System Requirements

Release 1.23, December 2019

GitHub, Inc.
88 Colin P Kelly Jr Street
San Francisco, CA 94107
United States

Copyright © 2020, GitHub, Inc. All rights reserved.

GitHub Automated Code Scanning release 1.23
Document published January 07, 2020

Contents

Introduction

What is GitHub Automated Code Scanning?

GitHub Automated Code Scanning combines deep semantic code search with data science insights to help developers ship secure code. At the heart of GitHub Automated Code Scanning is the semantic code analysis and scanning engine CodeQL. Built on top of CodeQL is LGTM, a web application that comes in two forms:

LGTM.com—a publicly available, free to use, cloud-based service that continuously analyzes over 135,000 open source projects worked on by over 700,000 developers, and containing data for more than 39 million commits.

LGTM Enterprise—the on-premises twin of LGTM.com. LGTM Enterprise:

- Provides dashboards, graphs, and other user interface elements to show automated code scanning results in the browser
- Includes machine-learning-based knowledge to prioritize and highlight potential issues, helping developers optimize their work
- Allows developers to run multiple language analysis in parallel
- Integrates with source code management, issue tracking, and code review platforms, and provides a REST API to support additional capabilities
- Includes an administration interface for managing user accounts and configuring and maintaining the system

Note
"LGTM" is the most commonly posted comment on code review platforms (such as GitHub). It means: "looks good to me."

About this document

This document describes the minimum system requirements for a deployment of GitHub's on-premises automated code scanning application: LGTM Enterprise. For more detailed information about the services mentioned in this document, see the administrator help.

Related documentation

- *LGTM Enterprise System Architecture* (PDF)
- *LGTM Enterprise Installation and Upgrade Guide* (PDF)
- LGTM Enterprise administrator help

  To access this, click **Admin help** at the top of the administration pages in LGTM Enterprise, or browse to help.semmle.com/lgtm-enterprise/admin.

Languages supported

LGTM Enterprise supports analysis of code written using the following languages:

- C and C++
- C#
- Go (also known as Golang)
- Java
- JavaScript/TypeScript
- Python

For detailed information about which versions of these languages are supported, and details of built-in support for frameworks, see Languages and compilers.

System requirements

In some systems the control pool consists of a single machine—referred to as the "controller." This also hosts the web and data aggregation services so that there is no separate web pool. At larger scales a web pool of additional machines can be added to increase capacity for web and data aggregation services, with the "controller" remaining on a separate machine.

Type of machine	Hardware	Software
Control pool machine[1]	Controller • 8 cores • 16 GB RAM • 1 TB free disk space[2] • Solid-state drive (SSD)[3]	*Operating system*—one of the following Linux distributions/versions: • CentOS 7 • Debian 9 • Red Hat Enterprise Linux 7.x (from 7.4) • Ubuntu 16.04 LTS Ubuntu 18.04 LTS
	Web pool[1] • 2 cores • 8 GB RAM • 50 GB free disk space • SSD recommended[3]	
Work pool machine *(running one worker daemon)*[1]	• 2 cores • 8 GB RAM minimum • 50 GB free disk space per worker daemon[2] • SSD recommended[3]	*Operating system*—Linux versions as above, or Windows 7 or later. Required on Windows: • The 64-bit version of the most recent release of OpenJDK 8, or Oracle JDK 8. Note: IBM Java, and other versions of Java are not supported. • Python 2.7 On Linux, the latest JDK 8 and Python 2.x are installed on the worker host machine by LGTM if they are not found.

[1] For more information about the control pool, web pool, and work pool, see the *LGTM Enterprise System Architecture* PDF, or the "System architecture overview" topic in the administrator help.

[2] You must size disk partitions appropriately for LGTM. For the control pool, disk usage in the `data_path` location (by default, `/var/lib/lgtm`) can grow unboundedly, so it should get the vast majority of whatever disk space is available. The free disk space requirement for other locations used by the control pool services amounts to less than 10 GB in total. For machines that host LGTM workers, the `temp_path` location (by default, `/var/cache/lgtm/workers`) must be at least big enough to hold the code from your largest repository several times over.

[3] We recommend that you use solid-state drives (SSDs) rather than hard disk drives (HDDs) for all of the machines, in both the control pool and the work pool. We strongly recommended that you use an SSD for the "controller" machine, as files are served from here to all other cluster machines.

Note

- During installation, upgrade and configuration processes, root access is required for each machine.
- Virtual machines (for example, using VMWare, Xen, or KVM) or physical machines are both suitable for the pools of machines. The terms "machines" and "virtual machines" are used interchangeably in the documentation.
- For installations on Amazon Web Services, using the supplied Amazon Machine Image (AMI), see the LGTM Enterprise Installation and Upgrade Guide for details of which Amazon EC2 instance type to use.

