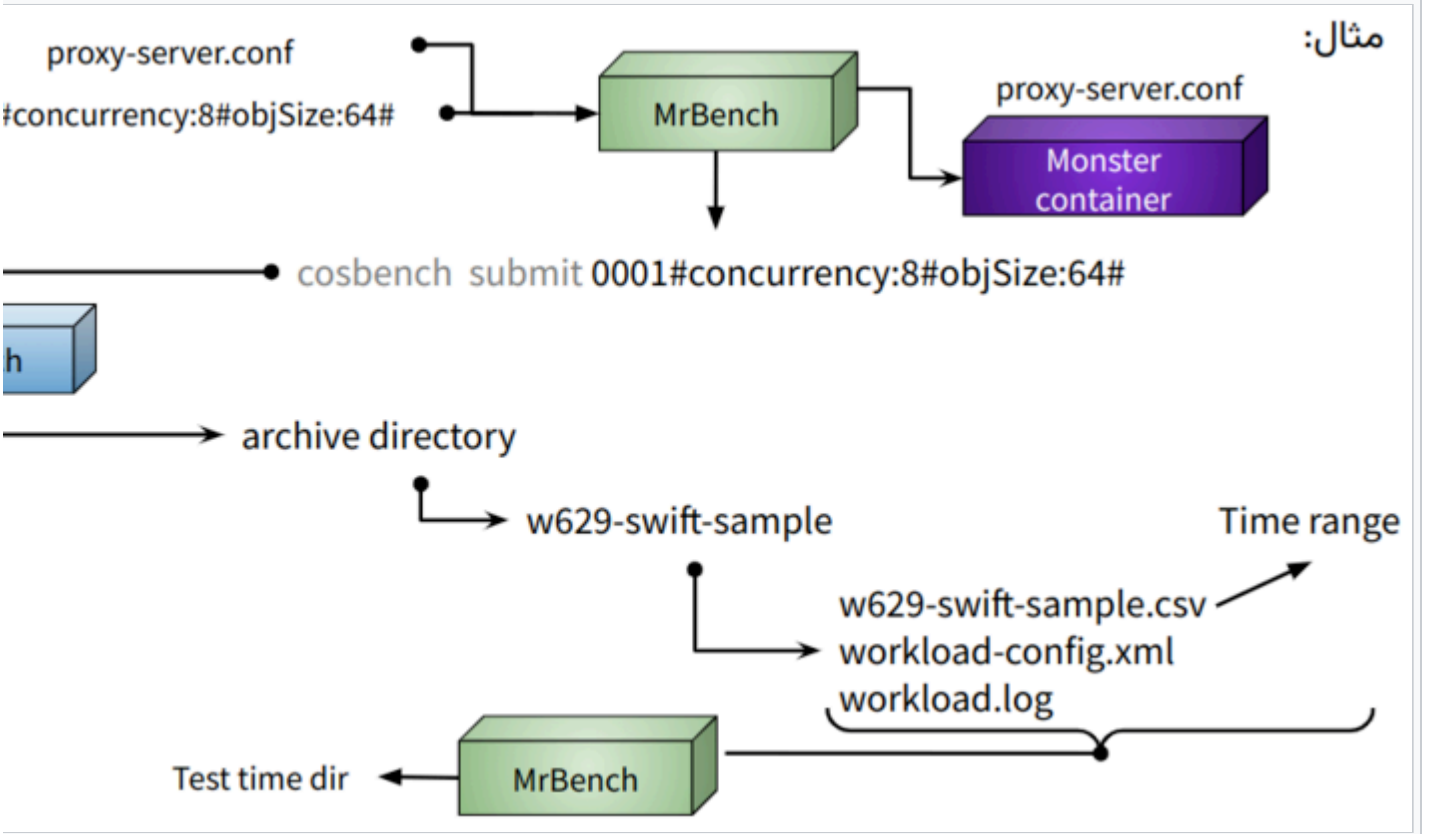
۱-۴-۱- شرح فایل کانفیگ و اجرای هر بخش: در فایل کانفیگ این ابزار فقط نیاز به ذکر نام کانتینر های هیولا و اطلاعات ssh سرور های میزبان آن است.

```
swift:

   r1z1s1:    # monster container name
         ssh_user: user    # user in host
         ip_swift: 0.0.0.0    # host ip
         ssh_port: 22    # host port

   r2z2s2:
         ssh_user: user
         ip_swift: 0.0.0.0
         ssh_port: 22
log:
   level: info  # values = debug - info - warning - error - critical
```



روند کار mrbench

روند کار این ابزار:



مثال از کارکرد mrbench

۱-۴-۲- روش استفاده از ابزار:

آرگومان های ورودی:

i- فایل تمپلیت workload ورودی به همراه آدرس کامل آن

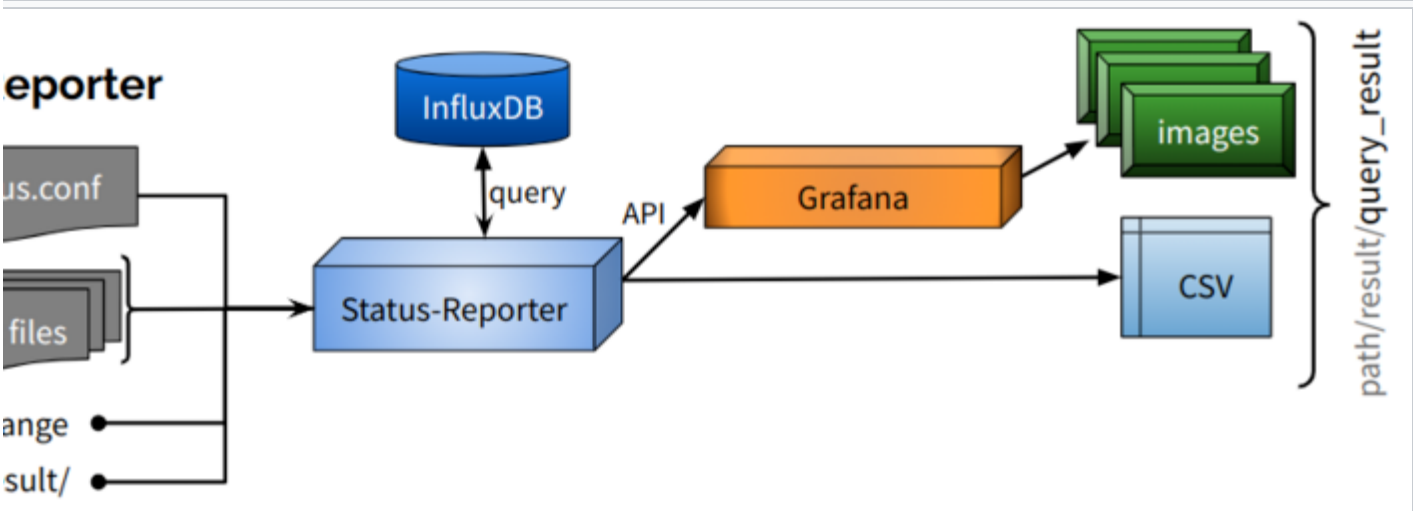
o- مسیر خروجی

cr- دایرکتوری که شامل کانفیگ های swift و ring می‌باشد

```
python3 mrbench.py -i <path to test config file> -o <output dir> -cr <path to config_ring dir>
```

۱-۵- ابزار status_reporter

این ابزار وظیفه ارسال query به influxdb را دارد و بر اساس measurement یا متریک های مورد نیاز جواب را دریافت کرده و به دو صورت فایل csv و فایل تصویر گراف نمایش می‌دهد. این ابزار در دایرکتوری status_reporter و فایل های کانفیگ آن در مسیر `/etc/kara/` در فایل status_reporter.conf و فایل های متریک آن در مسیر اصلی خود ابزار در دایرکتوری metrics قرار دارد. این ابزار برای ساخت تصاویر از نرم افزار grafana استفاده می‌کند به گونه ای که یک داشبورد گرافانا با فرمت json. که از قبل export شده است را در دایرکتوری jsons این ابزار قرار می‌دهیم و ابزار با استفاده از API گرافانا داشبورد را import می‌کند و تصاویر پنل های آن را به ازای هر سرور هیولا به صورت مجزا ذخیره می‌کند پس از این کار یک نمونه تصویر تجمیع شده از تمام تصاویر هر داشبور به ازای هر سرور نیز ایجاد می‌کند که در مراحل بعدی در اسناد کاتب استفاده می‌شوند.



