Developing Zeek scripts with style ✨

Munich Zeek meetup

```
@load-sigs frameworks/signatures/detect-windows-shells

# Load all of the scripts that detect software in various protocols.
@load protocols/ftp/software
@load protocols/smtp/software
@load protocols/ssh/software
@load protocols/http/software
# The detect-webapps script could possibly cause performance trouble when
# running on live traffic.  Enable it cautiously.
#@load protocols/http/detect-webapps

# This script detects DNS results pointing toward your Site::local_nets
# where the name is not part of your local DNS zone and is being hosted
# externally.  Requires that the Site::local_zones variable is defined.
@load protocols/dns/detect-external-names

# Script to detect various activity in FTP sessions.
@load protocols/ftp/detect

# Scripts that do asset tracking.
@load protocols/conn/known-hosts
@load protocols/conn/known-services
@load protocols/ssl/known-certs

# This script enables SSL/TLS certificate validation.
@load protocols/ssl/validate-certs

# This script prevents the logging of SSL CA certificates in x509.log
@load protocols/ssl/log-hostcerts-only

# If you have GeoIP support built in, do some geographic detections and
# logging for SSH traffic.
@load protocols/ssh/geo-data
# Detect hosts doing SSH bruteforce attacks.
:
```

```
@load-sigs frameworks/signatures/detect-windows-shells

# Load all of the scripts that detect software in various protocols.
@load protocols/ftp/software
@load protocols/smtp/software
@load protocols/ssh/software
@load protocols/http/software
# The detect-webapps script could possibly cause performance trouble when
# running on live traffic.  Enable it cautiously.
#@load protocols/http/detect-webapps

# This script detects DNS results pointing toward your Site::local_nets
# where the name is not part of your local DNS zone and is being hosted
# externally.  Requires that the Site::local_zones variable is defined.
@load protocols/dns/detect-external-names

# Script to detect various activity in FTP sessions.
@load protocols/ftp/detect

# Scripts that do asset tracking.
@load protocols/conn/known-hosts
@load protocols/conn/known-services
@load protocols/ssl/known-certs

# This script enables SSL/TLS certificate validation.
@load protocols/ssl/validate-certs

# This script prevents the logging of SSL CA certificates in x509.log
@load protocols/ssl/log-hostcerts-only

# If you have GeoIP support built in, do some geographic detections and
# logging for SSH traffic.
@load protocols/ssh/geo-data
# Detect hosts doing SSH bruteforce attacks.
:
```

Outline

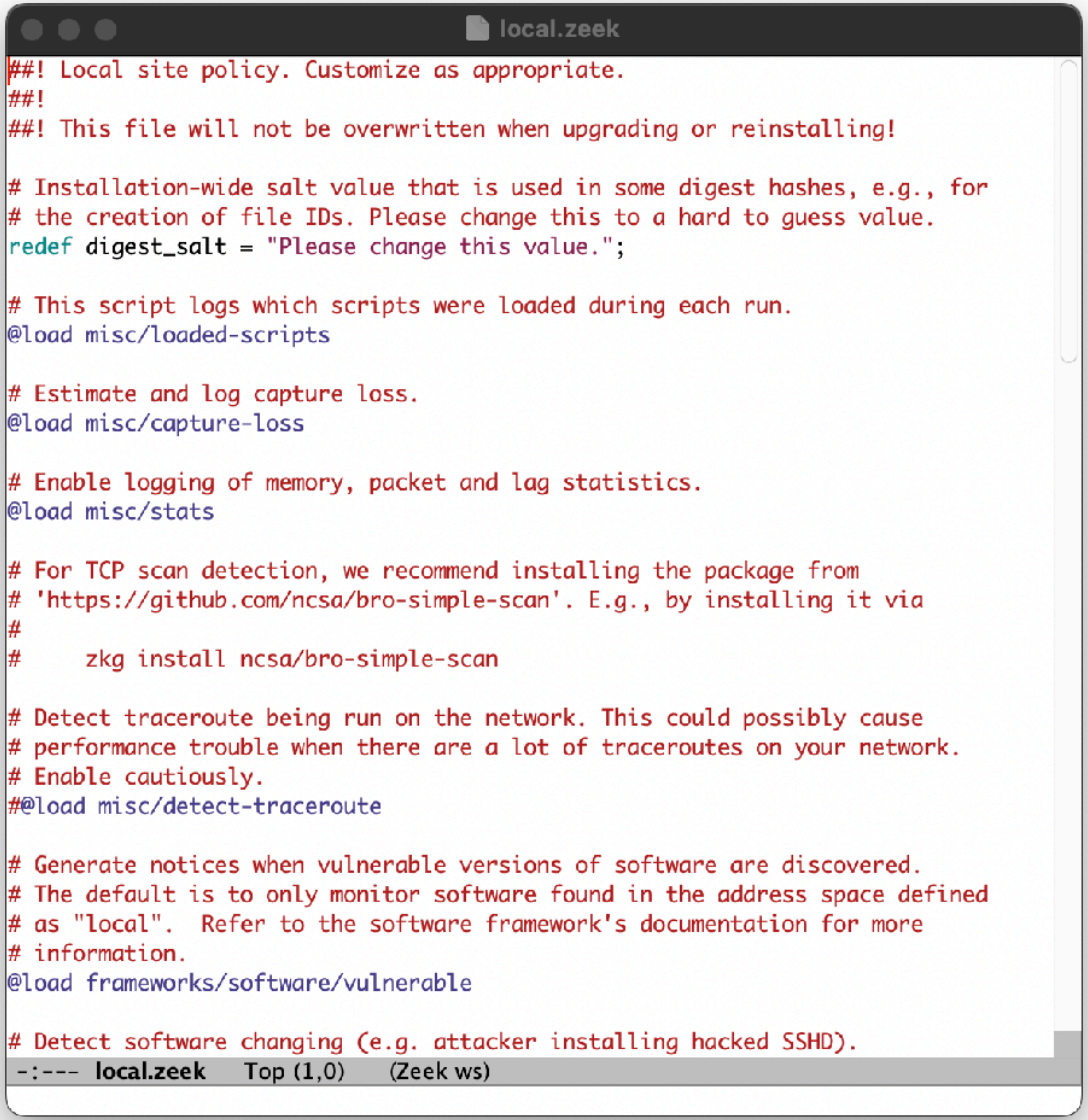
1. Syntax highlighting
2. Source code formatting
3. Linting
4. IDE-like experience for developing Zeek code

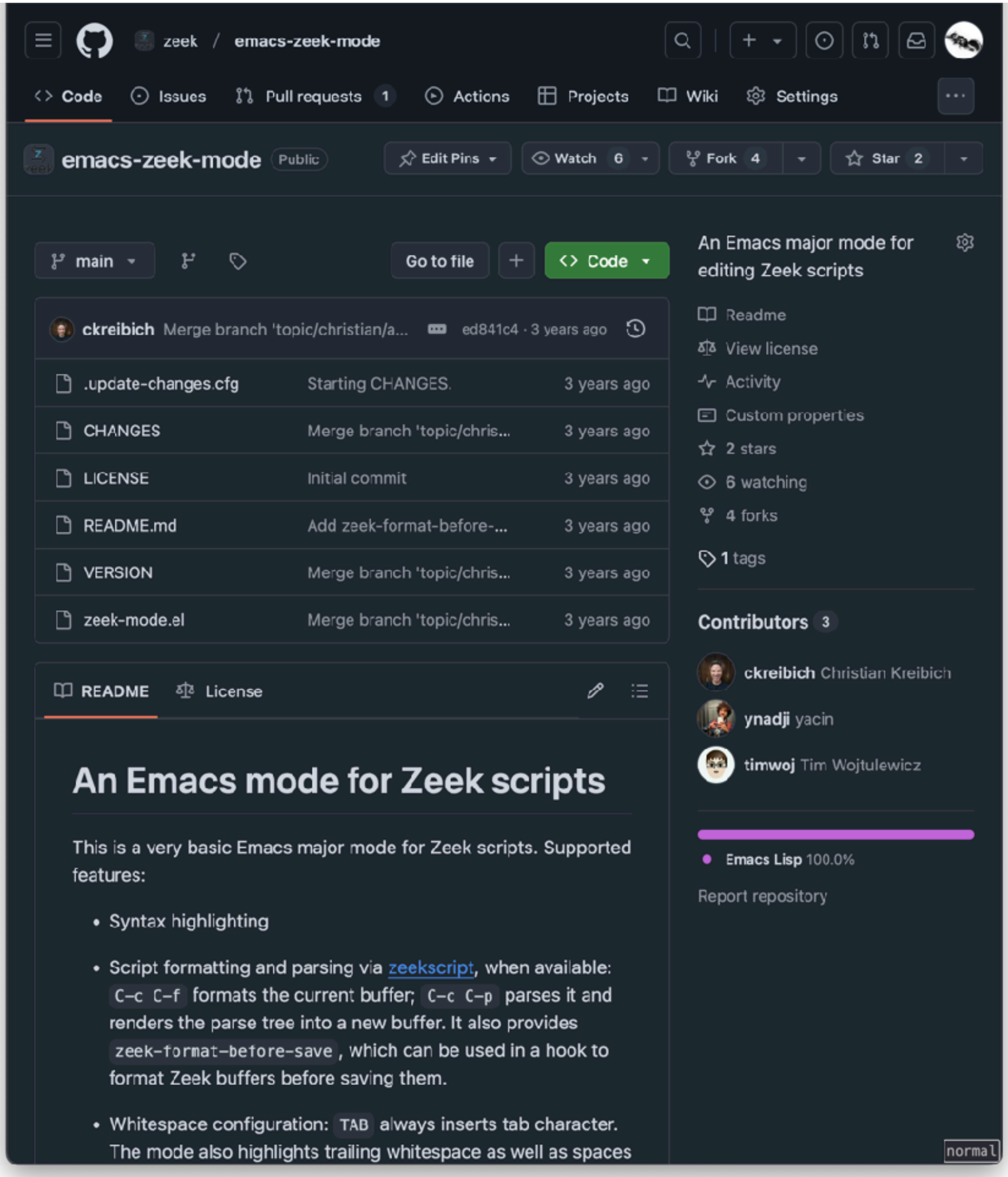
Syntax highlighting

local.zeek

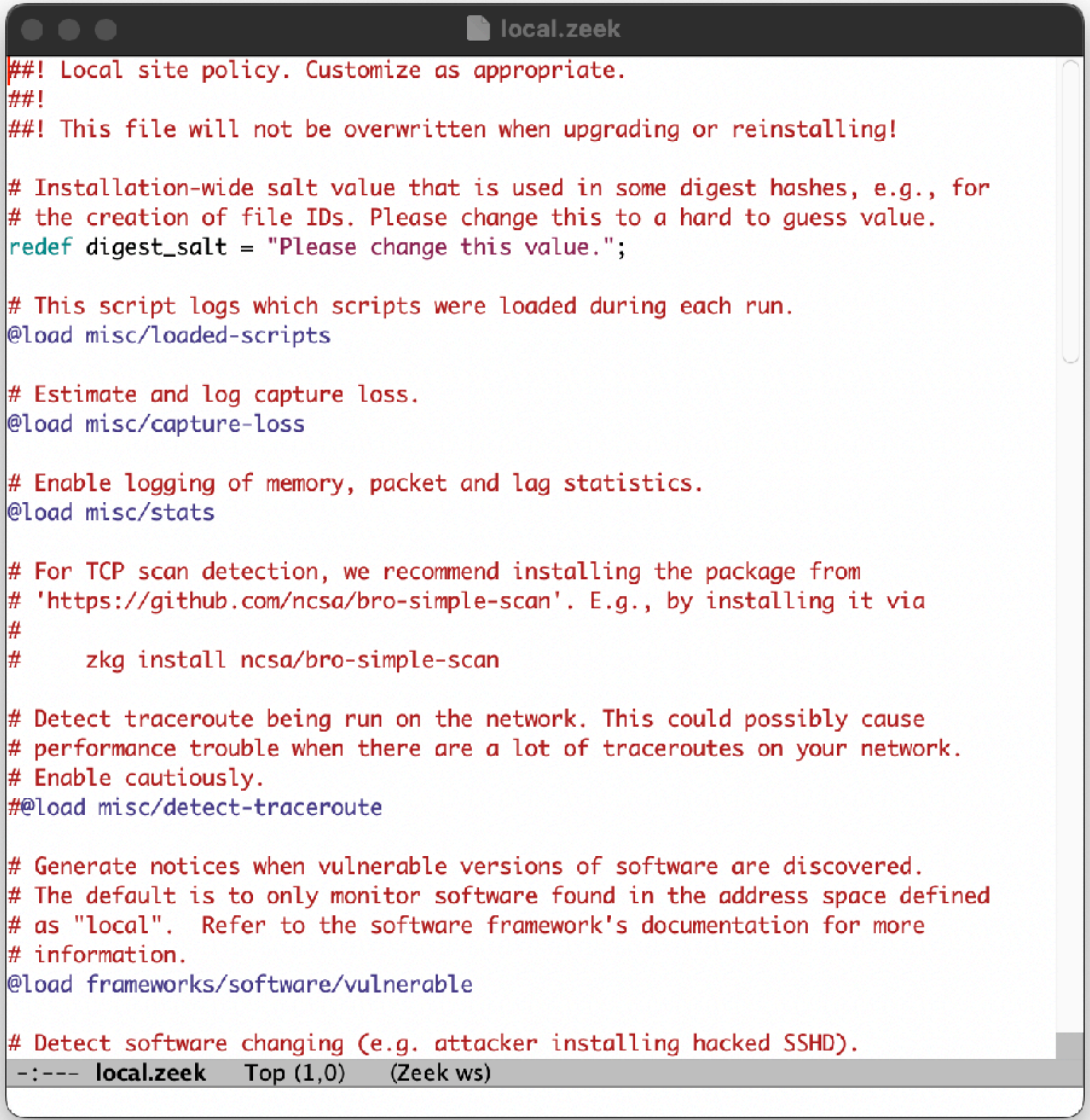
```
##! Local site policy. Customize as appropriate.
##!
##! This file will not be overwritten when upgrading or reinstalling!

# Installation-wide salt value that is used in some digest hashes, e.g., for
# the creation of file IDs. Please change this to a hard to guess value.
redef digest_salt = "Please change this value.";

# This script logs which scripts were loaded during each run.
@load misc/loaded-scripts

# Estimate and log capture loss.
@load misc/capture-loss

# Enable logging of memory, packet and lag statistics.
@load misc/stats

# For TCP scan detection, we recommend installing the package from
# 'https://github.com/ncsa/bro-simple-scan'. E.g., by installing it via
#
#     zkg install ncsa/bro-simple-scan

# Detect traceroute being run on the network. This could possibly cause
# performance trouble when there are a lot of traceroutes on your network.
# Enable cautiously.
#@load misc/detect-traceroute

# Generate notices when vulnerable versions of software are discovered.
# The default is to only monitor software found in the address space defined
# as "local".  Refer to the software framework's documentation for more
# information.
@load frameworks/software/vulnerable

# Detect software changing (e.g. attacker installing hacked SSHD).
```