The computational power required for the work pool depends on the number of projects to analyze, their size, build times (for compiled languages such as Java), and the average commit frequency.

As this information is usually difficult to ascertain in advance, it makes sense to plan for an incremental deployment based on the following information:

- Minimum deployment size for LGTM Enterprise
- Hardware requirements at scale
- Incrementally adding capacity

Third-party software installed by LGTM

LGTM Enterprise uses the following third-party software. If these resources are not found on the relevant machines during installation of LGTM Enterprise, they are automatically installed.

- Java JRE 8 (if necessary, OpenJDK 8 will be installed)
- Python 2.7
- PostgreSQL 9.5 or later
- RabbitMQ Server
- Nginx
- Git
- Subversion
- Solr
- Minio

Solr and Minio are supplied within the LGTM distribution. If the other software listed above needs to be installed it is downloaded from an external repository. Except where the version number is shown above, the latest available version is downloaded. Any dependencies are also installed.

Access to third-party packages during installation

The simplest way to install LGTM Enterprise is to enable internet access and to run the installation script supplied with the application. This enables the installer to download and install the required packages from their official online package repositories using the standard tools available in your Linux distribution. If you're using Red Hat Enterprise or CentOS, this will require access to the EPEL (Extra Packages for Enterprise Linux) repository. Once LGTM has been deployed, internet access can be closed down.

If internet access is not available, you can do either of the following:

- Configure the machine so that it can access an internally hosted mirror that contains these package repositories and their dependencies. The installer will download and install the required packages from your internal mirror.

- *For installations on Red Hat or CentOS:* Extract the third-party dependencies pack before you run the installer. Details for doing this are included in the installation instructions.

Work pool hardware requirements

The machines in the work pool run one or more worker daemons that perform the "heavy lifting" involved in building and analyzing your projects. The number of worker daemons you need depends on the amount of work that needs to be processed and various aspects of your projects, such as the build commands that are required and the number and size of dependencies for each project. You can choose to run several workers on a well resourced machine, or just one or two workers on a less well resourced machine.

Important

When determining an initial setup, you need to consider the following questions:

- Are you going to be using automated code review (where LGTM runs analysis on each newly submitted pull request and then adds a comment with a link to the analysis results)? This requires more resource than if you are just running analysis on changes that have already been merged into the default branch of the repository.
- Do your projects have a large history of commits that you want LGTM Enterprise to analyze? Is it important to you to have analysis data for the commit history soon? LGTM performs historical analysis during times that the workers would otherwise be idle (for example, over weekends), so this does not necessarily require more workers, but if you want to speed up this process you should consider running additional general workers, for perhaps 60–90 days, until the system has computed all of the historical data. You can then scale down the number of workers.

Worker estimation

You can use the following calculations to give you a starting point for the number of worker daemons you are likely to need. During testing you can then scale up or down the number of worker daemons.

These calculations include an "LGTM analysis factor." This is a number between 3 and 5. Use 5 if workers run on a single thread. Use a number closer to 3 if workers are configured to use multiple threads. For details about configuring workers to use multiple threads, see Enabling parallel processing.

The examples below are for a single imaginary project (Project A). On a machine with 4 single-thread cores and 32 GB of RAM, Project A usually takes about 1.5 hours to build (outside of LGTM). We'll configure the LGTM workers so that they can use up to 4 cores in parallel and, as a result, we'll use an LGTM analysis factor of 3.5 in our calculations. On average, 5 pull requests are raised against this project every day and 15 commits are merged into the default branch per day.

With automated code review

```
number of workers = ( (build time in hours * LGTM analysis factor) *
((submitted pull requests + commits to the default branch) per day) ) /
24hrs
```

Using our Project A example, this is:

```
( (1.5 * 3.5) * (5 + 15) ) / 24 = 4.38
```

To analyze Project A, and use automated code review, you should initially configure 5 worker daemons. Each worker must be able to use up to 4 threads in parallel. If you run multiple worker daemons on a worker host, the host must have adequate resources to allow each worker to use up to 4 threads in parallel, and RAM should be similarly scaled up.

Without automated code review

```
number of workers = ( (build time in hours * LGTM analysis factor) *
(commits to the default branch per day) ) / 24hrs
```

Using our Project A example, this is:

```
( (1.5 * 3.5) * (15) ) / 24 = 3.28
```

To analyze Project A, without automated code review, you should allow 4 worker daemons (with each worker able to use 4 threads).

Disk space requirement

The machines you use to host the workers should have at least 50 GB of free disk space available per worker. We recommend you use host machines with solid-state drives (SSDs).

Work pool software requirements

The LGTM work pool must be set up with machines that include the right software and the correct environment to enable checkout and analysis of the codebases you want to analyze.