روند کار status-reporter

۱-۵-۱- شرح فایل کانفیگ و اجرای هر بخش:

فایل کانفیگ از ۳ بخش اصلی تشکیل شده است که هر کدام در ادامه توضیح داده شده اند:

۱-۵-۱-۱- بخش اطلاعات دیتابیس و گرافانا: شما می توانید این بخش از کانفیگ را به ازای هر یک از سرور های mc تکرار کنید. در این قسمت ابزار به گونه ای عمل می‌کند که پس از دریافت اطلاعات دیتابیس و سرور mc و نرم افزار گرافانا, تصاویر و csv هایی از بازه های زمانی داده شده به آن را دریافت می‌کند. برای دریافت گزارش در قالب عکس نیاز است ابتدا از داشبورد مورد نظر در گرافانا یک فایل json استخراج شود و در دایرکتوری jsons ابزار ذخیره شود پس از آن با استفاده از API داشبورد های موقت در گرافانا import شده و تصاویری از پنل های درون آن با استفاده از پلاگین image renderer خود گرافانا به ازای هر host استخراج می‌شود و یک یا چند نمونه تجمیع شده از عکس

ها نیز با توجه به نیاز کاربر تولید می‌شود و همزمان با این فرآیند تولید عکس نتایج query هایی که مستقیما به دیتابیس ارسال می‌شوند نیز به صورت فایل های csv برای هر سرور هیولا ساخته شده و در دایرکتوری هایی با نام زمان گزارش ذخیره می‌شوند.

```yaml
influxdbs:
   MC:
      grafana_dashboards:
         remove_dashboards: True    # remove temporary dashboard of images
         time_variable: 10s    # time variable in dashboards
         dashboards_name:
               - Performance_Overview
               - Partial_Monitoring
               #- custom
         custom_panels:  # needs to select custom dashbord in dashboards section
               - network
               - cpu
         report_images:
           # panels_per_row: 2
           # panels_per_column: 3
           # max_panels: 9      # panels_per_row ingnored if set max_panels
           # panel_width: 800   # size in pixel
           # panel_height: 400  # size in pixel
      grafana_api_key: "add your API-key here"
      grafana_port: 3000
      grafana_ip: 0.0.0.0 # ip of grafana host
      influx_port: 8086
      influx_ip: 0.0.0.0 # ip of influxdb host
      databases:
         opentsdb:
            hosts:
               mc:
                  - "mc1"
                  - "mc2"
               monster:
                  - "r2z2s2-controller"
                  - "r1z1s1-controller"
```

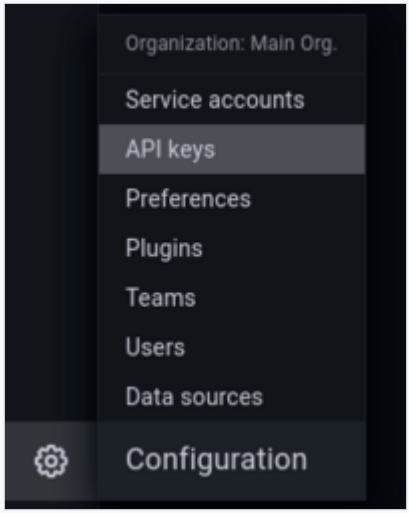
> **نکته: در داشبورد هایی که توسط کاربر ساخته شده اند و به این ابزار داده می‌شوند باید حتما از متغیر هایی با این نام ها استفاده شود تا فرآیند استاندارد سازی داشبورد ها به درستی انجام شود.**
>
> **نمایش و انتخاب سرور ها = hostIs (هاست ایز)**
>
> **تایم فریم = timeVariable**

شرح بخش کانفیگ ذکر شده:

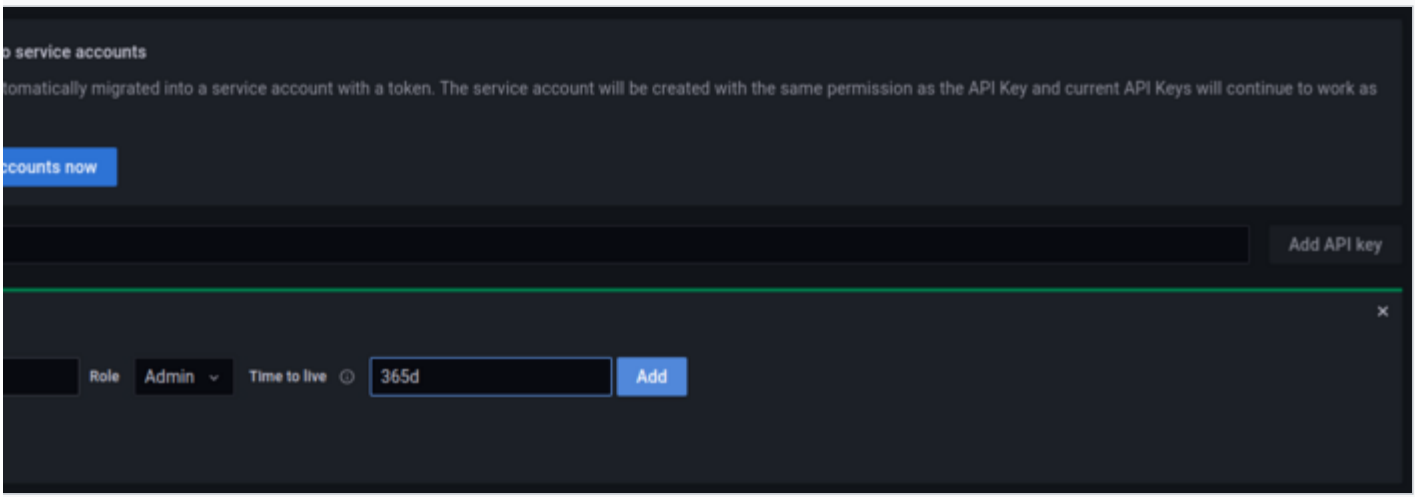
- **grafana_dashboards:**

- **remove_dashboards:** برای حذف کردن یا نکردن داشبورد موقت import شده به گرافانا است.
- **time_variable:** متغیر بازه های زمانی در گراف پنل ها است که مقدار آن باید در داشبورد ها تعریف شده باشد.
- **dashboards_name:** لیستی از اسامی داشبورد های موجود در دایرکتوری jsons. اگر این داشبورد ها از قبل در گرافانا وجود داشته باشند به انتهای اسامی آنها یک شماره اضافه شده و سپس import می‌شوند.
- **custom_panels:** این ابزار این قابلیت را نیز دارد که داشبورد هایی شخصی سازی شده تولید کند به این صورت که اگر یک فایل json داشبورد خالی از پنل استخراج کنید و نام آن را به custom تغییر دهید می‌تواند فایل های json استخراج شده از دیگر پنل ها را که در همان مسیر jsons قرار دارد به داشبورد خالی اضافه کند و عکس را از آن ها دریافت کند.
- **report_images:** در ابتدای این بخش گفته شد که پس از دریافت تصاویر برای هر داشبورد و سرور هیولا یک یا چند عکس تجمیع شده مشابه داشبورد از آنها ساخته می‌شود در این قسمت از فایل کانفیگ می‌توان ابعاد عکس تجمیع شده و تعداد کل تصاویر پنل درون آن و تعداد پنل های در سطر و ستون آن را مشخص کرد.

- **grafana_api_key:** برای ارسال و دریافت اطلاعات به گرافانا نیاز است از API آن استفاده شود برای این کار می‌توان مشابه تصاویر زیر یک API-key در گرافانا ساخت. پس از ایجاد API-key با رول admin و نام و تاریخ اعتبار دلخواه کد نمایش داده شده را در کانفیگ status_reporter قرار دهید.



مرحله اول



مرحله دوم

- **grafana_port** و **grafana_ip:** در این دو بخش باید ip , port سرور میزبان گرافانا ذکر شود.
- **influx_port** و **influx_ip:** در این دو بخش باید ip , port سرور میزبان influxdb ذکر شود.
- **databases:** در این بخش نیز لیستی از دیتابیس های موجود و سرور های هویلا در گروه بندی های مختلف موجود در هر دیتابیس قرار دارد. می‌توان این بخش را به ازای هر دیتابیس متفاوت تکرار کرد.

۱-۵-۱-۲- بخش اطلاعات فایل های متریک:

بخش دوم کانفیگ مربوط به فایل های measurement یا metric است در این قسمت عملیات ریاضی هر فایل و آدرس آن ذکر شده و امکان تغییر آنها در سوئیچ های ورودی یا آرگومان برنامه نیز وجود دارد. درون هر فایل لیستی از متریک های ارسال شده توسط netdata و ذخیره شده در influxdb وجود دارد که امکان کامنت کردن متریک های غیر ضروری نیز برای آنها فراهم شده.

