---

title: LeadingAgile Code Analysis Tool Docker Images

1. Gather Analysis Docker Images

1. Container Usage

Run a container of the image. Using the run_gather.sh script will be easiest:

1.1. Run Gather Tools

1.2. Gather Tools Available

The analysis/gather docker image has several tools that can be used. This is like the 'plumbing' tools in git where the analysis/gather−cli image is like git porcelain tools. These can be used separately from the gather−cli, just be aware of their requirements and sequencing (some depend on the outputs of others).

1.2.1. gather

The main metrics gathering tool. Creates the metrics/ folders in the output folders and the data collected from the static analysis tools. Also creates the linguist/ data for the collection of langauages used and the frequency/ data from analysing the git commit history.

```
usage: metrics.gather -r THE_REPO -o OUTPUT_FOLDER [--run-name
    RUN_NAME] [--start-date START_DATE] [-t NUM_STEPS] [-h]

Gather static code analysis statistics for a repository covering a
    number of time increments.

Required:
  -r THE_REPO, --repo THE_REPO
                        A git repository URI or directory path
  -o OUTPUT_FOLDER, --output-folder OUTPUT_FOLDER
                        The desired output folder for reports to be
                            written.

Optional:
  --run-name RUN_NAME   A name for the gather run. Defaults to start
      date (if given) or current datetime.
  --start-date START_DATE
                        The starting date to gather statistics from in
                            YYYY-MM-DD format.
  -t NUM_STEPS, --steps NUM_STEPS
                        The number of weeks (steps) to analyze.
  -h, --help            Show this help message and exit
```

1.2.2. statistics

Creates the statistics data from the analysis of the data collected by gather. The output is in the statistics/ folder.

```
usage: metrics.statistics --run-folder RUN_FOLDER [--team-config
    TEAM_CONFIG] [-h]

Generate statistics from a run of the gather tool. All output is
    written to the run folder.

Required:
  --run-folder RUN_FOLDER
                        The run folder where the generated plots will
                            be written.

Optional:
  --team-config TEAM_CONFIG
                        A configuration file in YAML format to denote
                            which repository files/folders belong to a
                            team.
  -h, --help            Show this help message and exit
```

1.2.3. plotting

Plots the statistics data and the data from the frequency and linguist analyses.

```
usage: metrics.plotting --run-folder RUN_FOLDER [-h]

Generate plots for statistics from a run of the gather tool. All
    output is written to the run folder.

Required:
  --run-folder RUN_FOLDER
                        The run folder where the generated plots will
                            be written.

Optional:
  -h, --help            Show this help message and exit
```

1.2.4. answers

Answers the GQM questions using the data from the metrics collection and statistics data.

```
usage: metrics.answers [-h] {repo,team} ...

Generate answers for assesment data from a run of the gather tool. All
    output is written to the run folder.

Common Options:
  -h, --help    Show this help message and exit

Answers Commands:
  {repo,team}   Answer commands. Use '<comand> --help' for help
      specific to the command
    repo        Create a summary answer for a repository that includes
        all languages.
    team        Create a summary answer for all the repostories of a
        team. An answer for each repo must have already been created
        (see metric.answers repo)
```

repo

Create a summary answer for a repository that includes all languages.

```
usage: metrics.answers repo --run-folder RUN_FOLDER [--team-config
    TEAM_CONFIG] [-h]

Required:
  --run-folder RUN_FOLDER
                        The run folder for the run data to be analyzed.

Optional:
  --team-config TEAM_CONFIG
                        A configuration file in YAML format to denote
                            which repository files/folders belong to a
                            team.
  -h, --help            Show this help message and exit
```

team

Create a summary answer for all the repostories of a team. An answer for each repo must have already been created (see metric.answers repo).

```
usage: metrics.answers team --output-folder OUTPUT_FOLDER --run-name
    RUN_NAME [--team-config TEAM_CONFIG] [-h]

Required:
  --output-folder OUTPUT_FOLDER
                        The output folder where the metric gather runs
                            are stored.
  --run-name RUN_NAME   The name of the run to summarize for the team.
      All repositories for the team must have a run with this name to
      be included in the summary. The run selected must have already
      had a repo
                        answer generated.

Optional:
  --team-config TEAM_CONFIG
                        A configuration file in YAML format to denote
                            which repository files/folders belong to a
                            team.
  -h, --help            Show this help message and exit
```