-:--- local.zeek   Top (1,0)   (Zeek ws) http://github.com/zeek/emacs-zeek-mode

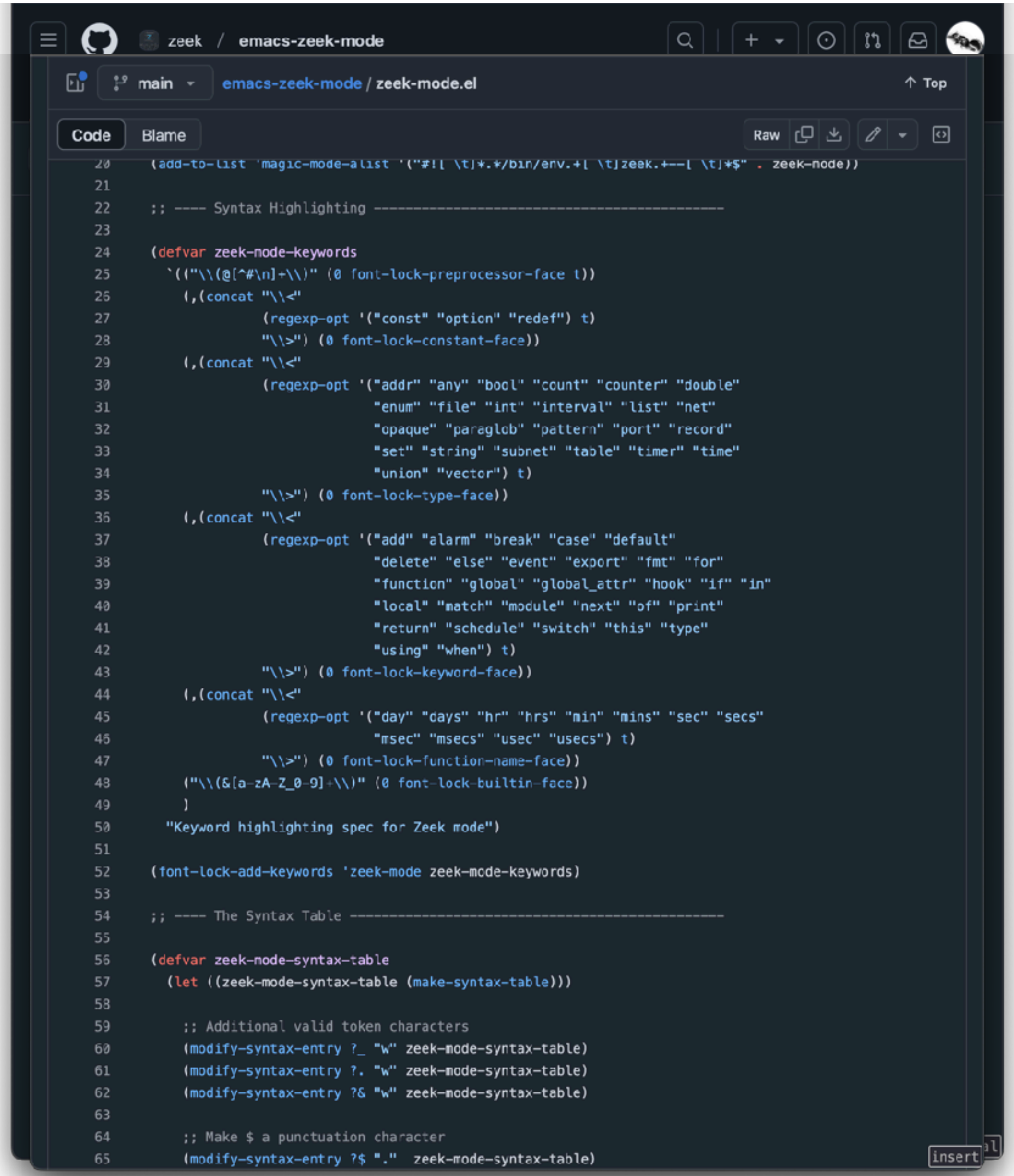




http://github.com/zeek/emacs-zeek-mode




```
##! Local site policy. Customize as appropriate.
##!
##! This file will not be overwritten when upgrading or reinstalling!

# Installation-wide salt value that is used in some digest hashes, e.g., for
# the creation of file IDs. Please change this to a hard to guess value.
redef digest_salt = "Please change this value.";

# This script logs which scripts were loaded during each run.
@load misc/loaded-scripts

# Estimate and log capture loss.
@load misc/capture-loss

# Enable logging of memory, packet and lag statistics.
@load misc/stats

# For TCP scan detection, we recommend installing the package from
# 'https://github.com/ncsa/bro-simple-scan'. E.g., by installing it via
#
#     zkg install ncsa/bro-simple-scan

# Detect traceroute being run on the network. This could possibly cause
# performance trouble when there are a lot of traceroutes on your network.
# Enable cautiously.
#@load misc/detect-traceroute

# Generate notices when vulnerable versions of software are discovered.
# The default is to only monitor software found in the address space defined
# as "local".  Refer to the software framework's documentation for more
# information.
@load frameworks/software/vulnerable

# Detect software changing (e.g. attacker installing hacked SSHD).
```


```
(add-to-list 'magic-mode-alist '("#![ \t]*.*/bin/env.+[ \t]zeek.+--[ \t]*$" . zeek-mode))

;; ---- Syntax Highlighting -------------------------------------------

(defvar zeek-mode-keywords
  `(("\\(@[^#\n]+\\)" (0 font-lock-preprocessor-face t))
    (,(concat "\\<"
              (regexp-opt '("const" "option" "redef") t)
              "\\>") (0 font-lock-constant-face))
    (,(concat "\\<"
              (regexp-opt '("addr" "any" "bool" "count" "counter" "double"
                            "enum" "file" "int" "interval" "list" "net"
                            "opaque" "paraglob" "pattern" "port" "record"
                            "set" "string" "subnet" "table" "timer" "time"
                            "union" "vector") t)
              "\\>") (0 font-lock-type-face))
    (,(concat "\\<"
              (regexp-opt '("add" "alarm" "break" "case" "default"
                            "delete" "else" "event" "export" "fmt" "for"
                            "function" "global" "global_attr" "hook" "if" "in"
                            "local" "match" "module" "next" "of" "print"
                            "return" "schedule" "switch" "this" "type"
                            "using" "when") t)
              "\\>") (0 font-lock-keyword-face))
    (,(concat "\\<"
              (regexp-opt '("day" "days" "hr" "hrs" "min" "mins" "sec" "secs"
                            "msec" "msecs" "usec" "usecs") t)
              "\\>") (0 font-lock-function-name-face))
    ("\\(&[a-zA-Z_0-9]+\\)" (0 font-lock-builtin-face))
    )
  "Keyword highlighting spec for Zeek mode")

(font-lock-add-keywords 'zeek-mode zeek-mode-keywords)

;; ---- The Syntax Table ----------------------------------------------

(defvar zeek-mode-syntax-table
  (let ((zeek-mode-syntax-table (make-syntax-table)))

    ;; Additional valid token characters
    (modify-syntax-entry ?_ "w" zeek-mode-syntax-table)
    (modify-syntax-entry ?. "w" zeek-mode-syntax-table)
    (modify-syntax-entry ?& "w" zeek-mode-syntax-table)

    ;; Make $ a punctuation character
    (modify-syntax-entry ?$ "."  zeek-mode-syntax-table)
```

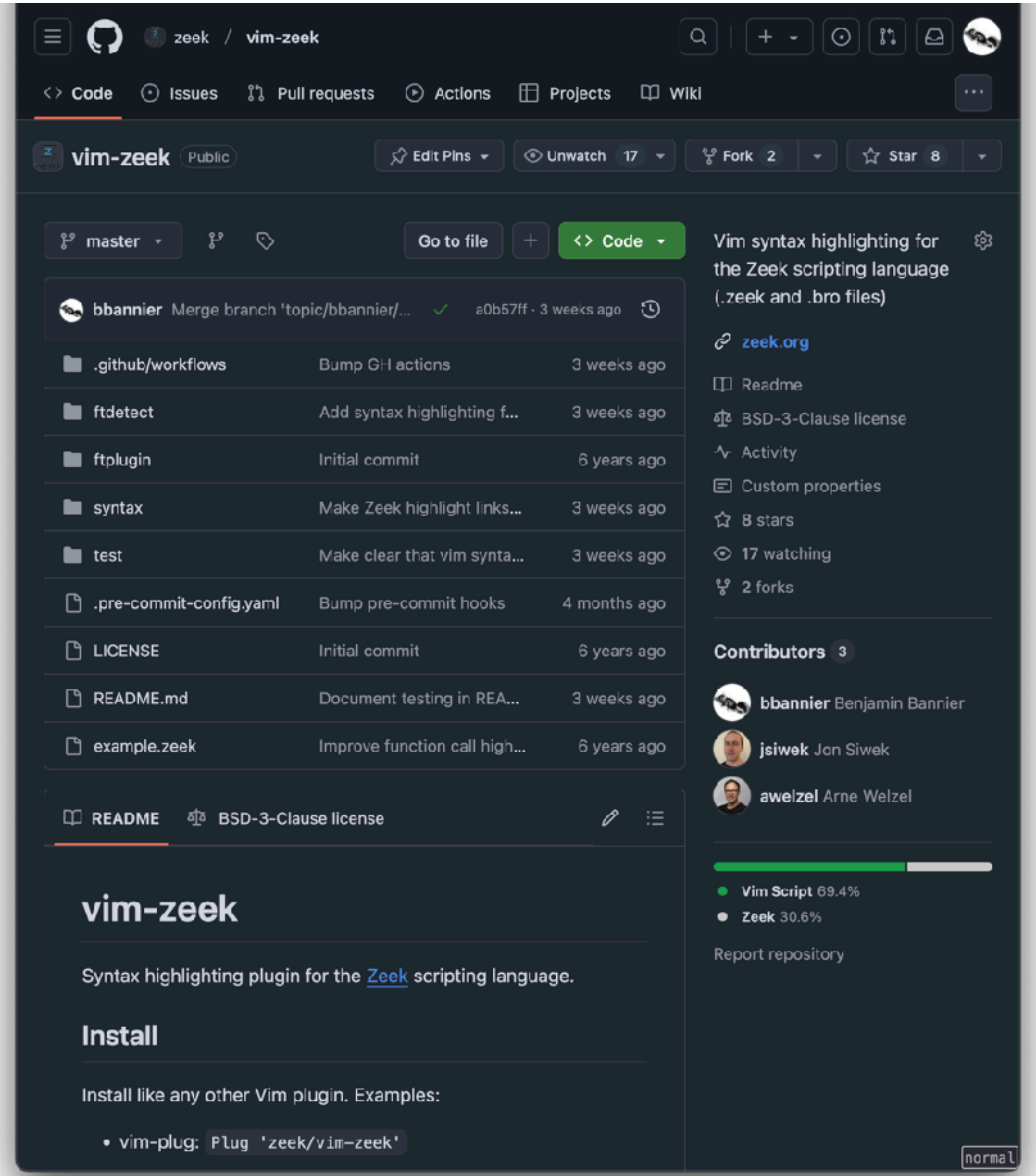
local.zeek (/opt/homebrew/Cellar/zeek/7.1.0/share/zeek/site) - NVIM

/opt/homebrew/Cellar/zeek/7.1.0/shar...

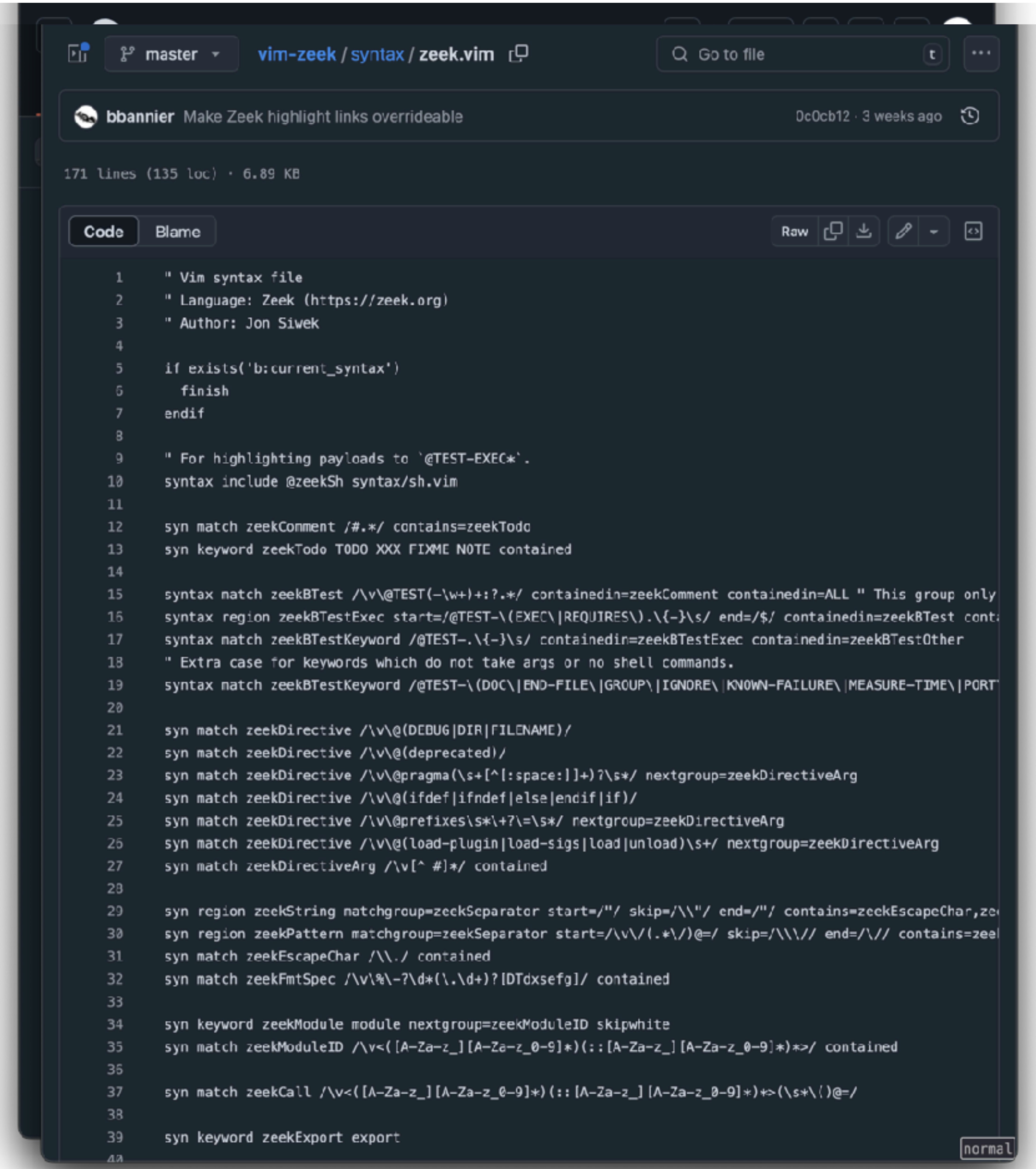
```
##! Local site policy. Customize as appropriate.
##!
##! This file will not be overwritten when upgrading or reinstalling!

# Installation-wide salt value that is used in some digest hashes, e.g., for
# the creation of file IDs. Please change this to a hard to guess value.
redef digest_salt = "Please change this value.";

# This script logs which scripts were loaded during each run.
@load misc/loaded-scripts

# Estimate and log capture loss.
@load misc/capture-loss

# Enable logging of memory, packet and lag statistics.
@load misc/stats

# For TCP scan detection, we recommend installing the package from
# 'https://github.com/ncsa/bro-simple-scan'. E.g., by installing it via
#
#     zkg install ncsa/bro-simple-scan

# Detect traceroute being run on the network. This could possibly cause
# performance trouble when there are a lot of traceroutes on your network.
# Enable cautiously.
#@load misc/detect-traceroute

# Generate notices when vulnerable versions of software are discovered.
# The default is to only monitor software found in the address space defined
# as "local".  Refer to the software framework's documentation for more
# information.
@load frameworks/software/vulnerable

# Detect software changing (e.g. attacker installing hacked SSHD).
@load frameworks/software/version-changes

# This adds signatures to detect cleartext forward and reverse windows shells.
@load-sigs frameworks/signatures/detect-windows-shells

# Load all of the scripts that detect software in various protocols.
@load protocols/ftp/software
@load protocols/smtp/software
@load protocols/ssh/software
@load protocols/http/software
# The detect-webapps script could possibly cause performance trouble when
# running on live traffic.  Enable it cautiously.
#@load protocols/http/detect-webapps

# This script detects DNS results pointing toward your Site::local_nats
```