فرمت نامگذاری متریک ها در فایل های موجود باید به این شکل باشد: **netdata.n.n.n**

این ابزار امکان تکمیل کردن اسم متریک ها را با استفاده از regex نیز دارد برای این کار کافی است اسم متریک مورد نظر را در فرمت های مثال زده شده بنویسید.

```
/netdata.system.cpu.*/
```

```
/netdata.disk_ops.sd[abc].reads/
```

بخش ذکر شده درون فایل کانفیگ:

```yaml
metrics:

  sum:
    path: ./../status_reporter/metrics/sum_metric_list.txt
  mean:
    path: ./../status_reporter/metrics/mean_metric_list.txt
  max:
    path: ./../status_reporter/metrics/max_metric_list.txt
  min:
    path: ./../status_reporter/metrics/min_metric_list.txt
```

۱-۵-۱-۳- بخش اطلاعات زمانی و خروجی:

در این قسمت امکان تعریف margin های زمانی از اول و آخر بازه ریپورت به منظور دقیقتر بودن خروجی وجود دارد و واحد آن به ثانیه است.

در قسمت آخر بازه زمانی گزارش قرار دارد که شامل زمان شروع و پایان گزارش است و فرمت آن tehran timestamp بوده و در صورتی که زمان دقیق و مشخصی برای بازه گزارش وجود نداشته باشد این امکان وجود دارد که از زمان حال حاضر اجرای برنامه تا یک بازه زمانی ماقبل آن نیز گزارش گرفته شود مشابه این فرمت ها:

now-2h , now (از زمان حال حاضر منهای ۲ ساعت قبل برای زمان شروع و زمان حال حاضر برای پایان)

now-2d , now (از زمان حال حاضر منهای ۲ روز قبل برای زمان شروع و زمان حال حاضر برای پایان)

```yaml
time:
  start_time_sum: 10  # increase your report start time
  end_time_subtract: 10  # decrease your report end time
  time_range: 2024-09-08 12:00:00,2024-09-08 12:20:00 # can take two format "now-nh,now-nh" or timestamp
"Y-M-D h-m-s,Y-M-D h-m-s"

output_path: /path/to/results
```

۱-۵-۲- روش های استفاده از ابزار:

در صورتی که نیاز می‌توان این موارد را در آرگومان ورودی به ابزار داد در غیر این صورت از فایل کانفیگ فراخوانی می‌شوند.

آرگومان های ورودی:

m- لیستی از فایل های متریک: path to metric file 1,path to metric file 2,path to metric file 3, ...

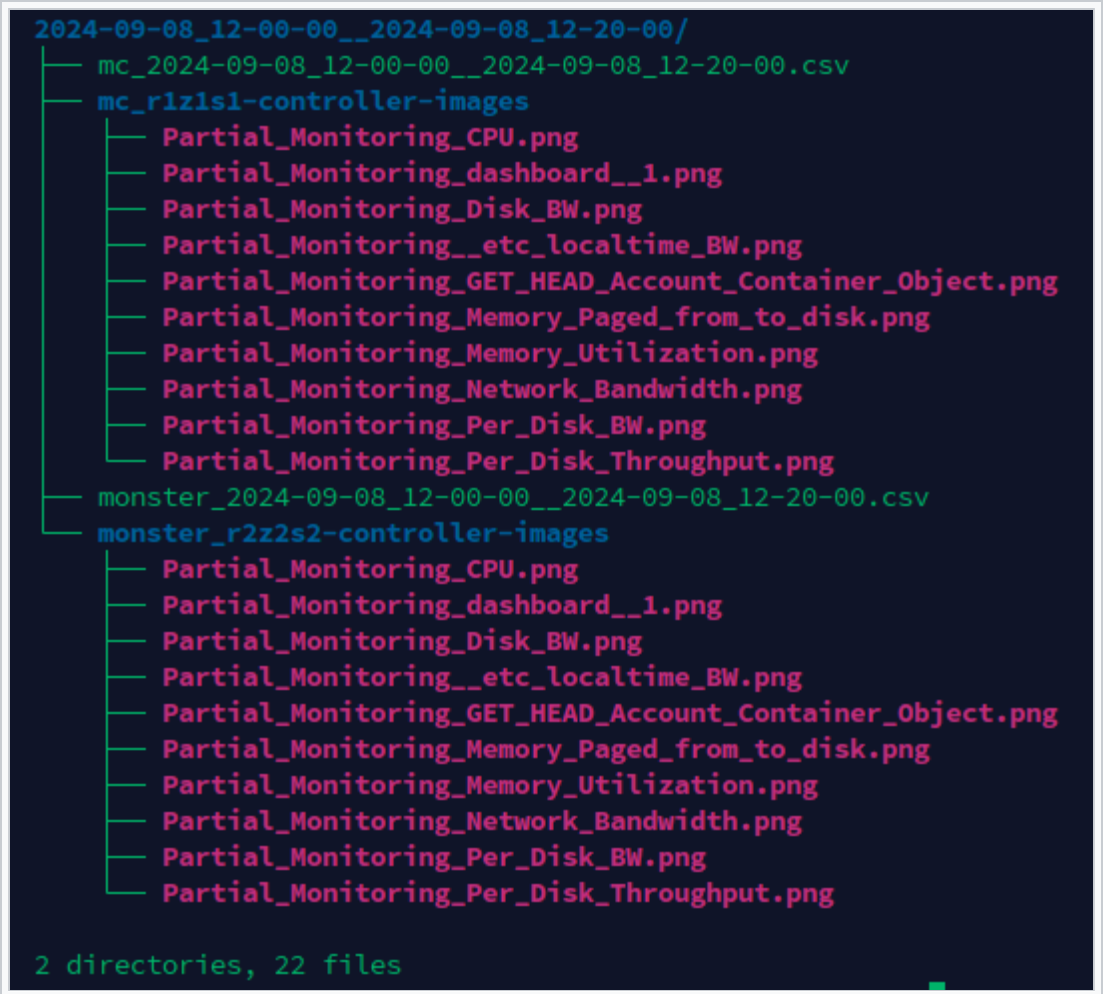
t- بازه زمانی تست 'start time,end time'

o- مسیر خروجی

img-- عملیات ساخت عکس ها

```
python3 status_reporter.py --img
```

در آخر نتایج در دایرکتوری داده شده به برنامه در دایرکتوری به نام بازه زمانی گزارش که در کانفیگ داده شده به صورت گراف هایی برای هر سرور هیولا و یک فایل csv برای هر گروه از سرور ها ذخیره می شود که اسم آن نیز بازه زمانی گزارش است مشابه تصویر پایین.