Checkout requirements

The essential requirements are:

1. **Appropriate client software** to allow the worker to check out code from the repository—that is, Git, Subversion or Team Foundation Version Control (TFVC). The LGTM Enterprise installer automatically installs Git and Subversion clients. For TFVC, you must install the Team Explorer Everywhere command-line client (TEE-CLC) on each worker host machine.

2. **Repository access** for all worker daemons. All of the worker daemons running on machines in the work pool need to be able to connect to all of the repository hosts you use. They also need read access to the code.

   > **Note**
   > All general and on-demand workers can be used to poll your repositories, to check whether new changes have been made to the code. This is true even for workers that will not be used to build and analyze code from a particular repository. So, typically, connections to all of your repository hosts occur from all of your worker host machines.

   Generally, workers use the authentication credentials you specify when you configure repository settings for an integration (see "Defining integrations with external systems" in the administrator help) to access the repositories. For Git-based repository providers, however, you have the option of configuring workers to connect over SSH, by setting the integration's **Checkout method** accordingly.

For more information, see "Configuring hosts to checkout code" in the LGTM Enterprise administrator help.

Analysis requirements

During the analysis of compiled languages, LGTM builds the code. This means that there must be at least one machine in the work pool that is configured with the environment needed to

build each codebase that you want to analyze. Interpreted languages often also depend on the presence of other software in the environment, and these must be made available too.

Core requirements

The core requirements listed below are the minimum required to analyze code written in each language. Your projects may have additional requirements. Compiled languages have further requirements (detailed in the following section).

Language	Essential
C/C++	Method of building the code (see next table).
C#	Method of building the code (see next table).
Go	Go toolchain (version 1.11 or later) If dependencies are managed using dep, install version 0.5.0 or later of dep. If dependencies are managed using Glide, install version 0.13.0 or later of Glide.
Java	Method of building the code (see next table).
JavaScript	No further requirements unless any TypeScript files are present. If the repository also includes TypeScript files, the TypeScript requirements must be met, or TypeScript analysis disabled.
Python	Access to the pip packaging management system. Installation of the packaging and virtualenv pip modules. Ability to download any packages that the Python codebases you intend to analyze depend on.
TypeScript	Node.js version 6.x or later.

Auto-deduction of build commands for compiled languages

For C/C++, C#, and Java code, LGTM Enterprise uses a series of heuristics to determine a suitable set of build commands, using an appropriate build system. This is referred to as "autobuild." For each language, the autobuild process chooses between a set commonly used build systems (listed in the following table), and runs commands for that system. If your organization uses one or more of these build systems, install them on the relevant worker host

machines. If some projects use other build systems, you also need to make these systems available. These projects will require a custom build configuration.

Language	Automated build configuration available for
C/C++	Linux workers: Autoconf, CMake, QMake, Meson, WAF, SCons, and Linux KBuild Windows workers: MSBuild
C#	MSBuild, .NET Core, Mono (Linux), and Visual Studio (Windows)
Java	Gradle, Maven and Ant

For more information, see "Configuring hosts to analyze code" in the administrator help.

Deployment

You can install LGTM Enterprise in a number of ways:

- By performing an interactive, or programmatic, installation on one or more clean virtual machines.
- Using a supplied Amazon Machine Image (AMI) to install a single-machine instance on Amazon Web Services.
- As a Dockerized deployment, in which LGTM Enterprise runs in a set of Docker containers.

This section of this guide deals with the first of these options. You can find out more about an AWS or Dockerized deployment in the LGTM Enterprise Installation and Upgrade Guide.

Operating systems

The machine(s) in the LGTM Enterprise control pool use Linux (see "System requirements" on page 7). GitHub provides packages in common formats (RPMs and Debian packages) for the LGTM Enterprise software, and an optional third-party dependencies RPM package that you can use if you need to run an offline installation on Red Hat or CentOS.

Machines in the work pool use Linux or Windows. These machines should be configured with an environment that will allow the LGTM worker daemons to check out and build the projects they will analyze. It's important to test that you can check out and build all of the relevant projects from these machines before you deploy LGTM. You use an RPM or Debian package to deploy LGTM to the work pool, and an MSI installer to deploy to Windows machines.

Minimum deployment

For an initial trial, a suitable cluster of machines is:

- Control pool: 1 virtual machine (VM)
- Work pool: 5–20 virtual machines

The hardware requirements defined in the system requirements are enough to support one worker per work pool VM. If you want to run more than one general worker per work pool VM, multiply the disk space, RAM and CPU cores accordingly.

> **Note**
> When you add projects with over 1 million lines of code to LGTM, you may need to increase the resources available on the work pool machines. More RAM and disk space may be required to build large projects. If the builds are parallelized, increasing the number of cores available on each machine may also be beneficial.