NORMAL | master | <ebrew/Cellar/zeek/7.1.0/share/zeek/site/local.zeek | zeek | utf-8[unix] | 0% ln:1/114≡%:1

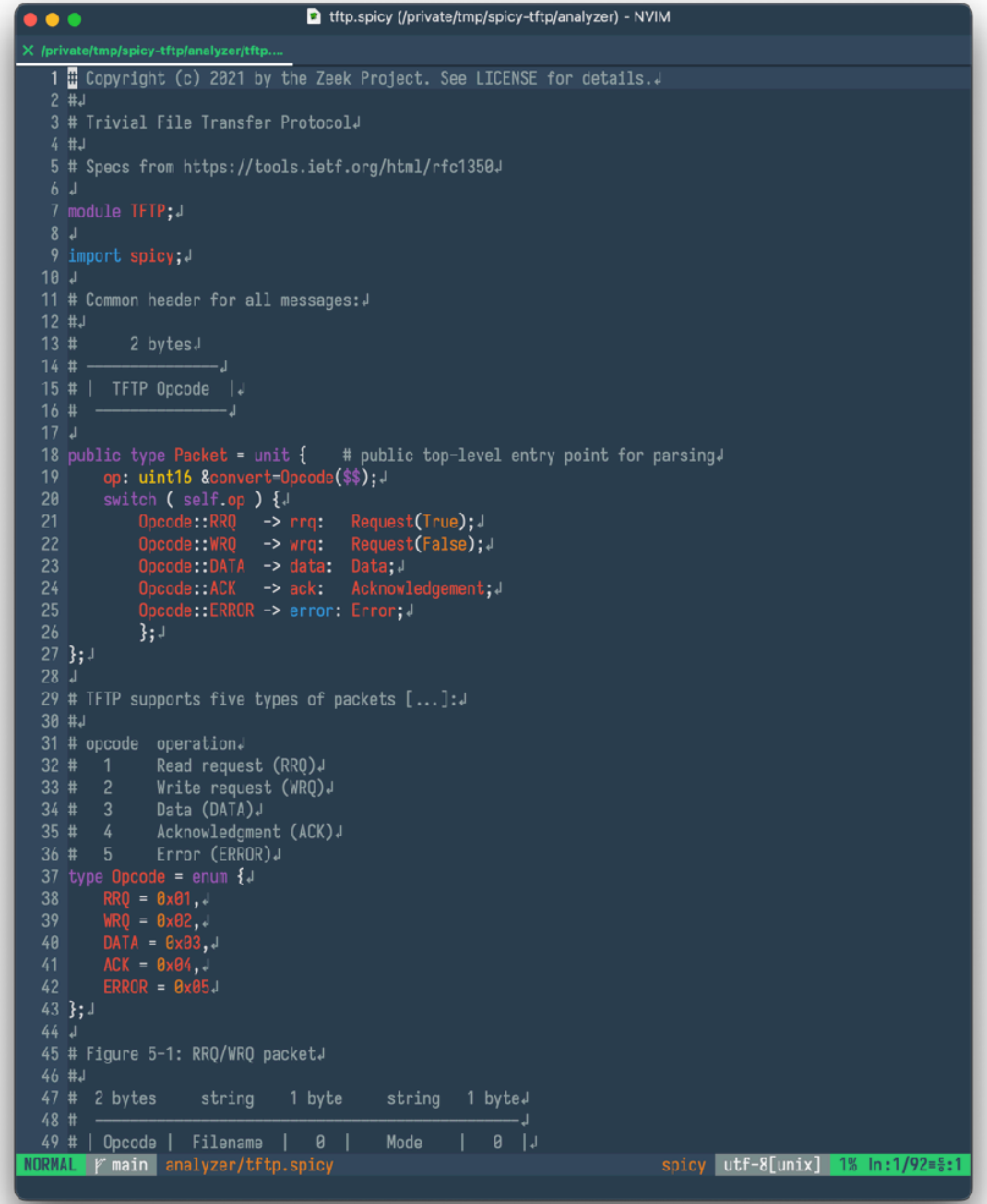
http://github.com/zeek/vim-zeek

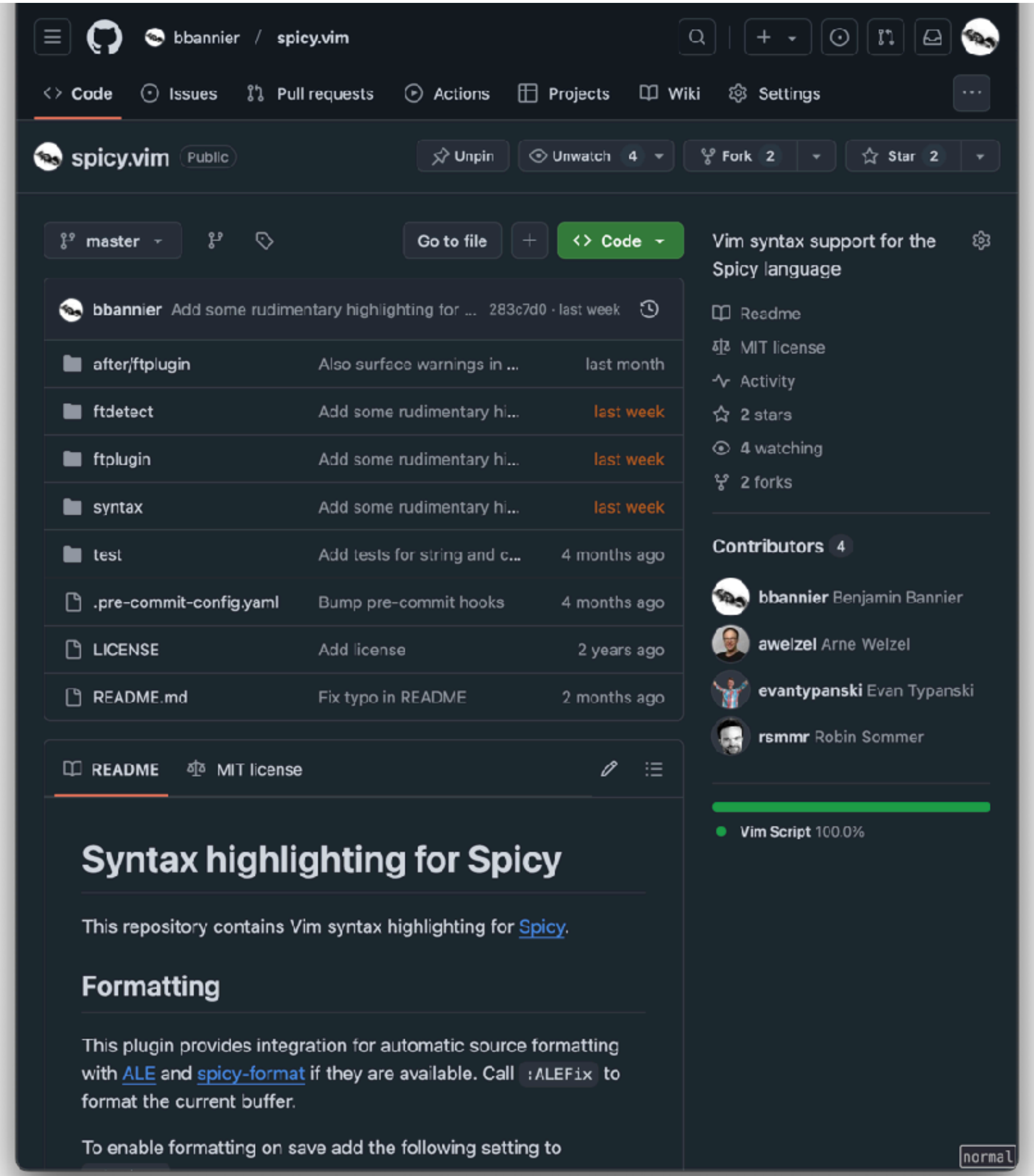


http://github.com/zeek/vim-zeek



http://github.com/bbannier/spicy.vim





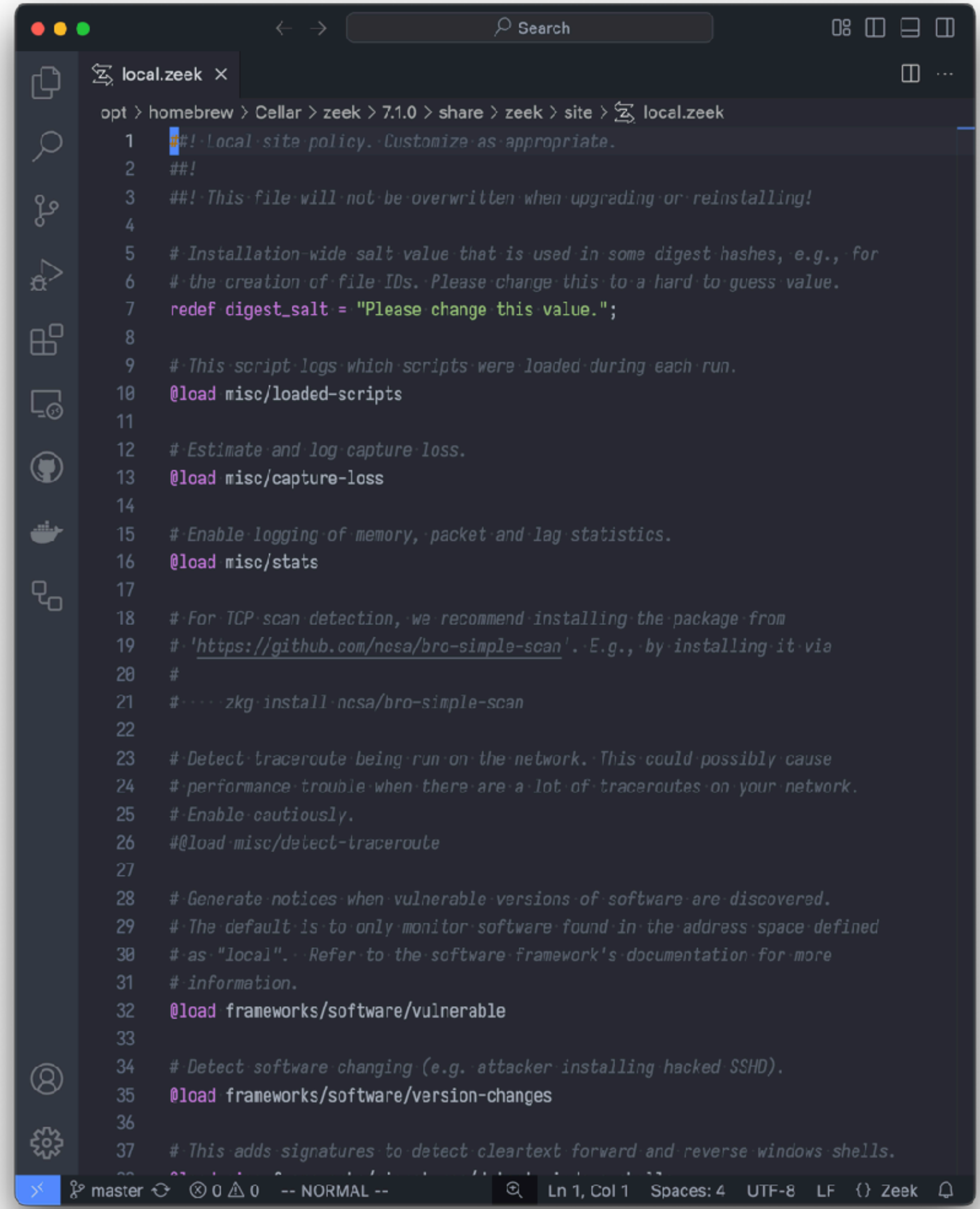
local.zeek

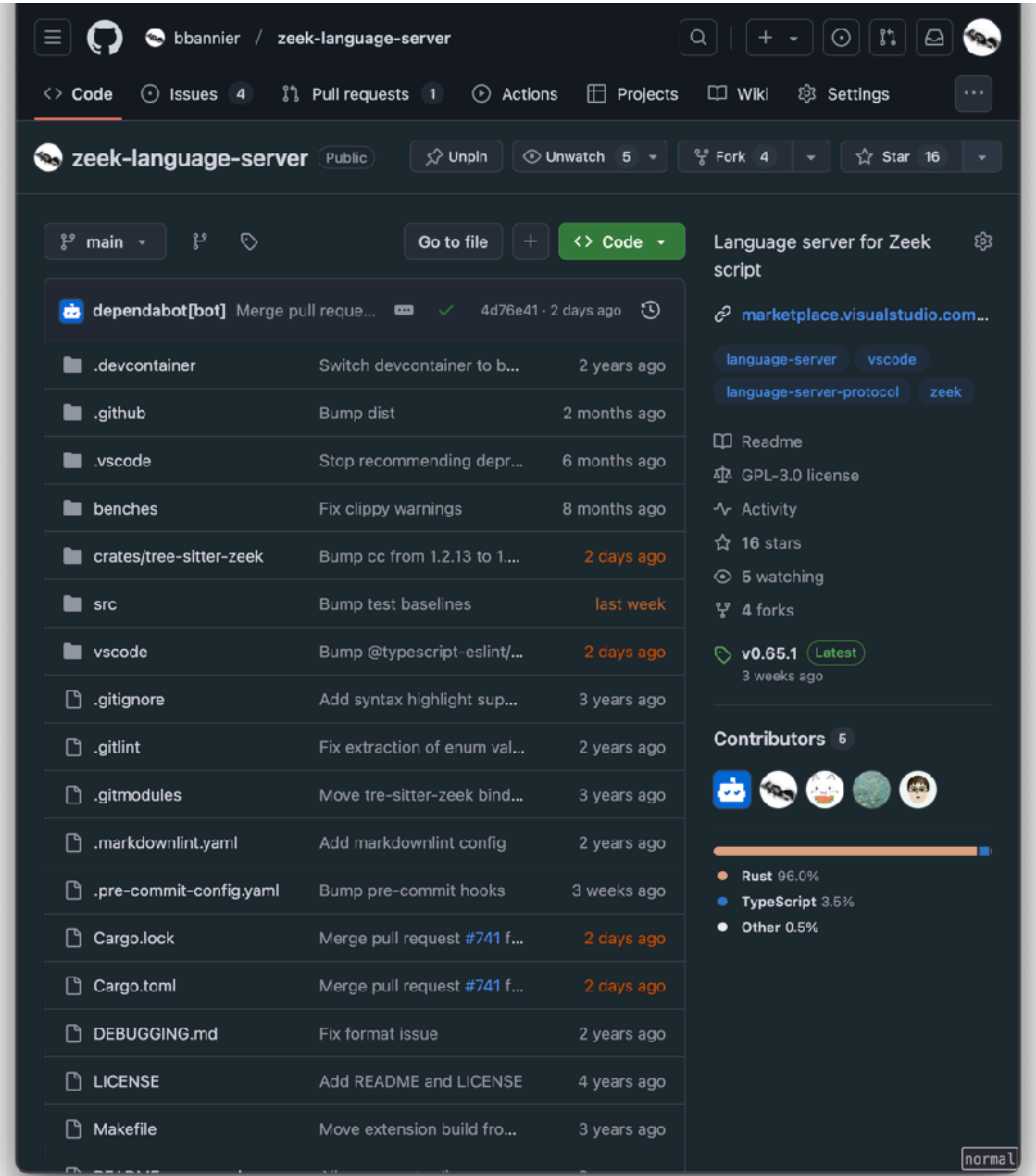
opt > homebrew > Cellar > zeek > 7.1.0 > share > zeek > site > local.zeek

```
##! Local site policy. Customize as appropriate.
##!
##! This file will not be overwritten when upgrading or reinstalling!

# Installation-wide salt value that is used in some digest hashes, e.g., for
# the creation of file IDs. Please change this to a hard to guess value.
redef digest_salt = "Please change this value.";

# This script logs which scripts were loaded during each run.
@load misc/loaded-scripts

# Estimate and log capture loss.
@load misc/capture-loss

# Enable logging of memory, packet and lag statistics.
@load misc/stats

# For TCP scan detection, we recommend installing the package from
# 'https://github.com/ncsa/bro-simple-scan'. E.g., by installing it via
#
#     zkg install ncsa/bro-simple-scan

# Detect traceroute being run on the network. This could possibly cause
# performance trouble when there are a lot of traceroutes on your network.
# Enable cautiously.
#@load misc/detect-traceroute

# Generate notices when vulnerable versions of software are discovered.
# The default is to only monitor software found in the address space defined
# as "local".  Refer to the software framework's documentation for more
# information.
@load frameworks/software/vulnerable

# Detect software changing (e.g. attacker installing hacked SSHD).
@load frameworks/software/version-changes

# This adds signatures to detect cleartext forward and reverse windows shells.
```