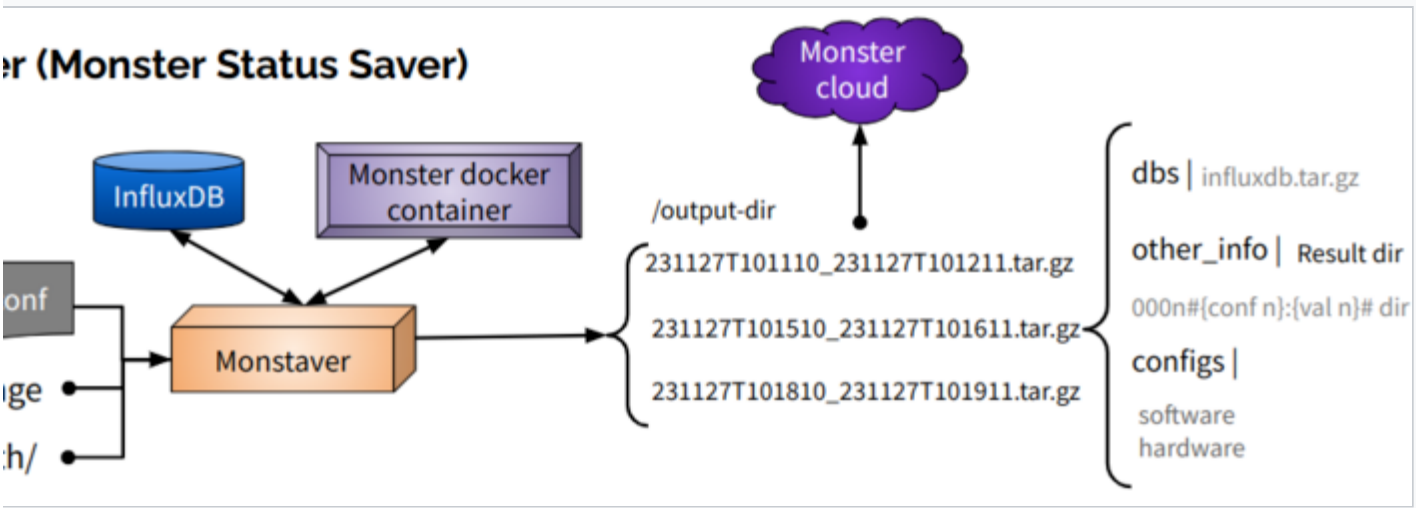
ساختار خروجی status_reporter

۶-۱- ابزار monstaver

این ابزار وظیفه دارد یک بازه زمانی دلخواه را از کاربر دریافت کند و در آن بازه زمانی از پایگاه داده influxdb یک نسخه پشتیبانی ایجاد کند و در مرحله بعدی کانفیگ های swift را که شامل ring و object و container و account هستند را نیز از هیولا دریافت به دایرکتوری نهایی محل بکاپ منتقل کند از دیگر ویژگی های ابزار این است که در کنار این کار می‌تواند نتایج تست و گزارش که توسط دیگر ابزار ها تولید می‌شود را نیز همراه با بکاپ و کانفیگ ها قرار داده و آنها را پس از دسته بندی برای ذخیره سازی و جابجایی راحت تر فشرده سازی کند و در صورت نیاز امکان restore کردن بکاپ های گرفته شده در influxdb برای آن نیز نظر گرفته شده است. از دیگر ویژگی های این ابزار می توان به آپلود فایل فشرده شده نهایی بکاپ در کلاستر هیولا به عنوان مثال zdrive اشاره کرد. فایل کانفیگ برنامه در مسیر `/etc/kara/monstaver.conf` قرار دارد.

> **نکته: در اولین بار اجرای ابزار در صورتی که برای user ذکر شده در فایل کانفیگ ssh_key تعریف نشده باشد آنها را ایجاد و در سرور های هویلا و mc کپی می‌کند. فقط درصورتی که ابزار فوق موفق به ساخت و کپی کلید ssh نشد دستورات زیر را برای تمام سرور های هویلا و mc اجرا کنید.**

```
su user
ssh-keygen (make sure new key just for new user and his .ssh directory)
ssh-copy-id -p <port> <user>@ip
```



روند کار monstaver

۱-۶-۱- شرح فایل کانفیگ و اجرای هر بخش:

کانفیگ این ابزار از دو بخش اصلی تشکیل شده. بخش اول برای بکاپگیری و بخش دوم برای بازیابی است. بخش بکاپگیری خود از سه قسمت تشکیل شده که به تفکیک هر کدام در ادامه توضیح داده شده اند:

۱-۶-۱-۱- بخش بکاپ (قسمت default):

در این بخش اطلاعات پایه فرآیند بکاپگیری قرار دارد که هر کدام در ذیل توضیح داده شده اند:

- **time , time_margin:** بازه زمانی بکاپ و مارجین های زمانی برحسب ثانیه که به زمان شروع افزوده و از زمان پایان بکاپ کسر می‌شوند تا بازه زمانی دقیقتری در بکاپ داشته باشیم.
- **input_path**: لیستی از مسیر های دلخواه که نیاز هستند در فایل نهایی وجود داشته باشند مانند مسیر خروجی تست و گزارش.
- **backup_output:** محل خروجی برنامه در سرور اجرا کننده کارا.
- **upload_to_monster:** اطلاعات کلاستر هیولا برای آپلود فایل بکاپ در آن. در صورت نیاز تداشتن به این بخش گزینه آپلود را false کنید نیاز به کامنت کردن آن نیست. موارد زیر مجموعه این بخش همراه با مثال:
  - **token_url:** "https://api.zdrive.ir/auth/v1.0/"
  - **public_url:** "https://api.zdrive.ir/v1/AUTH_user"
  - **username:** "user:user"
- **backup-options:** در این بخش موارد انتخابی برای قرار گرفتن در فایل بکاپ نهایی وجود دارد که شامل اطلاعات سخت افزاری و نرم افزاری سرور هیولا و اطلاعات کانتینر هیولا و کانفیگ های swift و ring است.

```
default:

   # time,margin: start,end
   time: 2024-09-1 15:00:00,2024-09-01 15:02:00 # can take two format "now-nh/d,now-nh/d" and "now-nh/d-now" or timestamp "Y-M-D h-m-s,Y-M-D h-m-s"
   time_margin: 10,10

   # some dir inside my local server
   input_paths:
         - /path/to/custom dir

   # output of all part in my local server
   backup_output: /tmp/influxdb-backup
```

```
   # monster storage info for upload backup
   upload_to_monster:
      upload: False
      token_url: # add your user token_url here
      public_url: # add ypur public_url here
      username: # add your username here
      password: # add your password here
      cont_name: kara # container name in monster

   # make backup from hardware/software/swift
   backup-options:
      hardware_backup: True
      software_backup: True
      swift_backup: True
```

۱-۶-۱-۲- بخش بکاپ (قسمت swift):

در این قسمت اطلاعات سرورهای هیولا و نام کانتینر ها وجود دارد و از این کانفیگ ها برای اتصال با ssh و دریافت فایل های swift , ring و کانفیگ های نرم افزاری و سخت افزاری سرور های هیولا استفاده می شود.

```
swift:

   r1z1s1:    # monster container name
         ip: 0.0.0.0    # host ip
         ssh_port: 22    # host port
         ssh_user: user    # user in host
   r2z2s2:
         ip: 0.0.0.0
         ssh_port: 22
         ssh_user: user
```

۱-۶-۱-۳- بخش بکاپ (قسمت db_sources):

در این قسمت اطلاعات mc که دارای کانتینر influxdb است به همراه mount point درون کانتینر که برای ذخیره سازی موقت بعضی از فایل ها استفاده می‌شود و اسامی دیتابیس های موجود در آن قرار دارد این بخش می‌تواند به ازای سرور های mc تکرار شود.

```
db_sources:

   MC:
      ip: 0.0.0.0
      ssh_port: 22
      ssh_user: user
      container_name: influxdb
      influx_volume: /var/lib/influxdb/KARA_BACKUP   # mount point inside influxdb container
      databases:    # list of databases
            - opentsdb
```

۱-۶-۱-۴- بخش بازیابی:

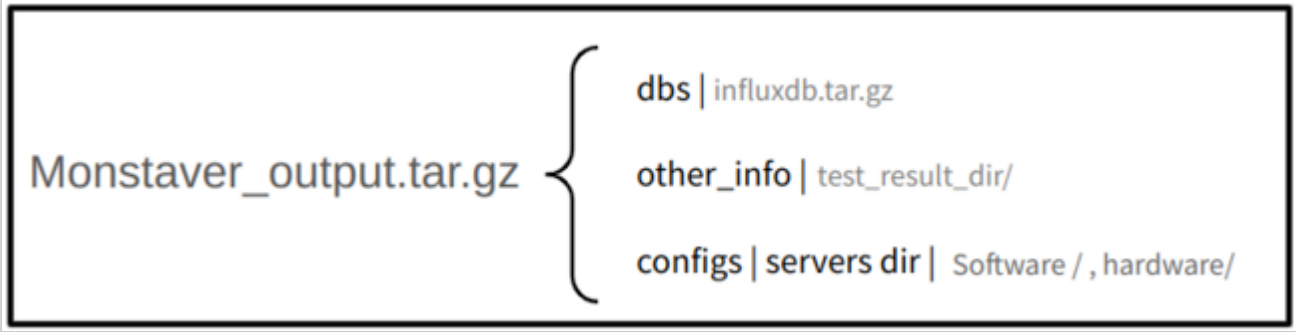
در این قسمت اطلاعات سرور mc دارای influxdb نوشته می شود. بیشتر کانفیگ ها مشابه بخش قبلی است فقط در قسمت databases از داخل فایل فشرده بکاپ اسم دیتابس مبدا استخراج می‌شود و مقدار prefix به آن اضافه شده و نام دیتابیس جدید که بکاپ روی آن بازگردانی خواهد شد ساخته می‌شود.

> **نکته: دیتابیس مبدا که در فایل فشرده قرار دارد باید حتما در influxdb وجود داشته باشد تا بتوان از روی آن داده ها بازگردانی شوند به دیتابیس جدید. این مورد از پیش نیاز های خود influx است و مربوط به معماری کارا نیست.**

```
influxdbs_restore:

   MC:
      ip: 0.0.0.0
      ssh_port: 22
      ssh_user: user
      container_name: influxdb
      influx_volume: /var/lib/influxdb/KARA_RESTORE   # mount point inside influxdb container
      databases:
         - prefix: "rst1_" # prefix of new database name
           location: /tmp/influxdb-backup/path/to/dbs/influxdb.tar.gz # backup file location
```

۱-۶-۲- خروجی برنامه:

فایل خروجی برنامه به صورت پیش فرض در مسیر /tmp/influxdb-backup/ قرار داشته و درون آن سه دایرکتوری اصلی وجود دارند: dbs برای بکاپ دیتابیس ها و other_info برای مسیر ها و فایل های دلخواه کاربر و configs برای اطلاعات سخت افزاری و نرم افزاری و swift که به ازای هر سرور هیولا وجود دارد.



