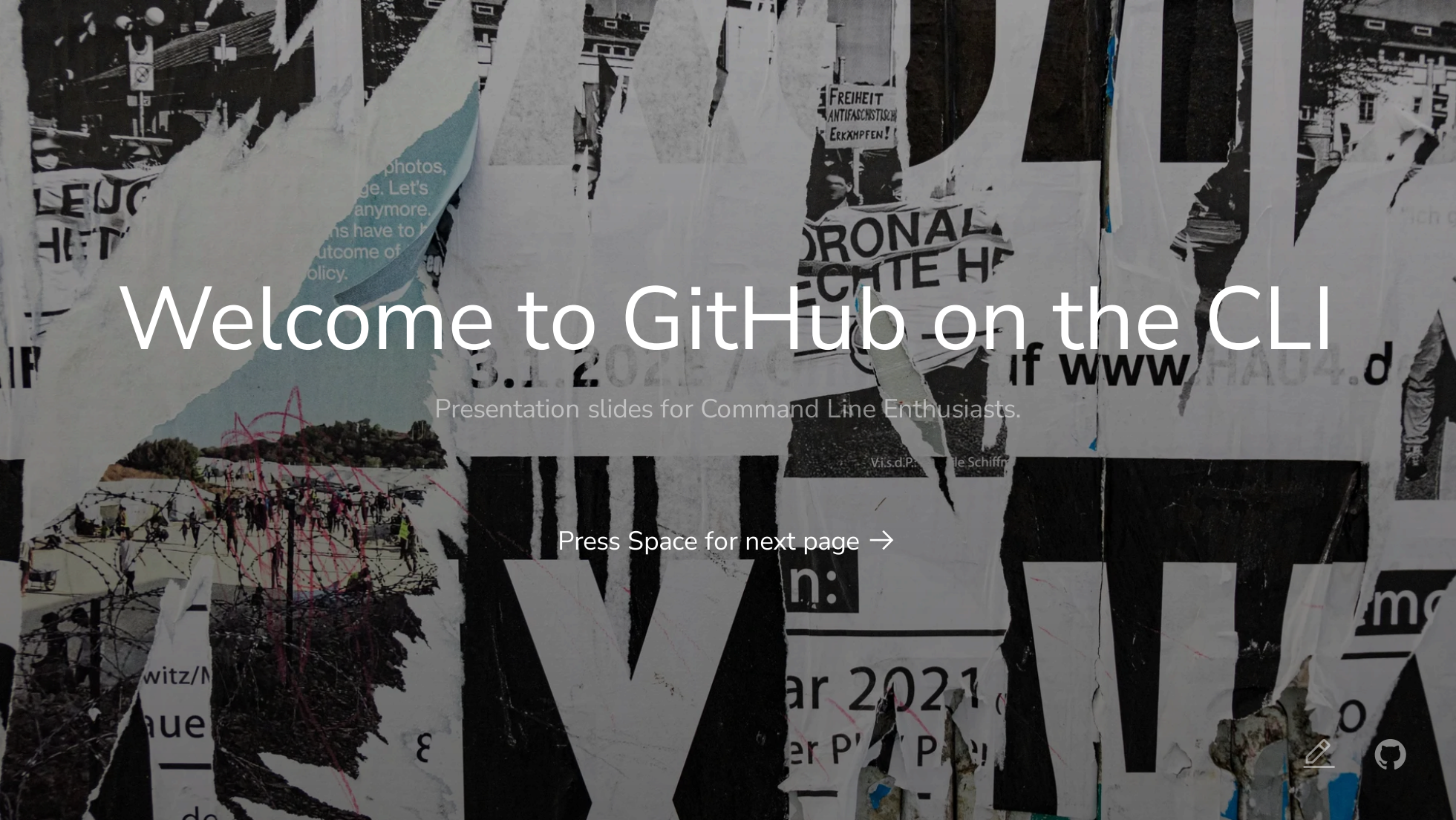

Welcome to GitHub on the CLI

Presentation slides for Command Line Enthusiasts.

Press Space for next page →

What is Github CLI (gh)?

gh brings GitHub to your terminal. Free and open source.

gh has come to stay as first-class building block in the GitHub universe (130 releases in 3.5y!)

gh is written in Go and therefore available on any platform.