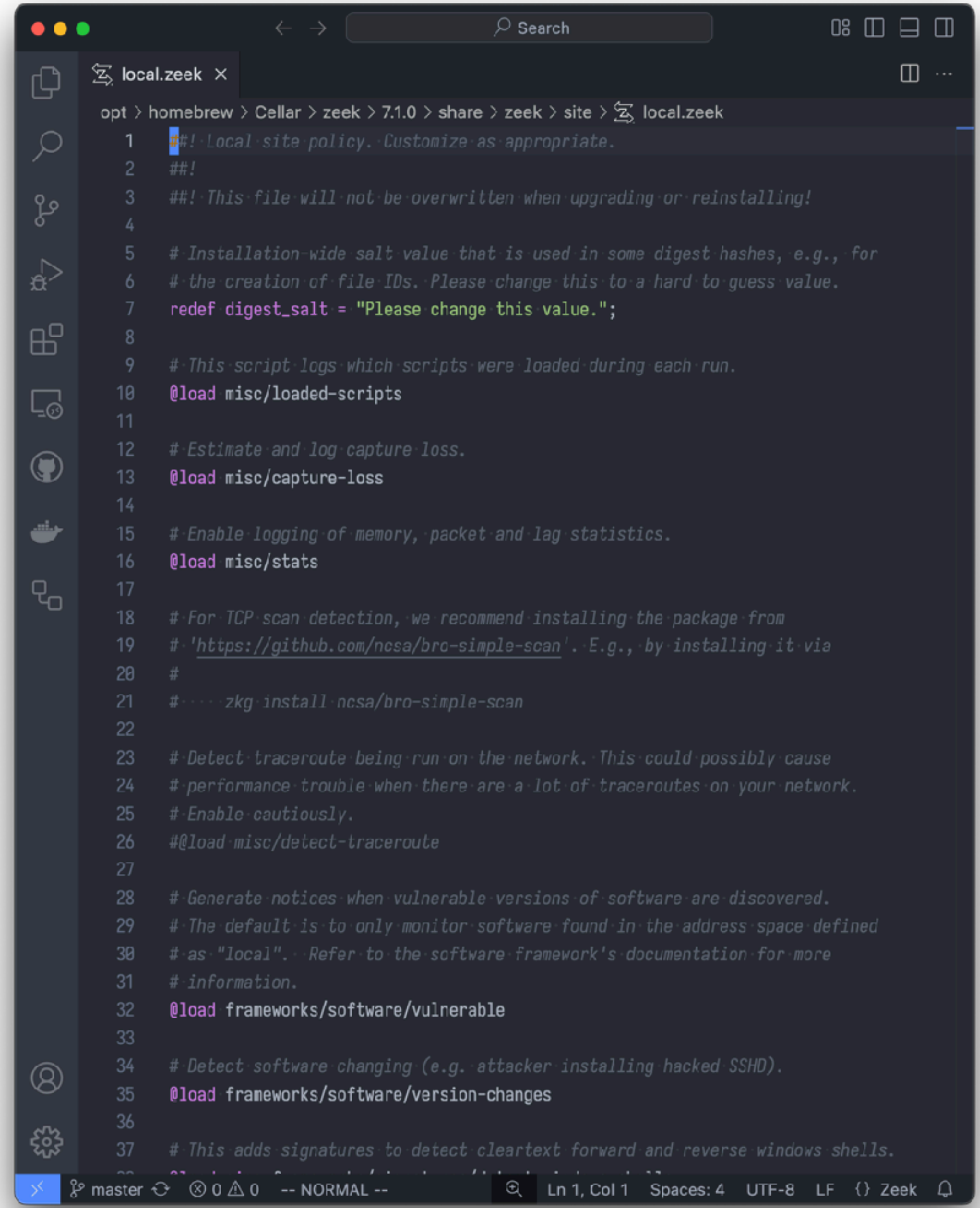
master 0 0 -- NORMAL -- Ln 1, Col 1 Spaces: 4 UTF-8 LF {} Zeek

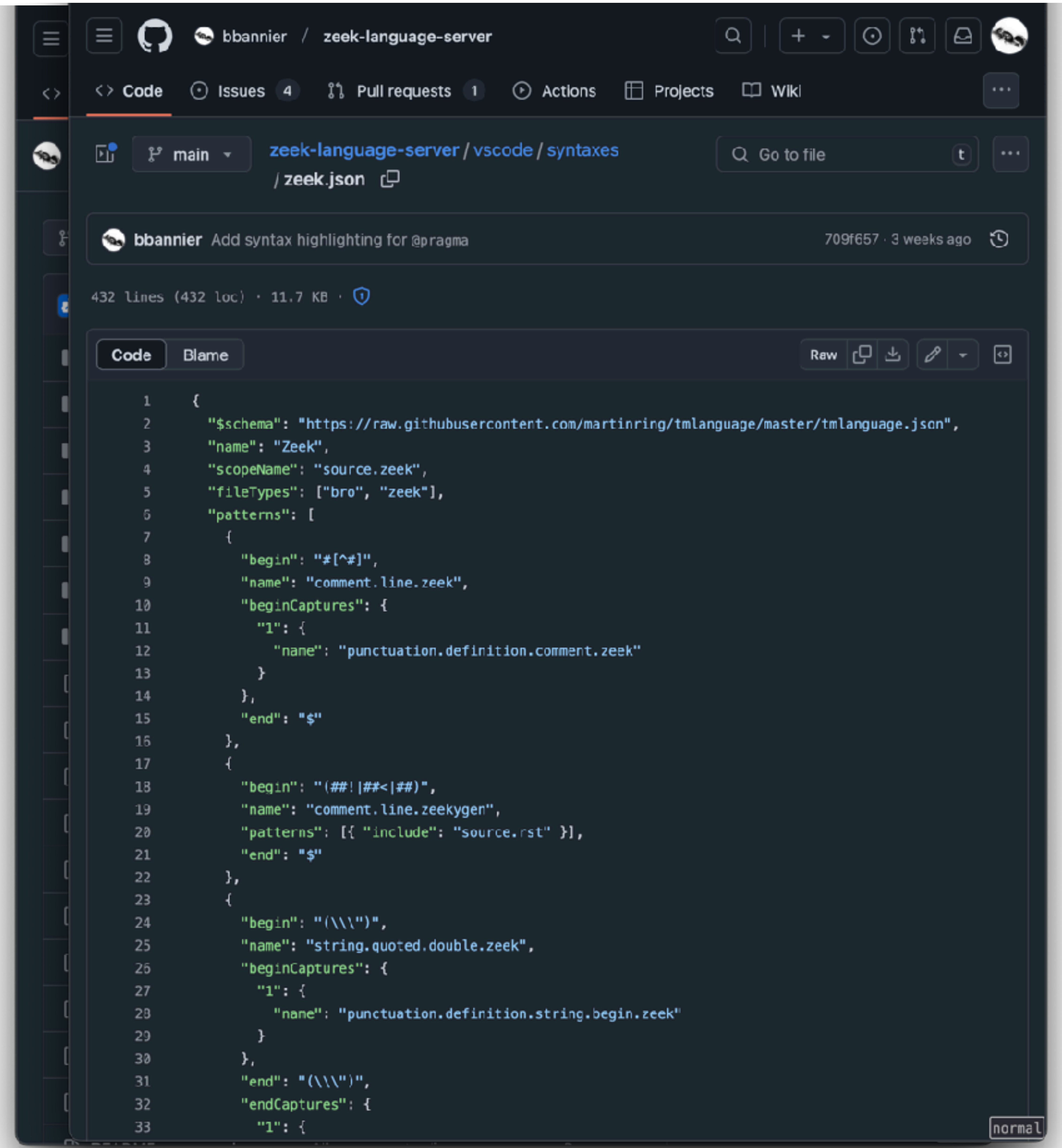
http://github.com/bbannier/zeek-language-server



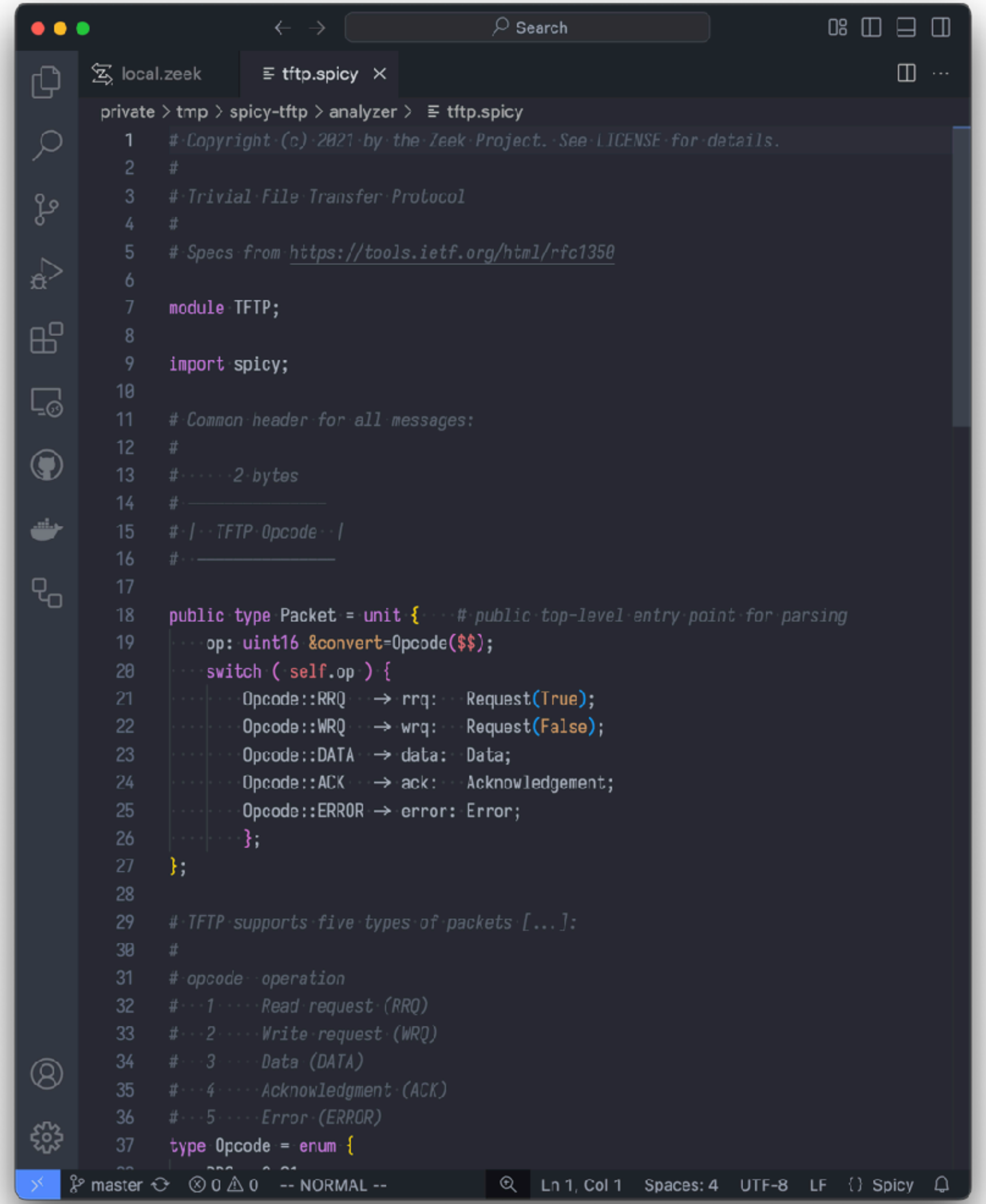


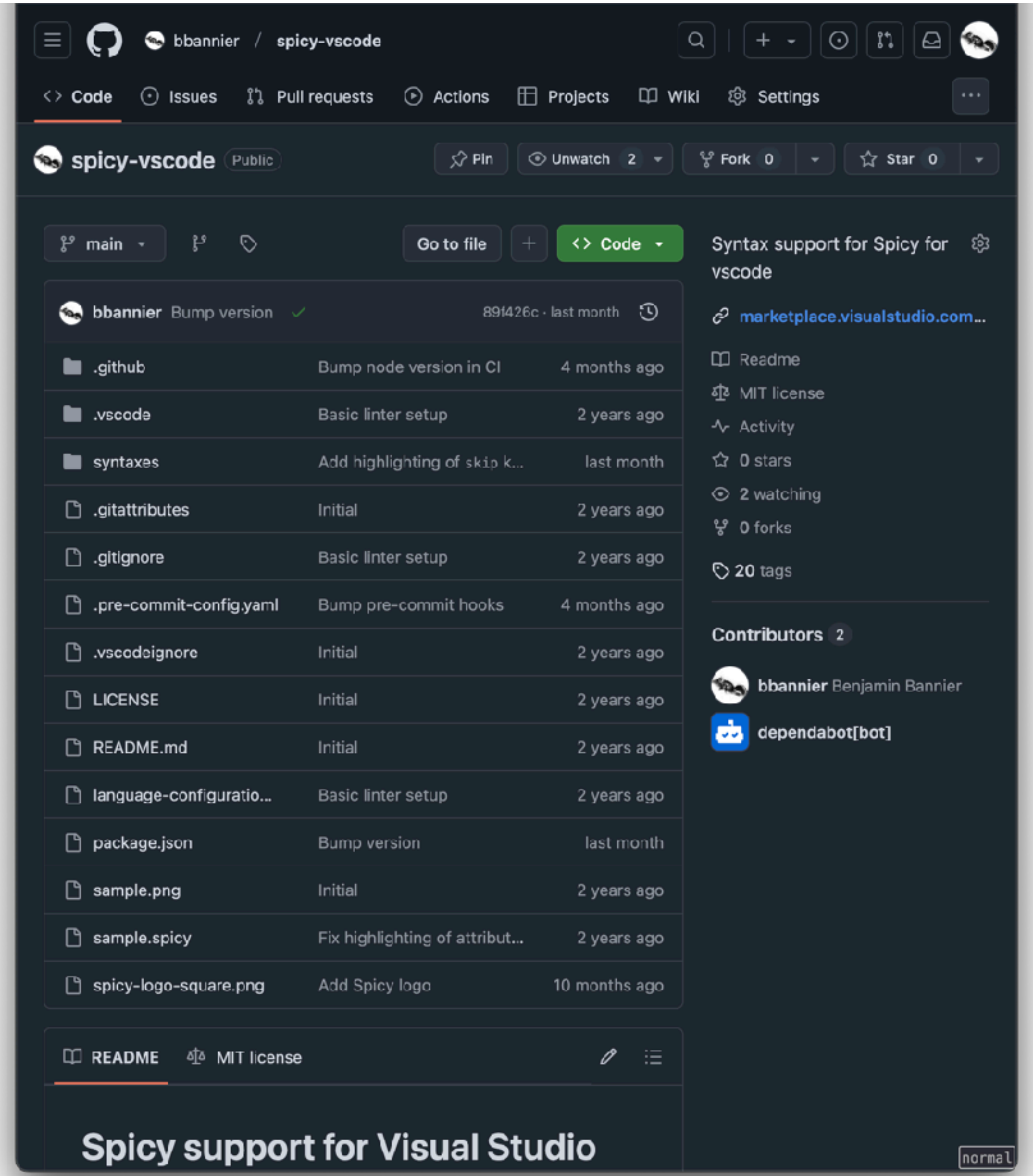
http://github.com/bbannier/zeek-language-server