خروجی monstaver

۱-۶-۳- روش استفاده از ابزار:

موارد ذکر شده در آرگومان های زیر در فایل کانفیگ نیز وجود دارند در صورت استفاده نکردن از آرگومان از فایل خوانده می‌شوند.

آرگومان های ورودی:

t- بازه زمانی بکاپ 'start time,end time'

i- مسیر یا فایل دلخواه کاربر برای قرار گرفتن در بکاپ

sw- بکاپ گرفتن از تنظیمات نرم افزاری سرور های هیولا

hw- بکاپ گرفتن از تنظیمات سخت افزاری سرور های هیولا

s- بکاپ گرفتن از کانفیگ های swift و ring

ib- بکاپ گرفتن از دیتابیس

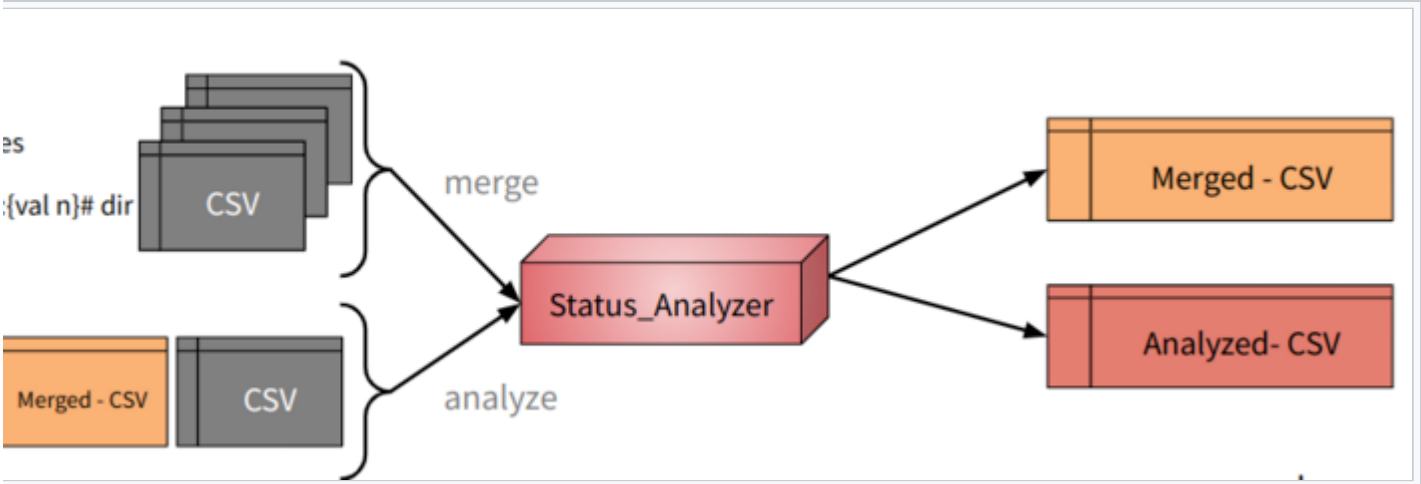
d- پاک کردن دایرکتوری خروجی نهایی ابزار و فقط ذخیره کردن خروجی در فرمت tar.gz

r- استفاده از ویژگی restore

```
Python3 monstaver.py
```

۱-۷- ابزار analyzer

این ابزار از در مجموع وظیفه انجام دو فرآیند را دارد. تحلیل یا آنالیز نتایج گزارش های دریافت شده از هیولا که به صورت csv هستند و ساخت تصاویر و گراف هایی آنالیز شده و همچنین تجمیع نتایج گزارش ها به یک فایل csv جامع برای همه گزارش های دریافت شده در بازه های زمانی مختلف.



روند کار status-analyzer

تجمیع (merge): این بخش برای فرآیند تجمیع نیاز به بیشتر از دو عدد فایل csv در ورودی دارد که می‌تواد آن ها را به صورت لیستی از فایل ها ("...,csv1,csv2,csv3") و یا دایرکتوری شامل چندین فایل ("/csv_dir/*") دریافت و پس از تجمیع آن ها نتیجه را در فایلی به نام merged.csv در مسیر داده شده توسط کاربر ذخیره می کند.

تحلیل(analyze):

این بخش وظیفه تجزیه و تحلیل یک csv را بر عهده دارد و برای انجام این کار به فایل کانفیگ `analyzer.conf` در مسیر `/etc/kara/` نیاز داریم . از جمله عملیات های ریاضی که برای اعمال روی ستون های csv در این ابزار تعریف شده می‌توان به می‌تواند میانگین (avg) و جمع (sum) و ضرب (mul) و تقسیم (div) بین ستون های مورد نظر را نام برد و برای تمام سطر های فایل ورودی دو مقدار میانگیم و تجمیع ذخیره کند و در آخر یک csv جدید با ستون هایی جدید که نتایج تحلیل در آنها وجود دارد ایجاد می‌کند. روند کار بخش تحلیل به این صورت است که در ورودی یک فایل csv دریافت می‌کند و پس از آن از درون فایل کانفیگ ابزار موارد مورد نیاز کاربر را دریافت و بر روی فایل اعمال می‌کند و در انتها خروجی این برنامه یک csv جدید است که نام آن از ترکیب اسم فایل اولیه و کلمه analyzed بوده است. برای درک بهتر کار کرد ابزار به بخش کانفیگ مراجعه کنید.

۱-۷-۱- شرح فایل کانفیگ و اجرای هر بخش:

```
transformation:

    csv:
        columns:
            cpu.io: # new_column_name
                operation: avg   # values = avg - sum - mul - div
                selected_columns:
                    - mean_system.cpu.idle
                    - mean_system.cpu.iowait

            cpu.info: # new_column_name
                operation: sum   # values = avg - sum - mul - div
                selected_columns:
                    - sum_system.cpu.system
                    - sum_system.cpu.user
```

```
            total_cpu_Usage:
                operation: mul   # values = avg - sum - mul - div
                selected_columns:    # values = name of columns or a number  # just select two columns
                    - sum_system.cpu.system
                    - 96

            perUser_cpu_Usage:
                operation: div   # values = avg - sum - mul - div
                selected_columns:    # just select two columns
                    - sum_system.cpu.user
                    - sum_system.cpu.system
        rows:
            - sum
            - avg
```

csv -۱-۱-۷-۱: در این قسمت بخش های مربوط به تحلیل و ساخت csv وجود دارد.

- **columns:** این قسمت نمایانگر تحلیل بر روی ستون های فایل ورودی است و عملیات های ریاضی مورد نیاز و ستون های انتخابی و نام ستون های جدید در زیر مجموعه این بخش قرار دارند. (در صورتی که نیاز به تحلیل بر روی ستون ها ندارید می‌توان کل این قسمت را حذف یا کامنت کرد)
  - در قسمت بعدی برای هر دسته از عملیات های تحلیل یک نام دلخواه در نظر گرفته شده است که برای اسم ستون جدید نیز در نظر گرفته می‌شود به عنوان مثال (cpu.io)
    - **operation:** عملیات های ریاضی موجود در این بخش نوشته می‌شوند.
    - **selected_columns:** در این قسمت لیستی از ستون های انتخابی برای تحلیل نوشته می‌شود و عملیات ریاضی روی آنها اعمال می‌شود برای عملیات تقسیم حتما باید دو ستون انتخاب شوند و برای عملیات ضرب حتما دو ستون یا بیشتر برای موارد دیگر در صورتی که کم تر از دو مورد باشند مقدار همان ستون چاپ می‌شود.

> **نکته: برای عملیات ضرب در این بخش می‌توان از اعداد نیز استفاده کرد تا در ستون انتخابی ضرب شوند.**

- **rows:** در این بخش تحلیل میانگین و تجمیع برای همه ستون ها انجام شده و در دو سطر جدید با نام های AVG , SUM ذخیره می‌شوند.