1.2.5. utility

Provides several utility functions:

```
usage: metrics.utility [-h] {frequency-subset} ...

Miscellaneous utilities

Common Options:
  -h, --help          Show this help message and exit

Utility Commands:
  {frequency-subset}  Utility commands. Use '<comand> --help' for help
      specific to the command
    frequency-subset  Create a subset configuration for each of the
        repositories based on the file commit frequency.
```

frequency-subset

Generate a team config for a subset of the files in the repositories based on the git frequency data.

```
usage: metrics.utility frequency-subset --output-folder OUTPUT_FOLDER
    --team-name TEAM_NAME --run-name RUN_NAME [--team-config
    TEAM_CONFIG] [--percent PERCENT] [--max-count MAX_COUNT]
    [--min-count MIN_COUNT]
                                        [-h]

Required:
  --output-folder OUTPUT_FOLDER
```

```
                        The output folder where the metric gather runs
                            are stored.
  --team-name TEAM_NAME
                        The name for the team or subset of the
                            repostories.
  --run-name RUN_NAME   The name of the run to summarize for the team.
      All repositories for the team must have a run with this name to
      be included in the summary. The run selected must have already
      had a repo
                        answer generated.
  --team-config TEAM_CONFIG
                        A configuration file in YAML format to denote
                            which repository files/folders belong to a
                            team.
  --percent PERCENT     The percent of the most frequently modified
      files to include.
  --max-count MAX_COUNT
                        The maximum number of the files to include.
  --min-count MIN_COUNT
                        The minimum number of the files to include.

Optional:
  -h, --help            Show this help message and exit
```

1.3. Running the Tools

The analysis/gather docker image is used to run the tools. Use docker run to start a container from the image. The first argument after the image name is the name of the tools to run (see the list above). The remaining arguments are specific to the particluar tool. To see the list of arguments for a tool, pass --help.

To get help for the statistics tool use a command like this:

```
docker run -it -rm
    leadingagilestudios.azurecr.io/analysis/gather:0.1.2 statistics
    --help
```

To run the gather tool to collect metrics for the Studios-private-test-data--line-chart-typescript repostitory, use a command like this:

```
docker run -it -rm \
    -v ~/metrics_data/:/opt/repos/ \
    -v ~/report_folder/:/opt/output/ \
    leadingagilestudios.azurecr.io/analysis/gather:0.1.2 \
    gather \
    -r /opt/repos/Studios-private-test-data--line-chart-typescript/ \
    -o /opt/output/from_docker \
    -t 52 \
    --run-name RUN_01
```

In each case, the tool name follows the image name and the tool parameters follow the tool name.

1.4. Run a Gather Dev Container

Open a linux (debian:buster) terminal at the code−analysis code folder.

Use your github username and PAT

Mount ~/metrics_data/ in the container as /opt/metrics_data (in case you need a host output folder)

Mount the folder with you already-cloned repositories as /opt/sample_code/ in the container.

Your host SSH keys from ~/.ssh will be registered inside the container so you ssh access should work.

```
docker run -it --rm \
       -e GITHUB_USR=<github_username> -e GITHUB_PAT=<github_PAT> \
       -v ~/projects/la/sample_code/:/opt/sample_code/ \
       -v ~/metrics_data/:/opt/metrics_data/ \
        leadingagilestudios.azurecr.io/analysis/gather-dev:0.4
```

NOTE: The code in the container is a COPY of the state of the code-analysis code at the time the Image was created. It is NOT a git repository. If you make changes, they will disappear when you exit the container. Containers are ephemeral!.

If you want to actually work on code while in the container, you have a couple of choices.

- Use the container as an experiment, let the code go when you exit, and redo it in the real world.
- git clone the code-repository repo into the container, work, commit, and push. Note: again, when you exit, this clone vanishes.
- Mount your host code folder into the container using a −v host_folder:container_folder option. This has risks. You are futzing with your host (MacOS) folder while in a Linux container. **Beware the Shenanigans!**

1.5. Using the Gather CLI Container

See the doc in the gather-cli Folder

2. Configuration for Using Images

2.1. SSH Config Updates

When running the Docker container from a Mac, you will need to update the .ssh/config for your SSH key with the following:

```
IgnoreUnknown UseKeychain
```

2.1.1. Example

```
Host *
  IgnoreUnknown UseKeychain
  AddKeysToAgent yes
  UseKeychain yes
  IdentityFile ~/.ssh/id_leadingagile
```

3. Pulling images from LeadingAgile Registry

- Be sure you have Pull permissions to the registry (see the admins)
- install the Azure CLI brew install azure−cli
- Log in to Azure using your LeadingAgile credentials az login
- Log in to the Azure Container Registry for Studios az acr login −−name leadingagilestudios
- Pull the image docker pull leadingagilestudios.azurecr.io/analysis/gather:0.4

The images that are available currently are:

- leadingagilestudios.azurecr.io/analysis/gather:0.4
- leadingagilestudios.azurecr.io/analysis/gather−dev:0.4
- leadingagilestudios.azurecr.io/analysis/gather−cli:0.4

Note that the versions for these will change as we deploy updated versions of the tools.

4. Building Images Locally

If you really need to build an image locally:

4.1. Build for Docker images locally

- Start in code−analysis base repo folder
- Run the build images script: ./tools/docker/build_gather_dockers.sh 0.4

This will build the gather, gather−dev, and gather−cli docker images.