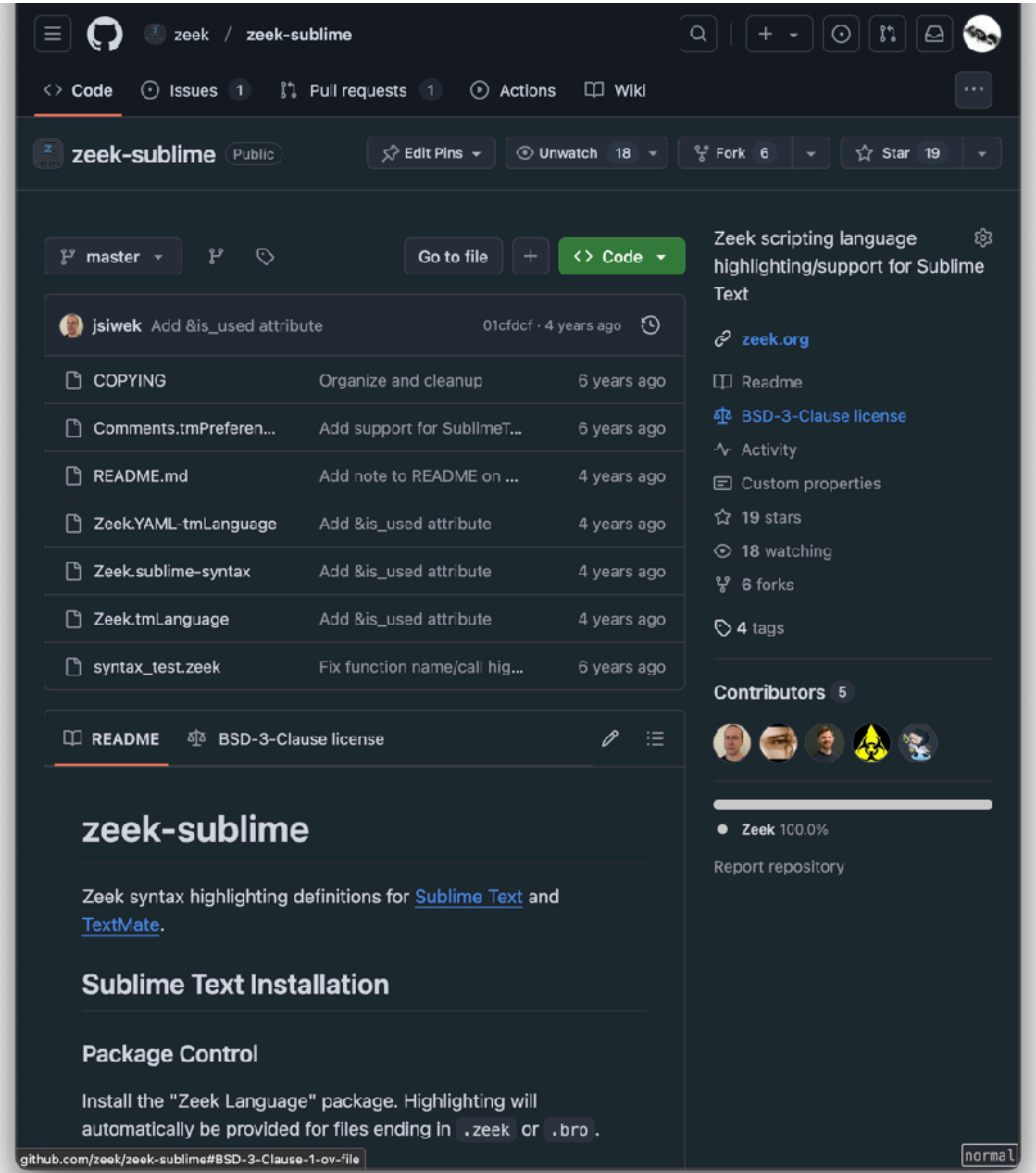


http://github.com/bbannier/spicy-vscode

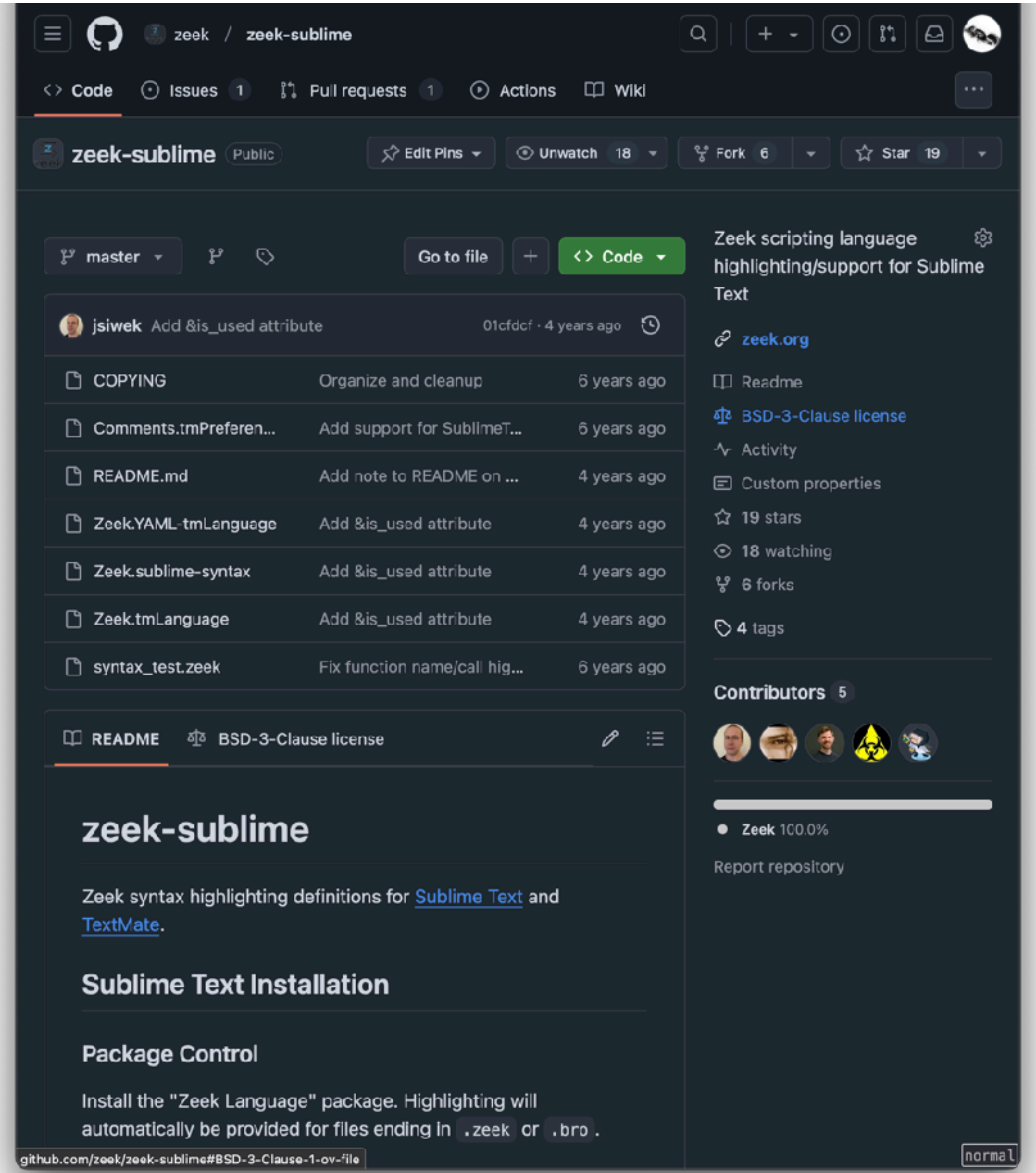


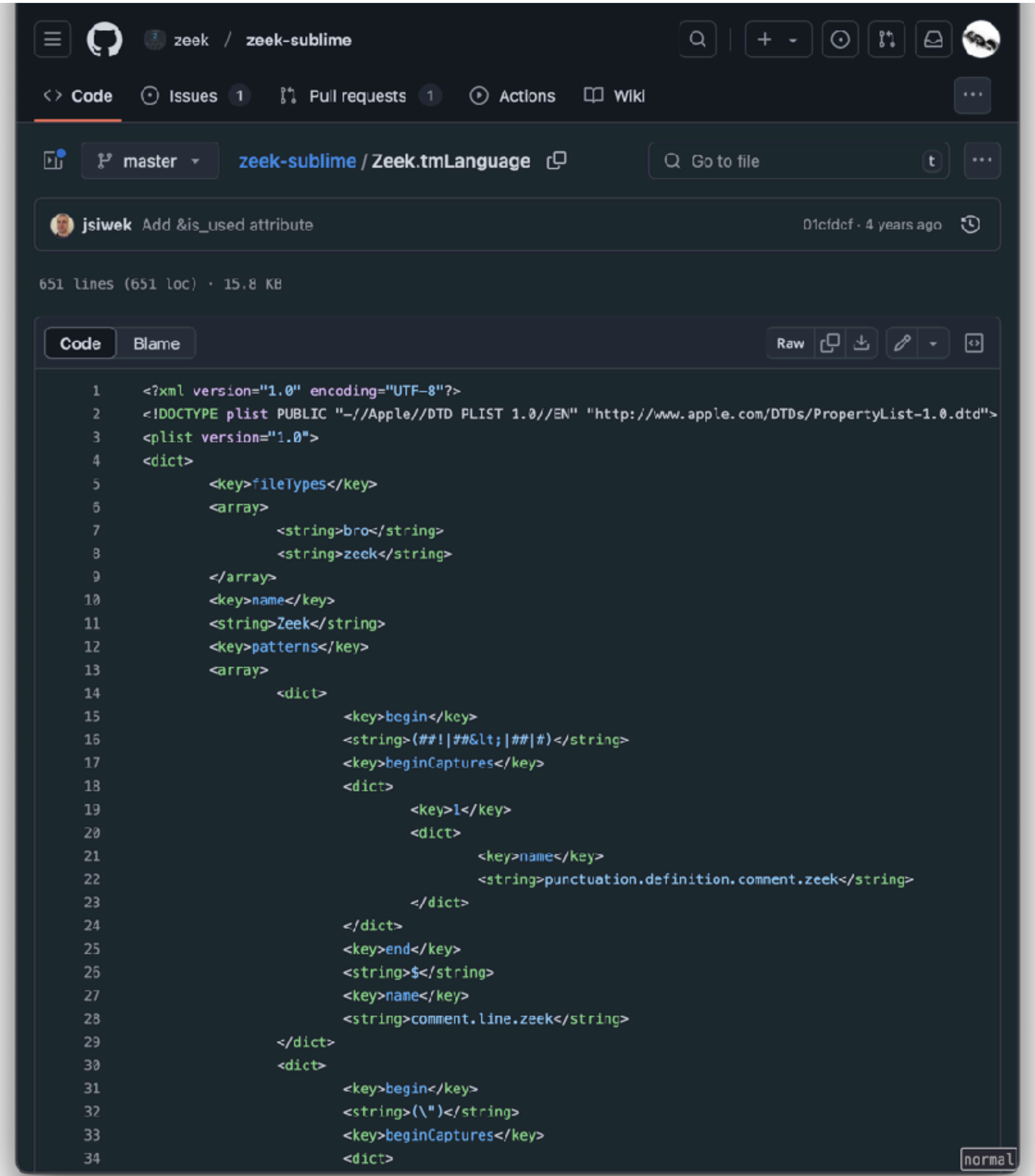


http://github.com/zeek/zeek-sublime

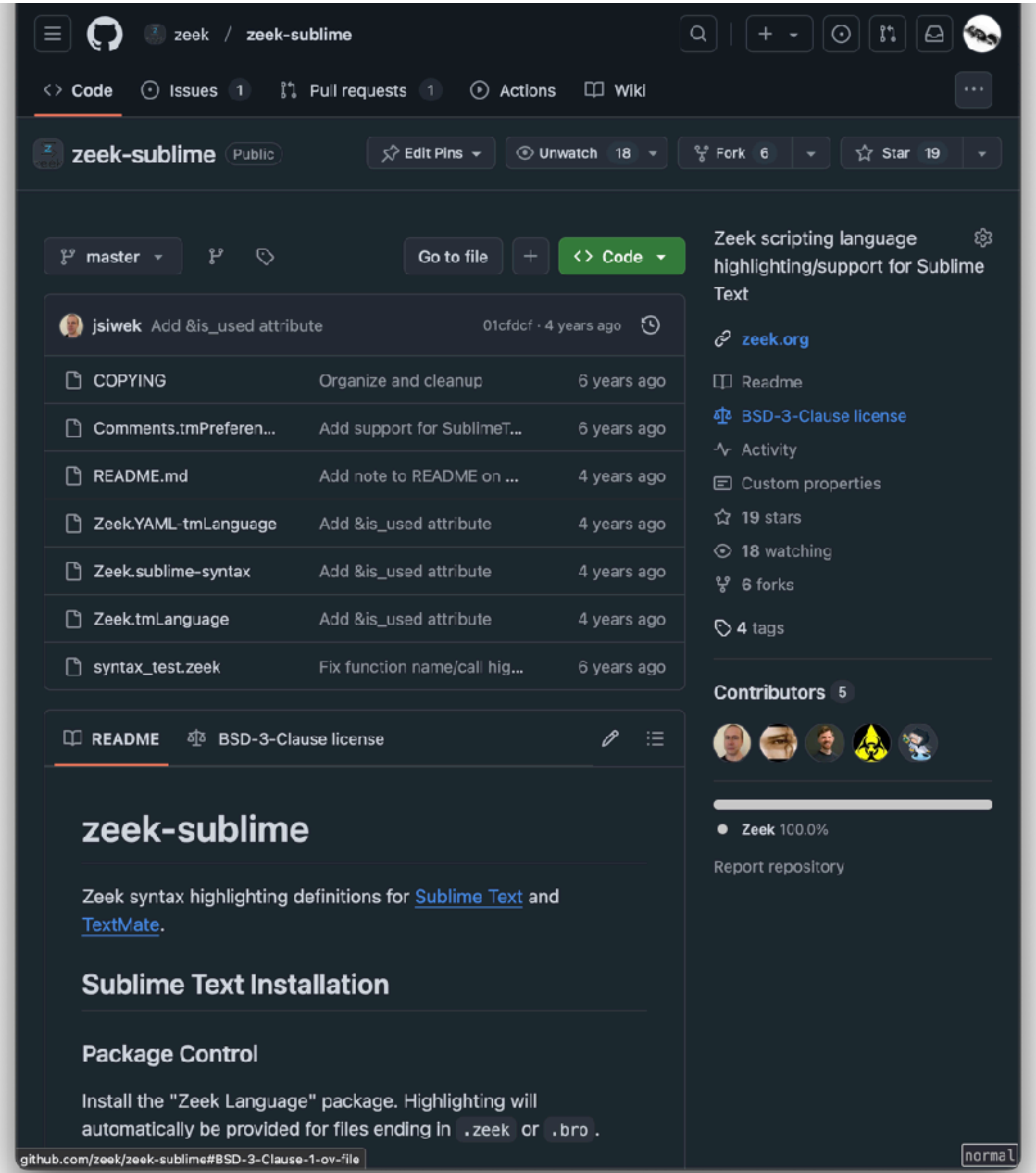


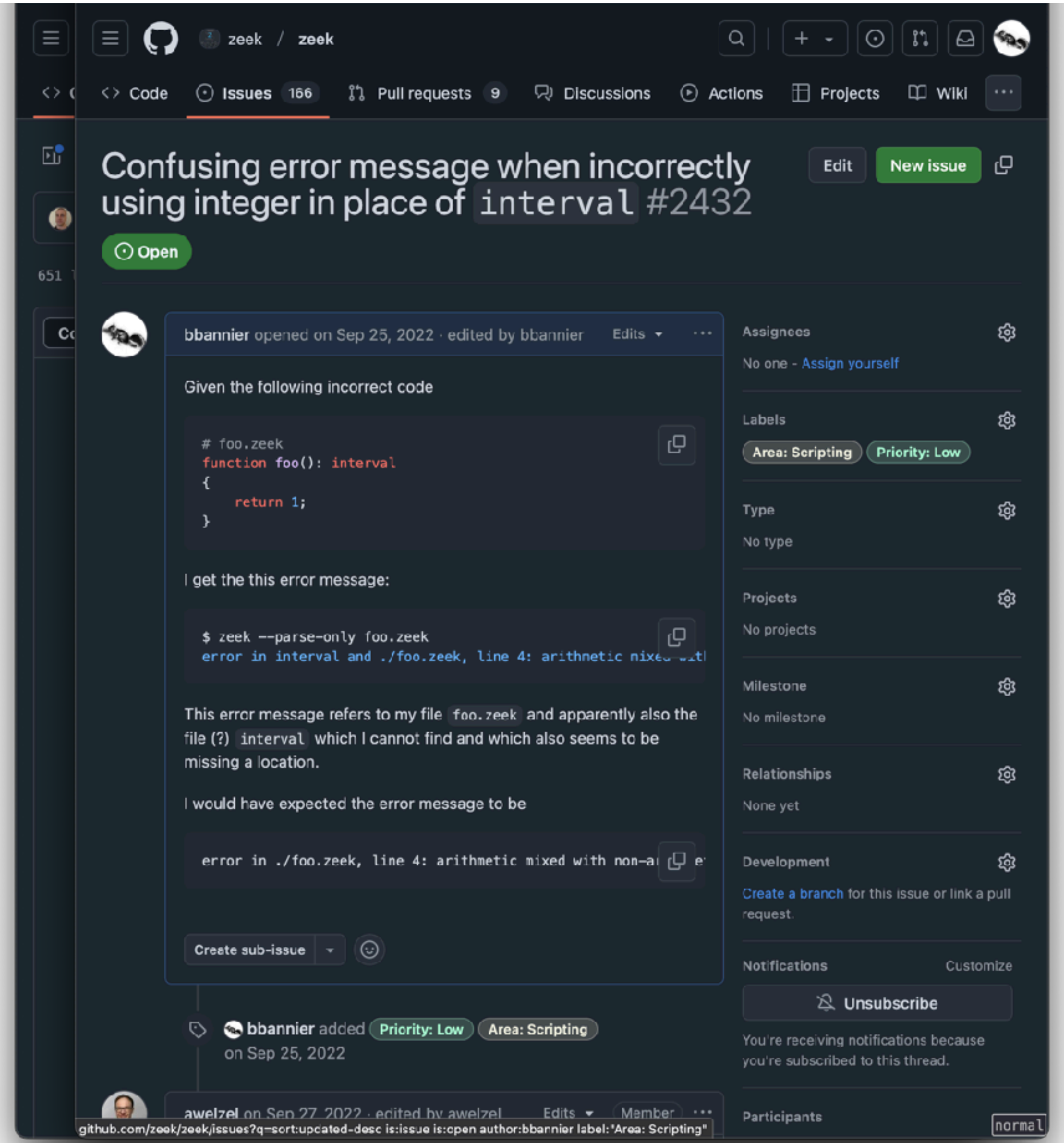
http://github.com/zeek/zeek-sublime



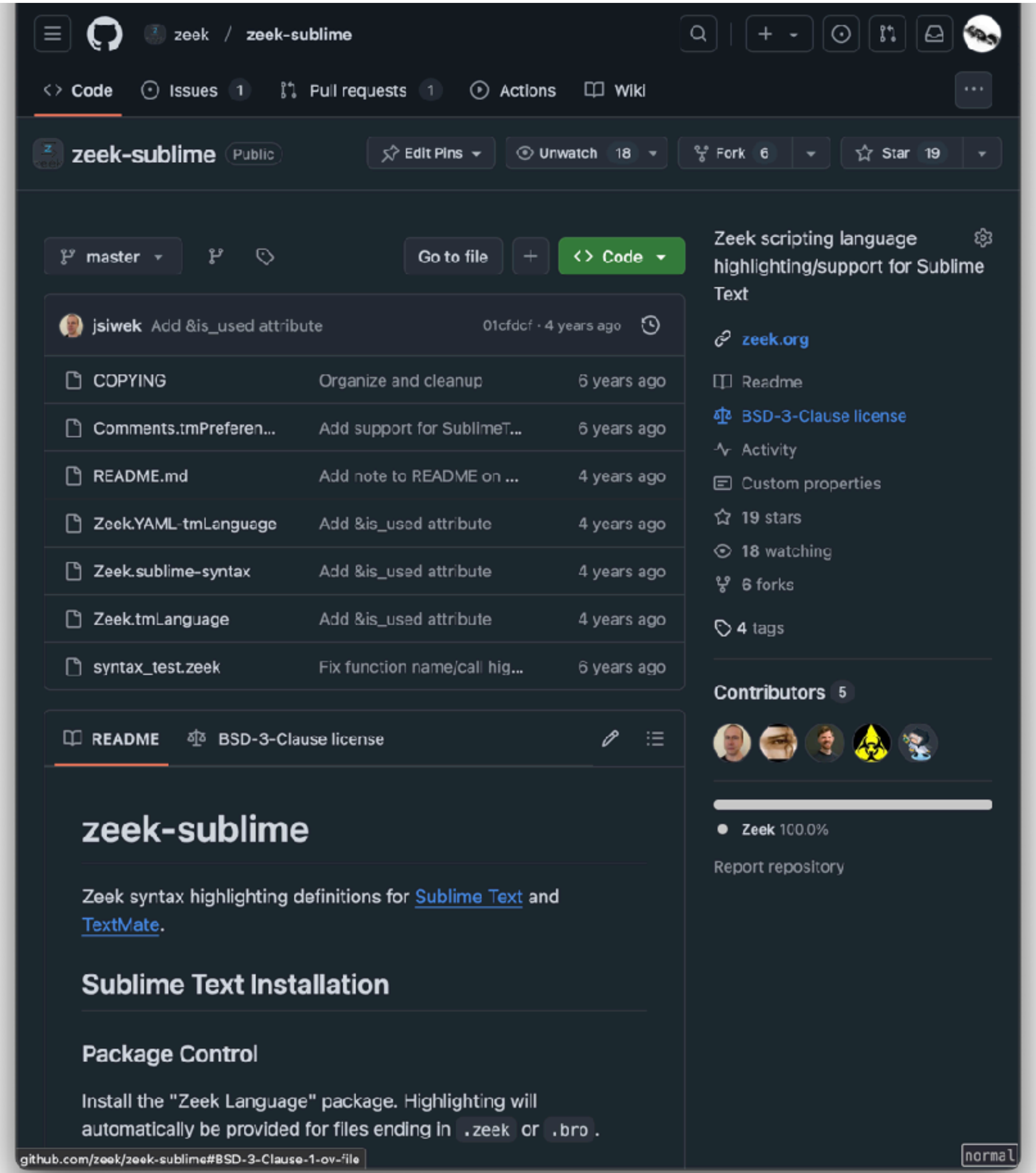


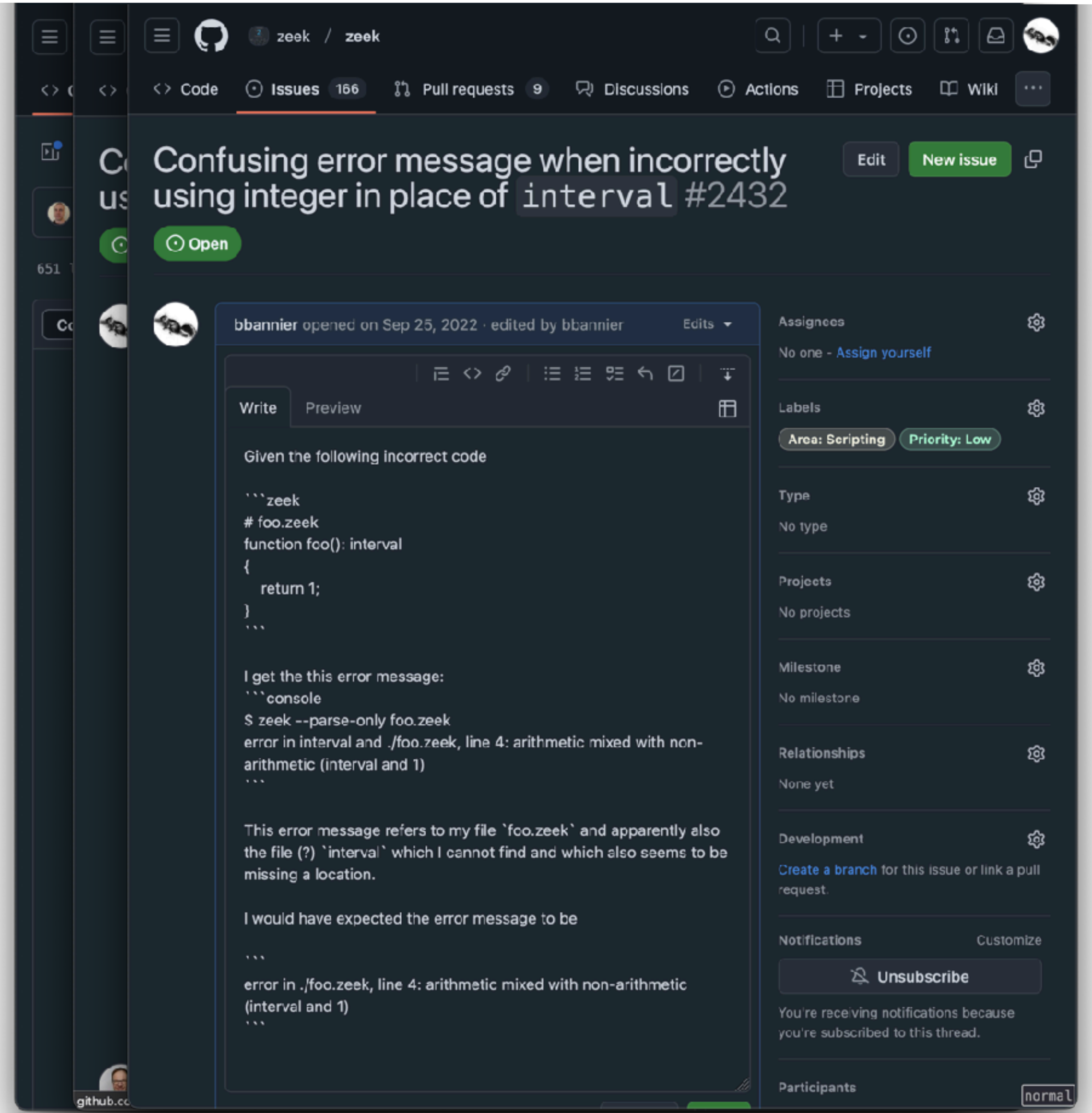
http://github.com/zeek/zeek-sublime



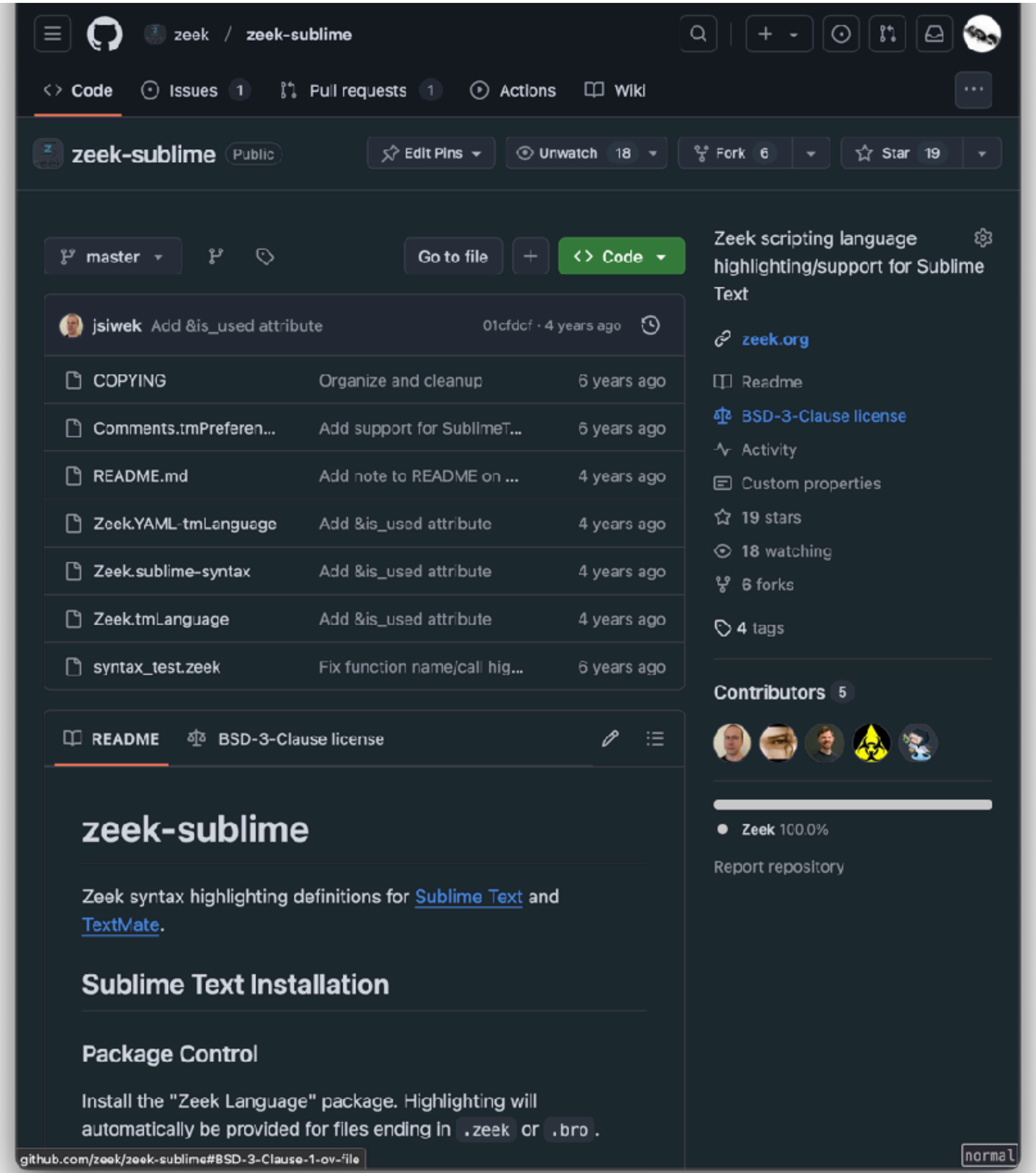


http://github.com/zeek/zeek-sublime

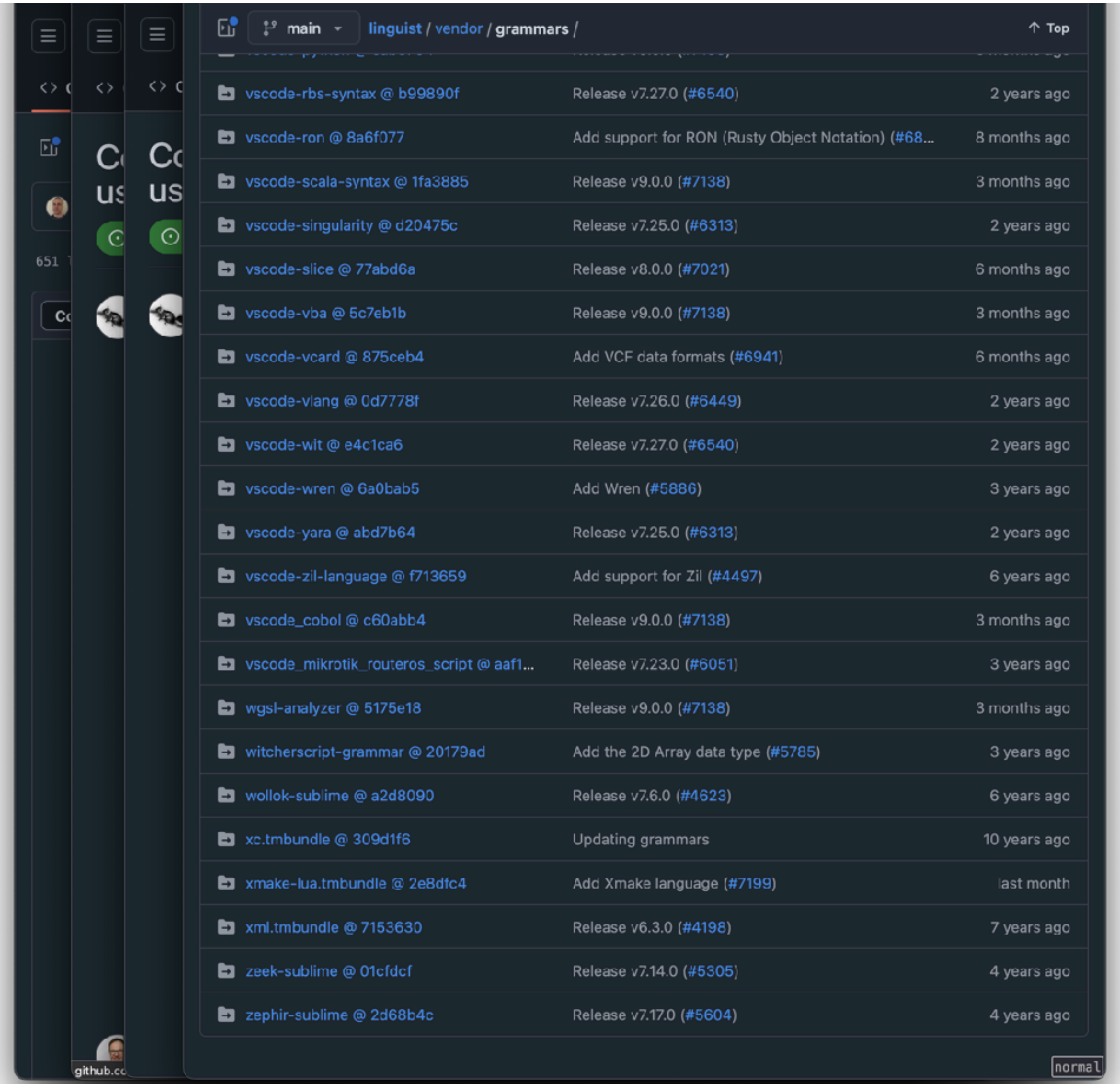




http://github.com/zeek/zeek-sublime



https://github.com/github-linguist/linguist

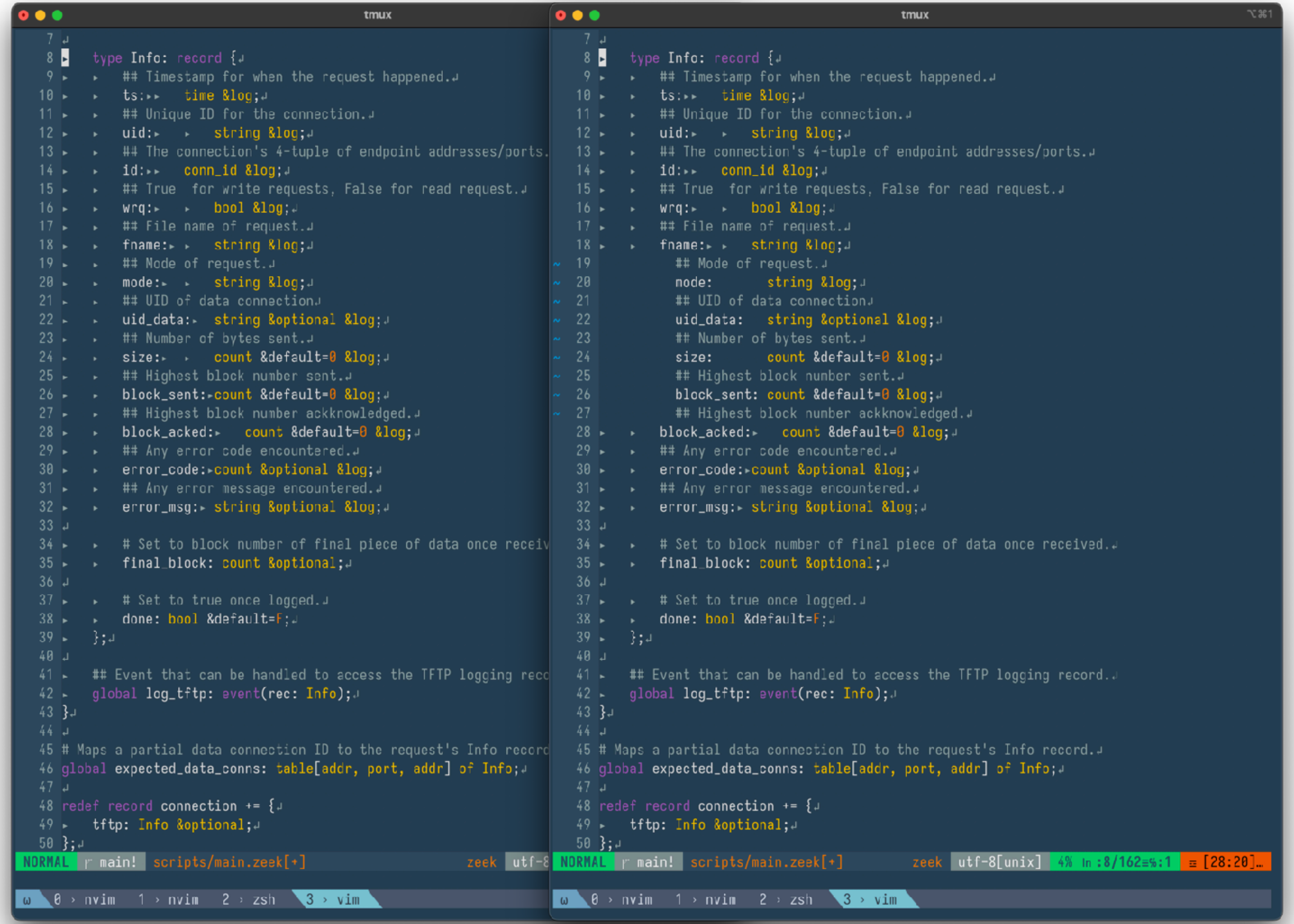


tmux

```
	type Info: record {
		## Timestamp for when the request happened.
		ts:		time &log;
		## Unique ID for the connection.
		uid:		string &log;
		## The connection's 4-tuple of endpoint addresses/ports.
		id:		conn_id &log;
		## True  for write requests, False for read request.
		wrq:		bool &log;
		## File name of request.
		fname:		string &log;
		## Mode of request.
		mode:		string &log;
		## UID of data connection
		uid_data:	string &optional &log;
		## Number of bytes sent.
		size:		count &default=0 &log;
		## Highest block number sent.
		block_sent:	count &default=0 &log;
		## Highest block number acknowledged.
		block_acked:	count &default=0 &log;
		## Any error code encountered.
		error_code:	count &optional &log;
		## Any error message encountered.
		error_msg:	string &optional &log;

		# Set to block number of final piece of data once received.
		final_block: count &optional;

		# Set to true once logged.
		done: bool &default=F;
	};

	## Event that can be handled to access the TFTP logging record.
	global log_tftp: event(rec: Info);
}

# Maps a partial data connection ID to the request's Info record.
global expected_data_conns: table[addr, port, addr] of Info;

redef record connection += {
	tftp: Info &optional;
};
```

NORMAL | main! | scripts/main.zeek[+] | zeek | utf-8[unix] | 4% ln :8/162≡%:1

ω | 0 › nvim | 1 › nvim | 2 › zsh | 3 › vim


tmux
type Info: record {
Timestamp for when the request happened.
ts: time &log;
Unique ID for the connection.
uid: string &log;
The connection's 4-tuple of endpoint addresses/ports.
id: conn_id &log;
True for write requests, False for read request.
wrq: bool &log;
File name of request.
fname: string &log;
Mode of request.
mode: string &log;
UID of data connection
uid_data: string &optional &log;
Number of bytes sent.
size: count &default=0 &log;
Highest block number sent.
block_sent: count &default=0 &log;
Highest block number acknowledged.
block_acked: count &default=0 &log;
Any error code encountered.