بخش ساخت گراف تحلیل شده در فایل کانفیگ مشابه مثال ذکر شده در پایین بوده و در ادامه به شرح آن می‌پردازیم.

```
    graph:
        g1:    # group name
            filter:
                Host_name: # name of target column
                    - r1z1s1-controller # name of target value
                    - r2z2s2-controller
            selected_columns:
                - sum_system.cpu.user: sum_system.cpu.system   # X_axis: Y_axis
                - mean_system.cpu.idle: mean_system.cpu.iowait
        g2:
            filter:
                Host_name:
                    - r2z2s2-controller
            selected_columns:
                - mean_system.cpu.idle: mean_system.cpu.iowait
```

graph -۲-۱-۷-۱: در این قسمت بخش های مربوط به ساخت گراف بر حسب ستون های فایل csv ورودی قرار دارد.

- در قسمت اول این بخش یک نام دلخواه برای هر گروه از گراف ها قرار دارد که این نام در اسم تصویر نهایی نیز وجود دارد.
  - **filter:** در این بخش یک فرآیند فیلتر کردن وجود دارد برای جلوگیری از تداخل اسم ستون ها و مقداری آنها در زمان ساخت گراف. این بخش به این صورت کار می‌کند که در خط بعدی آن باید اسم ستونی که تفکیک کننده بخش مورد نیاز ما از دیگر بخش ها است نوشته شوند به عنوان مثال: ما نیاز

به گرافی برای متریک های دو سرور r1z1s1 و r2z2s2 داریم در این بخش نام سرورها در ستون Host_name نوشته شده پس از این ستون برای تفکیک موارید مورد نیاز استفاده می‌کنیم.

- **selected_columns:** در این بخش لیستی از ستون ها نوشته می‌شود که نیاز است گراف برای آنها ساخته شود. در هر خط از این لیست اسم دو ستون مقابل یک دیگر نوشته می‌شود با علامت ":" میان آنها ستون اول می‌شود محور x و ستون دوم محور y و برای آنها بر حسب یکدیگر گرافی ساخته می‌شود.

۱-۷-۲- روش استفاده از ابزار:

آرگومان های ورودی:

-M: نمایانگر عملیات merge یا تجمیع

-o: مسیر خروجی برنامه

-sc: لیستی از csv های مورد نیاز برای تجمیع (csv1,csv2,csv3,...) یا دایرکتوری شامل چندین فایل (/dir/*)

-A: نمایانگر عملیات analyze یا تحلیل

-c: اسم و مسیر فایل csv نیازمند تحلیل

-k: نمایانگر اینکه ستون های اولیه فایل تحلیل شده در آن باقی بمانند یا حذف شوند و فقط ستون های جدید در آن باشند.

-G: نمایانگر عملیات ساخت گراف

```
python3 analyzer.py -M -o <output> -sc <csv1,csv2,csvn> -A -c <csv for analyze> -k -G
python3 analyzer.py -M -o <output> -sc <csv_dir/*> -A -c <csv for analyze> -k -G
```

۱-۸- ابزار report_recorder

۱-۸-۱- پیشنیاز های ابزار:

قبل از اجرای این ابزار چه در سناریو manager و چه به صورت مجزا نیاز است که فایل کانفیگ pywikibot که ابزار رابط بین report_recorder و کاتب است کانفیگ شود. برای این کار به مسیر کتابخانه های پایتون رفته `usr/local/lib/python <n> /dist-/packages/report_recorder_bot/pywikibot/families` این مسیر می‌تواند بسته به ورژن پایتون متفاوت باشد. پس از آن فایل kateb_family.py را ویرایش کرده و دامنه و نام کاربری و و رمز عبور وب سرور کاتب در صورت وجود داشتن در این بخش ذکر کنید مشابه مورد ذیل:

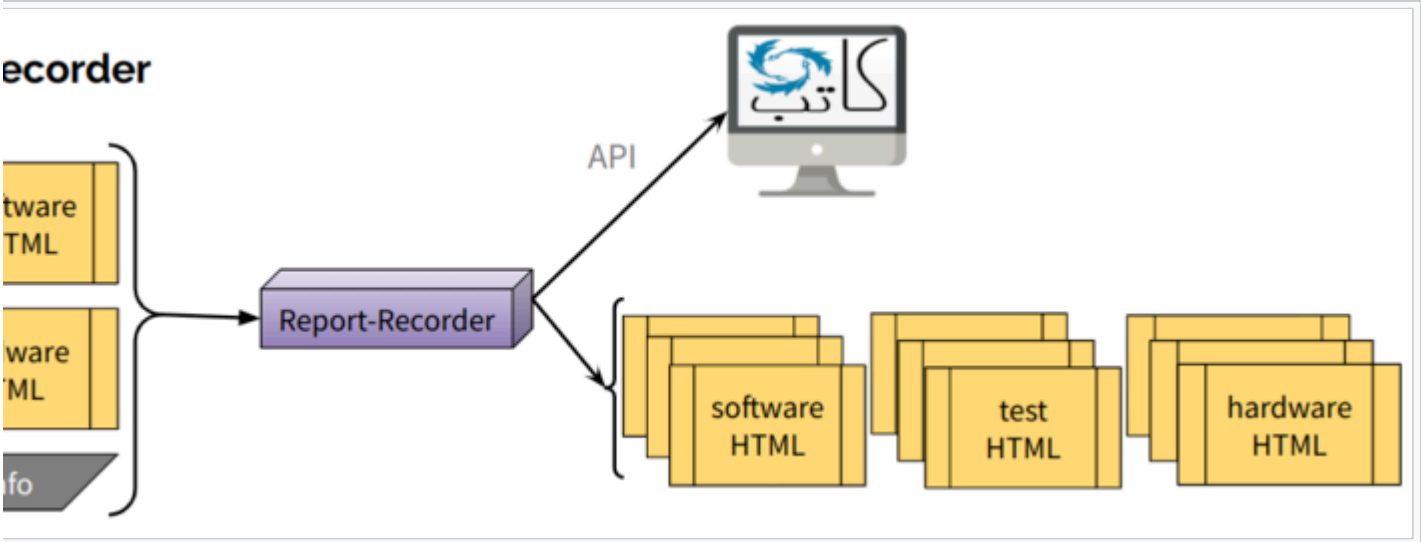
```
    name = 'kateb'
    langs = {
        'fa': 'user:pass@kateb.burna.ir',  # uesr:pass of web server@kateb_domain  Attention!! do not type '#' in user or pass here, replace it with Ascii code "%23"
    }
```

> **نکته: در صورتی که در نام کاربری یا رمز وب سرور از علامت "#" استفاده شده بود کد جدول Ascii آن یعنی ۲۳ را به فرمت 23% بنویسید.**

۱-۸-۲- روند کار ابزار:

این ابزار وظیفه ایجاد مستندات از گزارش های دریافت شده برای تست های هیولا را دارد و آنها را در فرمت html ذخیره و سپس در سامانه کاتب بارگذاری می‌کند. روند کار به این صورت است که در ورودی یک قالب html برای مشخصات سخت افزاری سرور های هیولا و یک قالب برای مشخصات نرم افزاری هیولا دریافت کرده و درون آنها در

مکان هایی مشخص و با استفاده از placeholder هایی مشخص اطلاعات سرور ها را جایگذاری می‌کند و چندین نوع قالب از آنها در خروجی می‌سازد. برای اطلاعات خود تست ها نیز فایل cvs تجمیع شده و دایرکتوری خروجی تست ها را دریافت و یکسری مستندات html برای مشخصات و کانفیگ های خود تست می‌سازد و سپس آنها همراه با گراف های هر تست در کاتب آپلود می‌کند.



روند کار report-recorder

۱-۸-۳- قالب های html اولیه سخت افزار و نرم افزار:

