NAME

scalar - A tool for managing large Git repositories

SYNOPSIS

scalar clone [--single-branch] [--branch <main-branch>] [--full-clone] <url> [<enlistment>]
scalar list
scalar register [<enlistment>]
scalar unregister [<enlistment>]
scalar run (all | config | commit-graph | fetch | loose-objects | pack-files) [<enlistment>]
scalar reconfigure [--all | <enlistment>]
scalar diagnose [<enlistment>]
scalar delete <enlistment>

DESCRIPTION

Scalar is a repository management tool that optimizes Git for use in large repositories. Scalar improves performance by configuring advanced Git settings, maintaining repositories in the background, and helping to reduce data sent across the network.

An important Scalar concept is the enlistment: this is the top-level directory of the project. It usually contains the subdirectory **src/** which is a Git worktree. This encourages the separation between tracked files (inside **src/**) and untracked files, such as build artifacts (outside **src/**). When registering an existing Git worktree with Scalar whose name is not **src**, the enlistment will be identical to the worktree.

The **scalar** command implements various subcommands, and different options depending on the subcommand. With the exception of **clone**, **list** and **reconfigure --all**, all subcommands expect to be run in an enlistment.

The following options can be specified *before* the subcommand:

-C <directory>
: Before running the subcommand, change the working directory. This option imitates the same option of **git**(1).

-c <key>=<value>
: For the duration of running the specified subcommand, configure this setting. This option imitates the same option of **git**(1).

COMMANDS

Clone

clone [<options>] <url> [<enlistment>]

Clones the specified repository, similar to **git-clone**(1). By default, only commit and tree objects are cloned. Once finished, the worktree is located at **<enlistment>/src**.

The sparse-checkout feature is enabled (except when run with **--full-clone**) and the only files present are those in the top-level directory. Use **git sparse-checkout set** to expand the set of directories you want to see, or **git sparse-checkout disable** to expand to all files (see **git-sparse-checkout**(1) for more details). You can explore the subdirectories outside your sparse-checkout by using **git ls-tree HEAD[:<directory>]**.

-b <name>, --branch <name>

Instead of checking out the branch pointed to by the cloned repository's HEAD, check out the **<name>** branch instead.

--[no-]single-branch

Clone only the history leading to the tip of a single branch, either specified by the **--branch** option or the primary branch remote's **HEAD** points at.

Further fetches into the resulting repository will only update the remote-tracking branch for the branch this option was used for the initial cloning. If the HEAD at the remote did not point at any branch when **--single-branch** clone was made, no remote-tracking branch is created.

--[no-]full-clone

A sparse-checkout is initialized by default. This behavior can be turned off via **--full-clone**.

List

list

List enlistments that are currently registered by Scalar. This subcommand does not need to be run inside an enlistment.

Register

register [<enlistment>]

Adds the enlistment's repository to the list of registered repositories and starts background maintenance. If **<enlistment>** is not provided, then the enlistment associated with the current working directory is registered.

Note: when this subcommand is called in a worktree that is called **src/**, its parent directory is considered to be the Scalar enlistment. If the worktree is *not* called **src/**, it itself will be considered to be the Scalar enlistment.

Unregister

unregister [<enlistment>]

Remove the specified repository from the list of repositories registered with Scalar and stop the scheduled background maintenance.

Run

scalar run (all | config | commit-graph | fetch | loose-objects | pack-files) [<enlistment>]

Run the given maintenance task (or all tasks, if **all** was specified). Except for **all** and **config**, this subcommand simply hands off to **git-maintenance**(1) (mapping **fetch** to **prefetch** and **pack-files** to **incremental-repack**).

These tasks are run automatically as part of the scheduled maintenance, as soon as the repository is registered with Scalar. It should therefore not be necessary to run this subcommand manually.

The **config** task is specific to Scalar and configures all those opinionated default settings that make Git work more efficiently with large repositories. As this task is run as part of **scalar clone** automatically, explicit invocations of this task are rarely needed.

Reconfigure

After a Scalar upgrade, or when the configuration of a Scalar enlistment was somehow corrupted or changed by mistake, this subcommand allows to reconfigure the enlistment.

With the **--all** option, all enlistments currently registered with Scalar will be reconfigured. Use this option after each Scalar upgrade.

Diagnose

diagnose [<enlistment>]

When reporting issues with Scalar, it is often helpful to provide the information gathered by this command, including logs and certain statistics describing the data shape of the current enlistment.

The output of this command is a **.zip** file that is written into a directory adjacent to the worktree in the **src** directory.

Delete

delete <enlistment>

This subcommand lets you delete an existing Scalar enlistment from your local file system, unregistering the repository.

SEE ALSO

git-clone(1), **git-maintenance**(1).

GIT

Part of the **git**(1) suite