error_code: count &optional &log;
Any error message encountered.
error_msg: string &optional &log;
Set to block number of final piece of data once receiv
final_block: count &optional;
# Set to true once logged.
done: bool &default=F;
};
Event that can be handled to access the TFTP logging reco
global log_tftp: event(rec: Info);
}
# Maps a partial data connection ID to the request's Info record
global expected_data_conns: table[addr, port, addr] of Info;
redef record connection += {
tftp: Info &optional;
};
NORMAL
main!
scripts/main.zeek[+]
zeek
utf-8
ω
0 › nvim
1 › nvim
2 › zsh
3 › vim
tmux
type Info: record {
Timestamp for when the request happened.
ts: time &log;
Unique ID for the connection.
uid: string &log;
The connection's 4-tuple of endpoint addresses/ports.
id: conn_id &log;
True for write requests, False for read request.
wrq: bool &log;
File name of request.
fname: string &log;
Mode of request.
mode: string &log;
UID of data connection
uid_data: string &optional &log;
Number of bytes sent.
size: count &default=0 &log;
Highest block number sent.
block_sent: count &default=0 &log;
Highest block number acknowledged.
block_acked: count &default=0 &log;
Any error code encountered.
error_code: count &optional &log;
Any error message encountered.
error_msg: string &optional &log;
Set to block number of final piece of data once received.
final_block: count &optional;
# Set to true once logged.
done: bool &default=F;
};
Event that can be handled to access the TFTP logging record.
global log_tftp: event(rec: Info);
}
# Maps a partial data connection ID to the request's Info record.
global expected_data_conns: table[addr, port, addr] of Info;
redef record connection += {
tftp: Info &optional;
};
NORMAL
main!
scripts/main.zeek[+]
zeek
utf-8[unix]
4% ln :8/162=%:1
[28:20]
ω
0 › nvim
1 › nvim
2 › zsh
3 › vim


tmux
type Info: record {
Timestamp for when the request happened.
ts: time &log;
Unique ID for the connection.
uid: string &log;
The connection's 4-tuple of endpoint addresses/ports.
id: conn_id &log;
True for write requests, False for read request.
wrq: bool &log;
File name of request.
fname: string &log;
Mode of request.
mode: string &log;
UID of data connection
uid_data: string &optional &log;
Number of bytes sent.
size: count &default=0 &log;
Highest block number sent.
block_sent: count &default=0 &log;
Highest block number acknowledged.
block_acked: count &default=0 &log;
Any error code encountered.
error_code: count &optional &log;
Any error message encountered.
error_msg: string &optional &log;
Set to block number of final piece of data once receiv
final_block: count &optional;
# Set to true once logged.
done: bool &default=F;
};