این دو فایل در دایرکتوری input_templates وجود دارند مشابه قالب های ذکر شده از placeholder های {sw_config} برای درج مشخصات نرم افزاری و {hw_config} برای درج مشخصات سخت افزاری استفاده می‌شود. موارد نوشته شده مقابل این placeholder ها المان هایی هستند که نمایانگر کانفیگ یا اطلاعاتی مورد نیاز برای چاپ در این بخش از سند هستند. پیشنهاد می‌شود از همین موارد پیشفرض استفاده شود یا در صورت نیاز المان ها حذف یا جابجا شوند و یا متن های موجود در فایل ها تغییر کنند ولی مورد جدیدی به جز موارد نوشته شده اضافه نگردد.

```
<h2>مشخصات سخت افزاری</h2>
<p>در این بخش مشخصات سخت افزاری هر کدام از سرور ها به تفصیل مورد بررسی قرار می گیرد. این مشخصات بر اساس تقسیم بندی شده اند قسمت های مختلف سخت افزار سرورها.</p>

<h3> سرور </h3>
{hw_config}:hardware,brand

<h3>پردازنده</h3>
{hw_config}:hardware,cpu
<a href="./subpages/{title}--CPU"> مشخصات سخت افزار پردازنده  </a>

<h3>حافظه اصلی</h3>
{hw_config}:hardware,memory
<a href="./subpages/{title}--Memory"> مشخصات سخت افزار حافظه </a>

<h3>شبکه</h3>
{hw_config}:hardware,net
<a href="./subpages/{title}--Network"> مشخصات سخت افزار شبکه </a>

<h3>دیسک و کنترولر</h3>
{hw_config}:hardware,disk
<a href="./subpages/{title}--Disk"> مشخصات سخت افزار دیسک و کنترولر </a>

<h3>PCI</h3>
<p> برای مشاهده اطلاعات به سند <a href="./subpages/{title}--PCI">  اطلاعات pci </a> مراجعه کنید.</p>

<h3>مادربورد</h3>
{hw_config}:hardware,motherboard

<h3>تنظیمات RAID</h3>
<p>این مشخصات بسته به برند سرور فیزیکی با دستورات متفاوتی گزارش می شود.</p>
<p>استفاده می شود ssacli از دستور HP به طور مثال برای سرور های.</p>
```

```
<p>در این سند مشخصات نرم افزاری کلاستر آمده است</p>

<h2>معماری کلاستر</h2>

<h3>نسخه نرم افزارهای استفاده شده در هر سرور</h3>
{sw_config}:software_version

<h2>مشخصات نرم افزاری</h2>
<p>در این بخش مشخصات نرم افزاری هر کدام از سرورها مورد بررسی قرار می گیرد. این مشخصات به دو بخش کلی
تنظیمات مربوط به هیولا و تنظیمات مربوط به سیستم تقسیم بندی می شود.</p>

<h3>تنظیمات مربوط به هیولا</h3>
<p>این تنظیمات شامل سرویس ها، وضعیت رینگ ها و فایل های کانفیگ هر سرویس می شود.</p>

<h4>وضعیت سرویس های هیولا</h4>
<h5>سرویس های اصلی</h5>
{sw_config}:swift_status,main

<h5>سرویس های آبجکت</h5>
{sw_config}:swift_status,object

<h5>سرویس های اکانت</h5>
{sw_config}:swift_status,account

<h5>سرویس های کانتینر</h5>
{sw_config}:swift_status,container

<h4>وضعیت رینگ ها</h4>
<h5>رینگ های آبجکت</h5>
{sw_config}:rings,object

<h5>رینگ های اکانت</h5>
{sw_config}:rings,account

<h5>رینگ های کانتینر</h5>
{sw_config}:rings,container

<h4>فایل های کانفیگ سرورها</h4>
<h5>پروکسی سرور ها</h5>
{sw_config}:server_confs,proxy

<h5>آبجکت سرور ها</h5>
{sw_config}:server_confs,object

<h5>اکانت سرور ها</h5>
{sw_config}:server_confs,account

<h5>کانتینر سرور ها</h5>
{sw_config}:server_confs,container

<h3>تنظیمات مربوط به سیستم</h3>
<p>در این قسمت تنظیمات مربوط به کرنل و سرویس های داخل داکر به ازای هر سرور قرار می گیرد.</p>

<h4>وضعیت سرویس ها</h4>
<h5>systemctl list-units</h5>
{sw_config}:systemctl

<h5>lsof | wc -l</h5>
{sw_config}:lsof

<h5>lsmod</h5>
{sw_config}:lsmod

<h4>تنظیمات sysctl</h4>
{sw_config}:sysctl
```

۱-۸-۴- تحلیل اسناد:

سند یا فایل html ساخته شده برای مشخصات سخت افزاری و نرم افزاری دارای تحلیل یا آنالیز بوده و تفاوت ها و نقاط مشترک سرور های هیولا چه از نظر سخت افزار و چه نرم افزار در آن ثبت شده است تا فرآیند عیب یابی سریع تر انجام شود. این تحلیل انجام شده برای مشخصات نرم افزاری مقداری پیشرفته تر بوده و می‌توان در زمان نمایش اشتراکات و تفاوت های سرور ها مواردی که مهم نیستند و اولویتی در تحلیل ندارند را با استفاده از فایل هایی که این موارد غیر ضروری در آنها ذکر شده اند فیلتر کرد.

روش کار این بخش به این صورت که در مسیر خروجی نهایی و ذخیره html ها یک دایرکتوری با نام unimportant_conf باید ایجاد کنید و درون آن فایل های برای هر بخش از سند که نیاز به تحلیل بیشتر دارد بسازید و داده های غیر ضروری را درون آنها قرار دهید مانند مثال ذیل :

ما در دایرکتوری و مسیر ذکر شده فایلی با نام sysctl-Unimportant_conf.txt ایجاد کرده این و درون آن می‌توان هر مقداری که اهمیتی در سند ندارد قرار دهیم مانند "kernel.random.boot_id" با این کار در بخش sysctl سند نرم افزار این مقدار قرار داد شده که از این دستور استخراج می‌شود در بخش غیر مهم جدول sysyctl در سند نوشته می‌شود و باقی موارد در بخش مهم جدول.

۱-۸-۵- روش استفاده و شرح فایل کانفیگ ابزار:

قبل از اجرای نرم افزار نیاز است که اطلاعات کاربری کاتب خود را در فایل user-config.py موجود در دایرکتوری report-recorder یا manager بسته به محل فراخوانی ابزار وارد نمایید. پس از باز کردن این فایل نام کاربری خود را در قسمت user name قرار دهید و پس از اولین اجرای نرم افزار و به هنگام آپلود اطلاعات در کاتب فقط در دفعه اول password خود را وارد می‌کنید رمز شما در یک فایل md5 ذخیره می‌شود. فایل کانفیگ برنامه نیز در مسیر /etc/kara/ و با اسم خود برنامه وجود دارد.

۱-۸-۵-۱- بخش اول فایل کانفیگ شامل اطلاعات اولیه و اصلی به شرح ذیل بوده:

```yaml
cluster_name: kara
scenario_name: test

tests_info:
  merged: /path/to/kara/results/analyzed/merged.csv
  merged_info: /path/to/kara/results/analyzed/merged_info.csv
  images_path: /path/to/results/
  test_tags:
    - "تست"
    - "کارایی"
    - "هیولا"

hw_sw_info:
  configs_dir: /tmp/influxdb-backup/path/to/backup dir
  software_template: /path/to/kara/report_recorder/input_templates/software.html
  software_tags:
    - "گزارشها"
    - "سیستم عامل"
    - "هیولا"
  hardware_template: /path/to/kara/report_recorder/input_templates/hardware.html
  hardware_tags:
    - "گزارشها"
    - "سخت افزار"
    - "هیولا"

kateb_list_page: "name of page" # the page include list of titles
output_path: /path/to/kara/report_recorder/output_htmls/
```

- **cluster_name:** اسم کلاستر هیولا که گزارش ها برای آن ایجاد می‌شوند.
- **scenario_name:** اسم سناریو یا نوع سندی که برای آن کلاستر ساخته خواهد شد.
- **merged:** فایل تجمیع شده همه تست های یک دسته.
- **tests_info:** بخش مربوط به اطلاعات اولیه و مورد نیاز برای اسناد تست های گرفته شده از هیولا.
  - **merged:** فایل تجمیع شده تمام csv های یک دسته تست گرفته شده.
  - **merged_info:** فایل تکمیلی اطلاعات هیولا در زمان هر تست به صورت تجمیع شده.
  - **این دو فایل ذکر شده در بخش بالا هم زمان با هم ایجاد می‌شوند و مکمل یکدیگر هستند.**
  - **images_path**: مسیر دایرکتوری والد همه تست ها که شامل تصاویر و گراف تست ها است.
  - **test_tags:** رده های اسناد نتایج تست.

- **hw_sw_info:** بخش مربوط به اطلاعات اولیه و مورد نیاز برای اسناد سخت افزار و نرم افزار هیولا و سرور آن.
  - **configs_dir:** اطلاعات موجود در اسناد از یکی از بکاپ های گرفته شده توسط ابزار monstaver استخراج می‌شود برای این منظور باید آدرس دایرکتوری بکاپ که در حالت فشرده نیست را در این بخش وارد کرد.
  - **software_template و hardware_template:** نام فایل های html ورودی و آدرس آنها.
  - **software_tags و hardware_template:** رده های مورد استفاده در این نوع اسناد به تفکیک.
- **kateb_list_page:** پس از ساخت اسناد و آپلود آنها در کاتب این ابزار می‌تواند برای راحتی در مدیریت و پیدا کردن آنها لیستی از آنها را در یکی از صفحات موجود در کاتب قرار دهد برای این کار در این بخش نام صفحه دلخواه خود را که از قبل در کاتب موجود است را بنویسید.
- **output_path:** مسیر خروجی و ذخیره همه html های ساخته شده توسط ابزار که دارای دو دایرکتوری برای صفحات فرعی و تصاویر موجود در اسناد است.