Ok, but why should I use GitHub CLI?

> Goodbye, context switching between your terminal and your browser for a seamless and mouse-less efficient developer experience.

- **Entire GitHub workflow:** Work with issues, pull request, checks, release and more on the CLI
- **Script, automate and customize** almost any action with builtin commands
- **GitHub API:** eventually also everything else available via the GitHub API
- **Enterprise-ready:** works with GitHub.com and GitHub Enterprise Server

Where can I get help on GitHub CLI?

gh has an excellent built-in help that follow common usage:

```
gh --help
gh COMMAND [SUBCOMMAND] --help
```

These built-in help texts are also available as a manual at https://cli.github.com/manual.

The official micro-website on Github CLI is at https://cli.github.com with a short visual introduction to the tool and links to the **manual** and the **release notes**. The latter are well maintained and a good read after every upgrade.

gh is open source. Find the repo at https://github.com/cli/cli

Agenda

1. Getting Started
2. Installing the GitHub CLI binary
3. Manage Permission Scopes
4. Working with Repositories
5. Working with Issues
6. Working with Pull Requests
7. Other Interesting Features
    1. GitHub REST API Usage
    2. Project Commands (new)
    3. Release Management
    4. Workflow Management
    5. Extending GitHub CLI
    6. Label Management

Getting Started

Installing and Configuring

Installing the GitHub CLI binary

Follow the installation instructions at https://github.com/cli/cli#installation for your platform.

For me on macOS using Homebrew:

```
brew install gh
```

On macOS:
gh is available via Homebrew, MacPorts, Conda, Spack

On Windows:
gh is available via WinGet, scoop, Chocolatey, Conda, and as downloadable MSI.

On Linux:
gh is available either via official repos or as package downloads - see Installing gh on Linux and BSD

Authenticating to GH and GHES

Login to github.com or our GHES

```
gh auth login
```

```
➜ gh auth login
? What account do you want to log into? GitHub Enterprise Server
? GHE hostname: github.unibe.ch
? What is your preferred protocol for Git operations? HTTPS
? Authenticate Git with your GitHub credentials? Yes
? How would you like to authenticate GitHub CLI? Login with a web browser

Press Enter to open github.unibe.ch in your browser...
✓ Authentication complete.
- gh config set -h github.unibe.ch git_protocol https
✓ Configured git protocol
✓ Logged in as michael-rolli
```

Authentication (cont.)

Show current authentication status

```
gh auth status
```

```
➜ gh auth status
github.com
  ✓ Logged in to github.com as mrolli (keyring)
  ✓ Git operations for github.com configured to use https protocol.
  ✓ Token: gho_************************************
  ✓ Token scopes: gist, read:org, read:project, repo, workflow

github.unibe.ch
  ✓ Logged in to github.unibe.ch as michael-rolli (keyring)
  ✓ Git operations for github.unibe.ch configured to use https protocol.
  ✓ Token: gho_************************************
  ✓ Token scopes: gist, read:org, repo, workflow
```

Manage Permission Scopes

As an example, we are going to add the read:project permission scope to the GHES token:

```
gh auth refresh --scopes read:project --hostname github.unibe.ch
```

For more information on managing permission scopes, see gh auth refresh --help

Setup the gh Credential Helper

Configure git to use GitHub CLI as the credential helper for all authenticated hosts

```
gh auth setup-git
```

This edits your git configuration file.

Working with Repositories

Creating, cloning, forking and all the rest

Overview of Repo Commands

```
Work with GitHub repositories.

USAGE
  gh repo <command> [flags]

GENERAL COMMANDS
  create:      Create a new repository
  list:        List repositories owned by user or organization

TARGETED COMMANDS
  archive:     Archive a repository
  clone:       Clone a repository locally
  delete:      Delete a repository
  deploy-key:  Manage deploy keys in a repository
  edit:        Edit repository settings
  fork:        Create a fork of a repository
  rename:      Rename a repository
  set-default: Configure default repository for this directory
  sync:        Sync a repository
  unarchive:   Unarchive a repository
  view:        View a repository
```

Listing Repositories

- Show all repos my Repositories on github.com

```
gh repo list
```

- Show all repos of IDSYS on github.com

```
gh repo list idsys-unibe-ch
```

- List all repos of IDSYS on our GHES

```
GH_HOST=github.unibe.ch gh repo list idsys-unibe-ch
```