Event that can be handled to access the TFTP logging reco
global log_tftp: event(rec: Info);
}
# Maps a partial data connection ID to the request's Info record
global expected_data_conns: table[addr, port, addr] of Info;
redef record connection += {
tftp: Info &optional;
};
NORMAL main! scripts/main.zeek[+] zeek utf-8
0 › nvim 1 › nvim 2 › zsh 3 › vim
tmux
type Info: record {
Timestamp for when the request happened.
ts: time &log;
Unique ID for the connection.
uid: string &log;
The connection's 4-tuple of endpoint address
id: conn_id &log;
True for write requests, False for read rec
wrq: bool &log;
File name of request.
fname: string &log;
Mode of request.
mode: string &log;
UID of data connection
uid_data: string &optional &log;
Number of bytes sent.
size: count &default=0 &log;
Highest block number sent.
block_sent: count &default=0 &log;
Highest block number acknowledged.
block_acked: count &default=0 &log;
Any error code encountered.
error_code: count &optional &log;
Any error message encountered.
error_msg: string &optional &log;
Set to block number of final piece of data on
final_block: count &optional;
# Set to true once logged.
done: bool &default=F;
};
Event that can be handled to access the TFTP log
global log_tftp: event(rec: Info);
}
# Maps a partial data connection ID to the request's Ir
global expected_data_conns: table[addr, port, addr] of
redef record connection += {
tftp: Info &optional;
};
NORMAL main! scripts/main.zeek[+] zeek utf-8[ur
0 › nvim 1 › nvim 2 › zsh 3 › vim
tmux
type Info: record {
Timestamp for when the request happened.
ts: time &log;
Unique ID for the connection.
uid: string &log;
The connection's 4-tuple of endpoint addresses/ports.
id: conn_id &log;
True for write requests, False for read request.
wrq: bool &log;
File name of request.
fname: string &log;
Mode of request.
mode: string &log;
UID of data connection
uid_data: string &optional &log;
Number of bytes sent.
size: count &default=0 &log;
Highest block number sent.
block_sent: count &default=0 &log;
Highest block number acknowledged.
block_acked: count &default=0 &log;
Any error code encountered.
error_code: count &optional &log;
Any error message encountered.
error_msg: string &optional &log;
Set to block number of final piece of data once received.
final_block: count &optional;
# Set to true once logged.
done: bool &default=F;
}
;
Event that can be handled to access the TFTP logging record.
global log_tftp: event(rec: Info);
}
# Maps a partial data connection ID to the request's Info record.
global expected_data_conns: table[addr, port, addr] of Info;
NORMAL main! scripts/main.zeek[+] zeek utf-8[unix] 4% ln :8/165≡%:1 ≡ [8]trai…
0 › nvim 1 › nvim 2 › zsh 3 › vim