In addition to the hardware and software requirements above, each VM in the work pool must be set up with the build environment that is required to build the projects that will be analyzed. For example:

- For C# analysis the correct versions of the C# compiler must be installed.
- Python analysis requires:
  - A supported version of Python 3 (see Languages and compilers)
  - Python 2.7, if you are going to be analyzing 2.7 code, or on a Windows worker host
  - The pip package installer
  - The packaging and venv modules, and the virtualenv tool

  For more details, see the topic "Setting up a worker host for Python analysis" in the administrator help.

Hardware requirements at scale

LGTM Enterprise is designed to scale successfully from analyzing the work of a few hundred developers to analyzing the work of many thousands of developers. The resources required change according to the nature of the repositories analyzed, size of the projects, number of developers and commit frequency. We recommend that you start with a minimum deployment and scale this as required.

You can monitor the free capacity of the work pool using the Infrastructure administration page. If you need to add extra capacity, you should provision additional machines and register them as part of the work pool.

In rare cases where there is a particularly large amount of data, the services on the controller can be hosted on different machines.

Other bottlenecks are usually best addressed by changing the control pool from a single machine to a cluster. For example:

- Create a web pool of several machines to load balance the web application and web proxies. These machines also process the results returned by the work pool.

- Host the database on a dedicated virtual machine with substantial storage.
- Host the file store on a dedicated virtual machine with substantial storage.

Examples

There's an enormous variation in how companies organize their source code and development workflow. This makes it difficult to predict the hardware requirements for LGTM Enterprise in advance of installation, which is why LGTM is designed to scale smoothly. To help you get an idea of the likely requirements, here are some examples of the resources used by production instances of LGTM.

LGTM.com

This instance analyzes the work of over 700,000 developers working on code in over 135,000 open source projects. The following resources are used on Google Cloud Platform:

- A dynamic control pool that scales up and down automatically according to the current workload.
- A dynamic work pool that typically runs between 600 and 1250 workers. Each worker runs on a separate, preemptible, virtual machine with 2 vCPUs, 7.5 GB of system memory, and 80 GB of disk space.

LGTM Enterprise

In this example, the instance of LGTM Enterprise analyzes the work of approximately 2,500 developers working on code in over 850 repositories. The following internal virtual machines are used:

- One controller running on a virtual machine with 6 vCPUs 2.6 GHz, 32 GB of system memory, and 1 TB disk space.
- A work pool containing 15 general workers and 1 query worker. These are spread across 8 virtual machines. Each of these virtual machines has 6 vCPUs 2.6 GHz, and 32 GB of system memory.

Incrementally adding and removing capacity

As new projects are added to LGTM Enterprise, the work pool (and occasionally the control pool) will need to be expanded. Machines in the work pool are stateless and can all be configured identically. This makes it easy to scale the pool up and down elastically in response to demand, by provisioning virtual machines from a common image.

System integrations

Integration	User authentication	User authorization	Code review integration
Azure DevOps Server (previously called TFS) 2017 or later (Git and TFVC projects)	✓		
Azure DevOps Services (previously called VSTS) (Git and TFVC projects)	✓	✓	Git projects only
Bitbucket Cloud (Git projects only)	✓	✓	✓
Bitbucket Server 4.5.0 or later (Git projects only)	✓	✓	✓ *
Git—*repository access only*			
GitHub.com	✓	✓	✓
GitHub Enterprise 2.9 or later	✓	✓	✓
GitLab.com	✓	✓	✓
GitLab Enterprise and Community Edition 9.0 or later	✓	✓	✓
OneLogin (SAML authentication)	✓		
Subversion—*repository access only* 1.8 or later			

* Requires that the Bitbucket Server trusts the LGTM instance's SSL certificate, and has the Post Webhooks plugin installed. For the latest information about which versions of Bitbucket the plugin supports, see the plugin's documentation.

Private repositories

User authorization ensures that access rights to projects, configured in a host system, are respected in LGTM Enterprise.

LGTM Enterprise supports adding projects from private repositories hosted on any of the supported cloud-based services for which there is user authorization.

Issue tracker integration

You can configure LGTM Enterprise to create issues in external issue tracking systems for alerts found by LGTM.

You define a webhook issue tracker integation in LGTM Enterprise and then configure an application to send and receive ticket details. For more details, see the topic "Adding issue tracker integration" in the administrator help.

Browser support

The following web browsers are supported for accessing the web pages of an LGTM Enterprise deployment:

- Google Chrome
- Mozilla Firefox
- Apple Safari
- Microsoft Edge

GitHub aims to provide an optimal user experience for the most recent version of all of the browsers mentioned above, and will support browser versions for up to three years after their release.