Having a hard-coded environment variable is not an ideal solution when you are using both GitHub platforms, therefore an elegant alias might come in handy:

```
# Add this to your .bashrc or .zshrc
alias lgh="GH_HOST=github.unibe.ch gh"
```

Forking and Cloning Repositories

- Clone an existing repo of mine

```
gh repo clone mrolli/testy
```

- Fork a repository of another user/organisation

```
gh repo fork idsys-unibe-ch/forkme4edu
```

This creates a fork in your user profile interactively. After forking you work on feature branches in your fork and then you create a pull request against the main branch of the upstream repository.

> This is the preferred way to contribute to repos where you don't have any permissions. The target repo only needs to be public.

Creating new repositories

- Create a new repo on github.com interactively

```
gh repo create
```

- Create a new repo using flags and clone it locally

```
gh repo create my-project --public --clone
```

- Create a new repo on github.com from an existing local repo:

```
git init
git add .
git commit -m "Initial commit"
gh repo create myorg/myrepo --public --source=.
git push -u origin main
```

Working with Issues

Creating, commenting, properties, templates, ...

Overview of Issue Commands

```
Work with GitHub issues.

USAGE
  gh issue <command> [flags]

GENERAL COMMANDS
  create:      Create a new issue
  list:        List issues in a repository
  status:      Show status of relevant issues

TARGETED COMMANDS
  close:       Close issue
  comment:     Add a comment to an issue
  delete:      Delete issue
  develop:     Manage linked branches for an issue
  edit:        Edit issues
  lock:        Lock issue conversation
  pin:         Pin a issue
  reopen:      Reopen issue
  transfer:    Transfer issue to another repository
  unlock:      Unlock issue conversation
  unpin:       Unpin a issue
  view:        View an issues
```

Examples for working with issues

- List closed issues:

```
gh issue list -s closed
```

- List all open issue that are relevant for me:

```
gh issue status
```

- List all issue in organisation that are relevant for me:

```
gh status -o idsys-unibe-ch
```

- View specific issue in terminal...

```
gh issue view
```

- Edit specific issue:

```
gh issue edit 123 --add-label bug
```

- Comment on specific issue using the content of a file:

```
gh issue comment 123 --body-file ../output.log
```

Working with Pull Requests

Creating, commenting, templates, merging ...

Overview of Pull Request commands

```
Work with GitHub pull requests.

USAGE
  gh pr <command> [flags]

GENERAL COMMANDS
  create:      Create a pull request
  list:        List pull requests in a repository
  status:      Show status of relevant pull requests

TARGETED COMMANDS
  checkout:    Check out a pull request in git
  checks:      Show CI status for a single pull request
  close:       Close a pull request
  comment:     Add a comment to a pull request
  diff:        View changes in a pull request
  edit:        Edit a pull request
  lock:        Lock pull request conversation
  merge:       Merge a pull request
  ready:       Mark a pull request as ready for review
  reopen:      Reopen a pull request
  review:      Add a review to a pull request
  unlock:      Unlock pull request conversation
  view:        View a pull request
```

Examples for working with pull requests

- List open pull requests:

```
gh pr list
```

- Work on an issue by creating a linked branch and switch to the branch:

```
gh issue develop 123 --checkout
```

- List all open PRs that are relevant for me:

```
gh gh pr status
```

- Create a new PR interactively:

```
gh pr create
```

- Show check runs on specific PR:

```
gh pr check 123
```

- Merge specific PR using the rebase merge method:

```
gh pr merge --rebase
```

On Merge Methods

When working with PR, Github features three different merge methods:

- Merge pull request
- Squash and merge
- Rebase and merge

The allowed merge methods can be configured on a per repo basis.

> In a project, wisely choose one and stick to it! There is no right or false!

More information on this topic:

- Official documentation on merge methods
- GitHub Merge straegies explained by examples

Also mentioned here shall be the merge queue feature that GitHub offers.

Other Interesting Features

API, projects, releases, workflows, aliases, labels, extensions, ...