۱-۸-۵-۲- بخش دوم فایل کانفیگ شامل اطلاعات مورد نیاز برای سند تست های هیولا بوده و دسته بندی اسناد و شکستند سند به چندین زیر مجموعه در این قسمت مشخص می‌شود.

```
classification:
  categories:
    LAT:
      comment: "در این تست‌ها میزان تاخیر کلاستر بررسی می‌شود."
      filter:
        workload.concurrency:
          - 1
      categories:
         P1:
           filter:
             workload.proxy:
               - 1
         P2:
           filter:
             workload.proxy:
               - 2
         P3:
           filter:
             workload.proxy:
               - 3
    BW:
      filter:
        workload.concurrency:
          - 72
          - 144
    TP:
      filter:
        workload.concurrency:
          - 256
          - 512

  maxTestsPerPage: 8 #Specifies the maximum number of tests that can be displayed on a single web page
when the 'auto-divider=True', default: 8
  autoDivider: False #default: True
  comment: "این متن در ابتدای سند نمایش داده می‌شود."
```

- **Classification**: این بخش نحوه شکسته شدن اسناد و بخش‌بندی آن‌ها را بر اساس پارامترهای تست مشخص می‌کند که شامل موارد زیر است:
  - **categories**: نام بخش‌های مختلف و نحوه فیلتر کردن آن‌ها در این بخش مشخص می‌شود. در مثال بالا بخش تمامی تست‌ها به سه دسته اصلی LAT-BW-TP تقسیم می‌شوند و دسته LAT نیز خود به سه بخش P1-P2-P3 تقسیم شده است.
  - **comment**: هر بخش می‌تواند یک کامنت داشته باشد. کامنت متنی است که در ابتدای آن دسته نمایش داده می‌شود.

- **filter**: در اینجا فیلترهای مربوط به هر دسته مشخص می‌شود. در مثال بالا تمامی تست‌ها با workload.concurrency=1 در بخش LAT و از بین این تست‌ها، آن‌هایی که مقدار workload.proxy آن‌ها برابر با 1 است در زیر بخش P1 قرار می‌گیرند. همچنین تست‌هایی که در آن‌ها مقدار workload.concurrency برابر با ۷۲ یا ۱۴۴ است در بخش BW قرار می‌گیرند.
- **maxTestsPerPage**: حداکثر تعداد تست یا گزارشی که در یک صفحه html یا کاتب قرار می‌گیرد.
- **autoDivider**: در صورتی که مقدار این پارامتر False باشد، شکستن اسناد تنها توسط دسته‌بندی‌ها و فیلترهای مشخص شده انجام می‌شود و مقدار maxTestsPerPage در نظر گرفته نمی‌شود.
- **comment**: متن قرار گرفته شده در این قسمت در ابتدای سند نمایش داده می‌شود.

۱-۸-۶- روش استفاده از ابزار:

در صورتی که فایل کانفیگ تنظیم شده باشد نیازی به استفاده از اکثر آرگومان های ذکر شده نیست و فقط با دستور ذکر شده در انتها می‌توان ابزار را اجرا کرد.

آرگومان های ورودی:

-st: مسیر و فایل قالب سند نرم افزار

-ht: مسیر و فایل قالب سند سخت افزار

-tp: عملیات ساخت فایل های html برای تست های گرفته شده

-o: مسیر خروجی تمام فایل های ساخته شده

-cn: اسم کلاستر

-sn: اسم سناریو

-cd: مسیر دایرکتوری بکاپ گرفته شده در ابزار monstaver

-m: فایل csv تجمیع شده (merged)

-mi: فایل تکمیلی csv تجمیع شده (merged_info)

-kl: اسم یک سند کاتب برای ایجاد لیست صفحات

-img: مسیر دایرکتوری والد همه تست ها و تصاویر

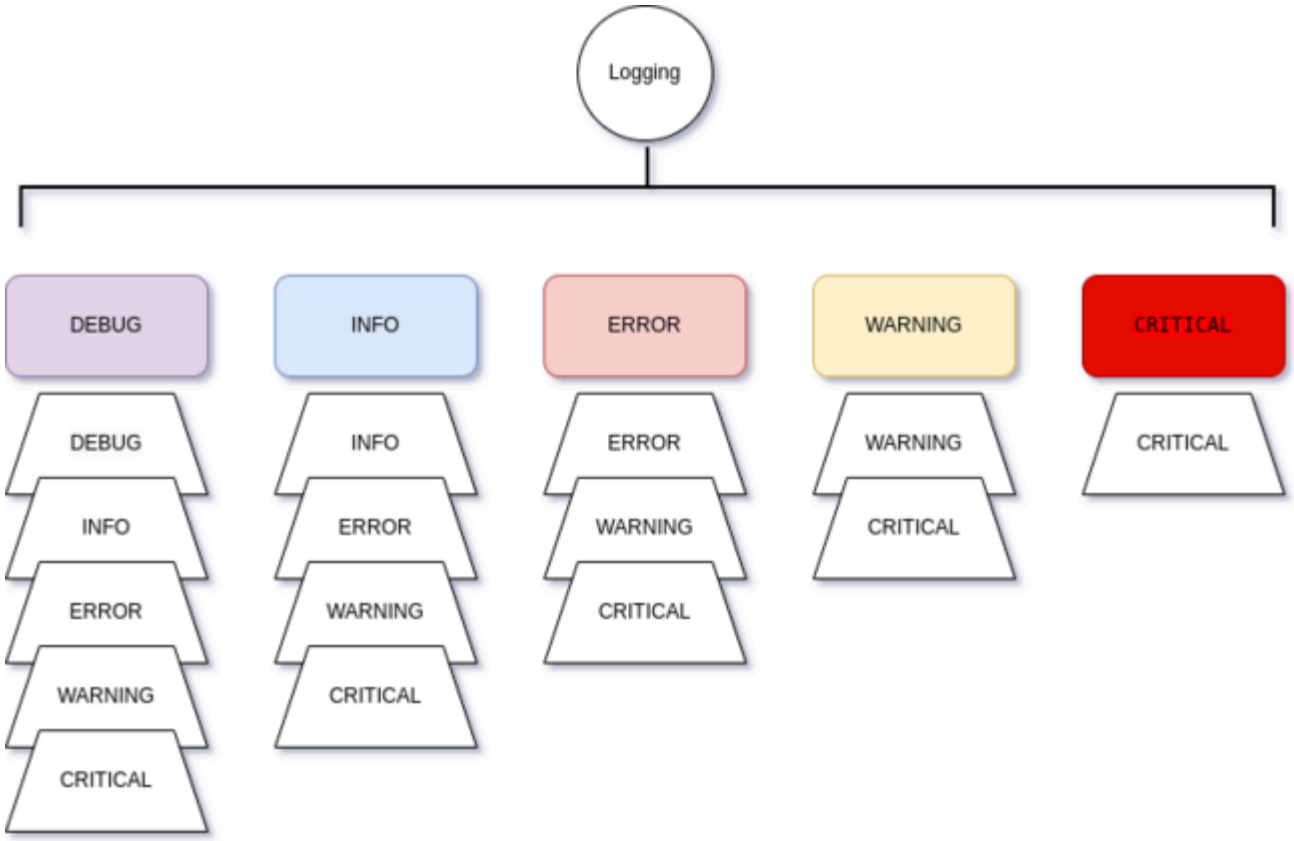
-H: عملیات ساخت html فقط ساخت و ذخیره فایل ها به صورت لوکال

-U: عملیات آپلود در کاتب (اگر در کنار -H استفاده شود همین سند های جدید آپلود می‌شوند اگر مجزا استفاده شود در مسیر خروجی اگر فایلی از قبل بود آپلود می‌شود)

```
python3 report_recorder.py -H -U
```

۲- لاگ ابزار ها

مسیر و فایل لاگ تمام ابزار ها یکسان بوده و برابر است با `/var/log/kara/all.log` و در ابزار های پیشرفته تر و دارای فایل کانفیگ در انتهای فایل این امکان وجود دارد که سطح نمایش لاگ ها را بین مقادیر debug - info - warning - error - critical تغییر داد. برای ابزار های ساده تر و بدون فایل کانفیگ سطح نمایش لاگ debug است . در تصویر ذیل سطح های نمایش لاگ شرح داده شده اند.



مشارکت‌کنندگان: ابوالفضل شفیعی, محمد تقوا, رضوان رضایی, زینب نخعی

-