GitHub REST API Usage

```
function setup_snow_autolinkref {
  autolinkref=$(gh api --method GET \
    -H "Accept: application/vnd.github+json" \
    -H "X-GitHub-Api-Version: 2022-11-28" \
    "/repos/$1/autolinks" --jq '.[] | select(.key_prefix=="SNOW-")'
  )

  if [ -n "$autolinkref" ]; then
    success "Autolink reference for SNOW already setup."
    return 0
  fi

  gh api \
    --method POST \
    -H "Accept: application/vnd.github+json" \
    -H "X-GitHub-Api-Version: 2022-11-28" \
    "/repos/$1/autolinks" \
    -f key_prefix="SNOW-" \
   -f url_template="https://serviceportal.unibe.ch/text_search_exact_match.do?sysparm_search=<num>" \
   -F is_alphanumeric=true
}
```

See https://docs.github.com/en/rest

Overview of Project Commands

```
Work with GitHub Projects. Note that the token you are using must have 'project'
scope, which is not set by default. You can verify your token scope by running
'gh auth status' and add the project scope by running 'gh auth refresh -s
project'.

USAGE
  gh project <command> [flags]

AVAILABLE COMMANDS
  close:       Close a project
  copy:        Copy a project
  create:      Create a project
  delete:      Delete a project
  edit:        Edit a project
  field-create: Create a field in a project
  field-delete: Delete a field in a project
  field-list:  List the fields in a project
  item-add:    Add a pull request or an issue to a project
  item-archive: Archive an item in a project
  item-create: Create a draft issue item in a project
  item-delete: Delete an item from a project by ID
  item-edit:   Edit an item in a project
  item-list:   List the items in a project
```

Managing Project Releases

It's possible to create releases based on tags right from the CLI. First prepare the tag to create a release for:

```
git checkout main
git tag -a -s -m "Release v1.0" v1.0
git push origin --tags
```

Then create the release - examples:

- Interactively create a release

```
gh release create
```

* Non-interactively create a release

```
gh release create v1.2.3 --notes "bugfix release"
```

* Use automatically generated release notes

```
gh release create v1.2.3 --generate-notes
```

* Create a release and start a discussion

```
gh release create v1.2.3 --discussion-category "General"
```

Managing Workflows and Check Runs

```
List, view, and run workflows in GitHub Actions.

  gh workflow <command> [flags]

AVAILABLE COMMANDS
  disable:     Disable a workflow
  enable:      Enable a workflow
  list:        List workflows
  run:         Run a workflow by creating a workflow_dispatch event
  view:        View the summary of a workflow
```

```
List, view, and watch recent workflow runs from GitHub Actions.

  gh run <command> [flags]

AVAILABLE COMMANDS
  cancel:      Cancel a workflow run
  delete:      Delete a workflow run
  download:    Download artifacts generated by a workflow run
  list:        List recent workflow runs
  rerun:       Rerun a run
  view:        View a summary of a workflow run
  watch:       Watch a run until it completes, showing its progress
```

Extensions

Overview of Extension Commands

```
AVAILABLE COMMANDS
  browse:      Enter a UI for browsing, adding, and removing extensions
  create:      Create a new extension
  exec:        Execute an installed extension
  install:     Install a gh extension from a repository
  list:        List installed extension commands
  remove:      Remove an installed extension
  search:      Search extensions to the GitHub CLI
  upgrade:     Upgrade installed extensions
```

Noteworthy extensions

```
❯ gh extension list
gh label             heaths/gh-label               v0.4.0
gh markdown-preview  yusukebe/gh-markdown-preview  23d1a241
```

Managing GitHub Labels

- Builtin labels commands

```
AVAILABLE COMMANDS
  clone:       Clones labels from one repository to another
  create:      Create a new label
  delete:      Delete a label from a repository
  edit:        Edit a label
  list:        List labels in a repository
```

- Commands in heaths/gh-labels extensions

```
Available Commands:
  completion  generate the autocompletion script for the specified shell
  create      Create the label <name> in the repository
  delete      Delete the label <name> from the repository
  edit        Edit the label <name> in the repository
  export      Export labels from the repository to <path>, or stdout if <path> is "-".
  help        Help about any command
  import      Import labels into the repository from <path>, or stdin if <path> is "-".
  list        List labels in the repository, optionally matching substring [name] in the label name or description
```

Thank you!

Questions? More feature demonstration?