Formatting

https://github.com/zeek/zeekscript/

```
❯ zeek-format -h
usage: zeek-format [-h] [--version] [--inplace] [--recursive] [FILES ...]

A Zeek script formatter

positional arguments:
  FILES             Zeek script(s) to process. Use "-" to specify stdin as a filename.
                    Omitting filenames entirely implies reading from stdin.

options:
  -h, --help        show this help message and exit
  --version, -v     show version and exit
  --inplace, -i     change provided files instead of writing to stdout
  --recursive, -r   process *.zeek files recursively when provided directories instead of
                    files. Requires --inplace.
```

https://github.com/zeek/zeekscript/

```
❯ zeek-format -h
usage: zeek-format [-h] [--version] [--inplace] [--recursive] [FILES ...]

A Zeek script formatter

positional arguments:
  FILES              Zeek script(s) to process. Use "-" to specify stdin as a filename.
                     Omitting filenames entirely implies reading from stdin.

options:
  -h, --help         show this help message and exit
  --version, -v      show version and exit
  --inplace, -i      change provided files instead of writing to stdout
  --recursive, -r    process *.zeek files recursively when provided directories instead of
                     files. Requires --inplace.
```

```
❯ pipx install zeekscript
  installed package zeekscript 1.3.1, installed using Python 3.13.2
  These apps are now globally available
    - zeek-format
    - zeek-script
done! ✨ 🌟 ✨
```

https://github.com/zeek/zeekscript/

```
❯ zeek-format -h
usage: zeek-format [-h] [--version] [--inplace] [--recursive] [FILES ...]

A Zeek script formatter

positional arguments:
  FILES              Zeek script(s) to process. Use "-" to specify stdin as a filename.
                     Omitting filenames entirely implies reading from stdin.

options:
  -h, --help         show this help message and exit
  --version, -v      show version and exit
  --inplace, -i      change provided files instead of writing to stdout
  --recursive, -r    process *.zeek files recursively when provided directories instead of
                     files. Requires --inplace.
```

```
❯ pipx install zeekscript
  installed package zeekscript 1.3.1, installed using Python 3.13.2
  These apps are now globally available
    - zeek-format
    - zeek-script
done! ✨ 🌟 ✨
```

```
❯ echo 'type Foo: record { a: bool; b   : count   ; c:string;};' | zeek-format
type Foo: record {
        a: bool;
        b: count;
        c: string;
};
```

https://github.com/bbannier/spicy-format/

```
❯ spicy-format -h
Usage:

Arguments:
  [INPUT_FILES]...  input files to operate on

Options:
  -s, --skip-idempotence     skip idempotency check
  -r, --reject-parse-errors  reject inputs with parse errors
  -i, --inplace              format file in place
  -h, --help                 Print help (see more with '--help')
  -V, --version              Print version
```

```
❯ curl --proto '=https' --tlsv1.2 -LsSf https://github.com/bbannier/spicy-format/releases/down
load/v0.24.0/spicy-format-installer.sh | sh
downloading spicy-format 0.24.0 aarch64-apple-darwin
installing to /Users/bbannier/.cargo/bin
  spicy-format
  spicy-format-update
everything's installed!
```

```
❯ echo 'type Foo = unit {a: uint8; b: bytes &eod; c:  /re/  ;};' | spicy-format
type Foo = unit {
    a: uint8;
    b: bytes &eod;
    c: /re/;
};
```

http://pre-commit.com/

Documentation Supported hooks Demo Download on GitHub

pre-commit

A framework for managing and maintaining multi-language pre-commit hooks.

main passing pre-commit.ci passed

Star

Introduction

Git hook scripts are useful for identifying simple issues before submission to code review. We run our hooks on every commit to automatically point out issues in code such as missing semicolons, trailing whitespace, and debug statements. By pointing these issues out before code review, this allows a code reviewer to focus on the architecture of a change while not wasting time with trivial style nitpicks.

As we created more libraries and projects we recognized that sharing our pre-commit hooks across projects is painful. We copied and pasted unwieldy bash scripts from project to project and had to manually change the hooks to work for different project structures.

We believe that you should always use the best industry standard linters. Some of the best linters are written in languages that you do not use in your project or have installed on your machine. For example scss-lint is a linter for SCSS written in Ruby. If you're writing a project in node you should be able to use scss-lint as a pre-commit hook without adding a Gemfile to your project or understanding how to get scss-lint installed.

We built pre-commit to solve our hook issues. It is a multi-language package manager for pre-commit hooks. You specify a list of hooks you want and pre-commit manages the installation and execution of any hook written in any language before every commit. pre-commit is specifically designed to not require root access. If one of your developers doesn't have node installed but modifies a JavaScript file, pre-commit automatically handles downloading and building node to run eslint without root.

Installation

http://pre-commit.com/

Documentation Supported hooks Demo Download on GitHub

pre-commit

A framework for managing and maintaining multi-language pre-commit hooks.

main passing pre-commit.ci passed

Star

Introduction

Git hook scripts are useful for identifying simple issues before submission to code review. We run our hooks on every commit to automatically point out issues in code such as missing semicolons, trailing whitespace, and debug statements. By pointing these issues out before code review, this allows a code reviewer to focus on the architecture of a change while not wasting time with trivial style nitpicks.

As we created more libraries and projects we recognized that sharing our pre-commit hooks across projects is painful. We copied and pasted unwieldy bash scripts from project to project and had to manually change the hooks to work for different project structures.

We believe that you should always use the best industry standard linters. Some of the best linters are written in languages that you do not use in your project or have installed on your machine. For example scss-lint is a linter for SCSS written in Ruby. If you're writing a project in node you should be able to use scss-lint as a pre-commit hook without adding a Gemfile to your project or understanding how to get scss-lint installed.

We built pre-commit to solve our hook issues. It is a multi-language package manager for pre-commit hooks. You specify a list of hooks you want and pre-commit manages the installation and execution of any hook written in any language before every commit. pre-commit is specifically designed to not require root access. If one of your developers doesn't have node installed but modifies a JavaScript file, pre-commit automatically handles downloading and building node to run eslint without root.

Installation

```
› bat .pre-commit-config.yaml
File: .pre-commit-config.yaml
repos:

- repo: https://github.com/bbannier/spicy-format
  rev: v0.24.1
  hooks:
  - id: spicy-format

- repo: https://github.com/zeek/zeekscript
  rev: 'v1.3.1'
  hooks:
  - id: zeek-format
```

http://pre-commit.com/

Documentation Supported hooks Demo Download on GitHub

pre-commit

A framework for managing and maintaining multi-language pre-commit hooks.

main passing pre-commit.ci passed

Star

Introduction

Git hook scripts are useful for identifying simple issues before submission to code review. We run our hooks on every commit to automatically point out issues in code such as missing semicolons, trailing whitespace, and debug statements. By pointing these issues out before code review, this allows a code reviewer to focus on the architecture of a change while not wasting time with trivial style nitpicks.

As we created more libraries and projects we recognized that sharing our pre-commit hooks across projects is painful. We copied and pasted unwieldy bash scripts from project to project and had to manually change the hooks to work for different project structures.

We believe that you should always use the best industry standard linters. Some of the best linters are written in languages that you do not use in your project or have installed on your machine. For example scss-lint is a linter for SCSS written in Ruby. If you're writing a project in node you should be able to use scss-lint as a pre-commit hook without adding a Gemfile to your project or understanding how to get scss-lint installed.

We built pre-commit to solve our hook issues. It is a multi-language package manager for pre-commit hooks. You specify a list of hooks you want and pre-commit manages the installation and execution of any hook written in any language before every commit. pre-commit is specifically designed to not require root access. If one of your developers doesn't have node installed but modifies a JavaScript file, pre-commit automatically handles downloading and building node to run eslint without root.

Installation

```
› git add test.spicy test.zeek
› bat *.zeek *.spicy

File: test.zeek

module foo;

export { type Foo: record { a: count; b: string &optional; }; }

File: test.spicy

module foo;

type X = unit { a: uint8;
b: bytes&eod;
c: /re/;
};

› git commit -m'Add scripts'
Format .spicy files.....................................................Failed
- hook id: spicy-format
- files were modified by this hook
zeek-format.............................................................Failed
- hook id: zeek-format
- files were modified by this hook

1 file processed, 0 errors
```

http://pre-commit.com/

Documentation Supported hooks Demo Download on GitHub

pre-commit

A framework for managing and maintaining multi-language pre-commit hooks.

main passing pre-commit.ci passed

Star

Introduction

Git hook scripts are useful for identifying simple issues before submission to code review. We run our hooks on every commit to automatically point out issues in code such as missing semicolons, trailing whitespace, and debug statements. By pointing these issues out before code review, this allows a code reviewer to focus on the architecture of a change while not wasting time with trivial style nitpicks.

As we created more libraries and projects we recognized that sharing our pre-commit hooks across projects is painful. We copied and pasted unwieldy bash scripts from project to project and had to manually change the hooks to work for different project structures.

We believe that you should always use the best industry standard linters. Some of the best linters are written in languages that you do not use in your project or have installed on your machine. For example scss-lint is a linter for SCSS written in Ruby. If you're writing a project in node you should be able to use scss-lint as a pre-commit hook without adding a Gemfile to your project or understanding how to get scss-lint installed.

We built pre-commit to solve our hook issues. It is a multi-language package manager for pre-commit hooks. You specify a list of hooks you want and pre-commit manages the installation and execution of any hook written in any language before every commit. pre-commit is specifically designed to not require root access. If one of your developers doesn't have node installed but modifies a JavaScript file, pre-commit automatically handles downloading and building node to run eslint without root.

Installation

```
❯ git add test.spicy test.zeek
❯ bat *.zeek *.spicy
───────┬──────────────────────────────────────────────
       │ File: test.zeek
───────┼──────────────────────────────────────────────
   1   │ module foo;
   2   │
   3   │ export { type Foo: record { a: count; b: string &optional; }; }
───────┴──────────────────────────────────────────────
───────┬──────────────────────────────────────────────
       │ File: test.spicy
───────┼──────────────────────────────────────────────
   1   │ module foo;
   2   │
diff --git a/test.spicy b/test.spicy
index d8e92b8..00b9c7b 100644
--- a/test.spicy
+++ b/test.spicy
@@ -1,6 +1,7 @@
 module foo;

-type X = unit { a: uint8;
-b: bytes&eod;
-c: /re/;
+type X = unit {
+    a: uint8;
+    b: bytes &eod;
+    c: /re/;
 };
diff --git a/test.zeek b/test.zeek
index 59aff2b..9cbe4f7 100644
--- a/test.zeek
+++ b/test.zeek
@@ -1,3 +1,8 @@
 module foo;

-export { type Foo: record { a: count; b: string &optional; }; }
+export {
+	type Foo: record {
+		a: count;
+		b: string &optional;
+	};
+}
(END)
```

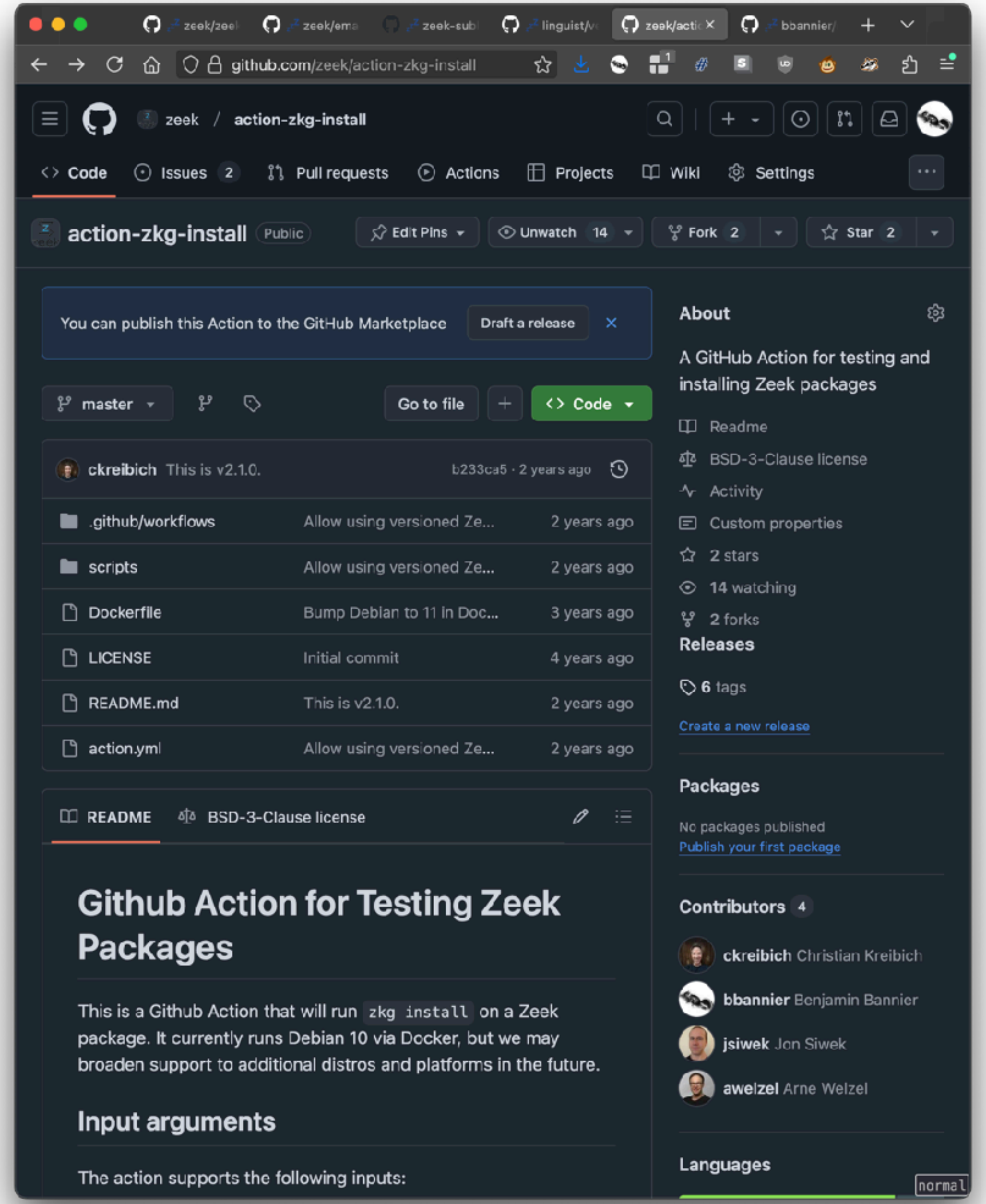
```
❯ bat .github/workflows/pre-commit.yml
File: .github/workflows/pre-commit.yml
name: pre-commit

on:
  pull_request:
  push:
    branches: [main]

jobs:
  pre-commit:
    runs-on: ubuntu-latest
    steps:
    - uses: actions/checkout@v4
    - uses: actions/setup-python@v5
    - uses: pre-commit/action@v3.0.1
```

```
❯ bat .github/workflows/zkg-install.yml
File: .github/workflows/zkg-install.yml
name: zkg-install

on:
  pull_request:
  push:

jobs:
  zkg-install:
    runs-on: ubuntu-latest
    strategy:
      matrix:
        zeek-version:
          - "zeek"
          - "zeek-lts"
    steps:
    - uses: actions/checkout@v4
    - uses: actions/setup-python@v5
    - uses: zeek/action-zkg-install@v2
      with:
        load_packages: true
        zeek_version: ${{ matrix.zeek-version }}
```

https://github.com/zeek/action-zkg-install

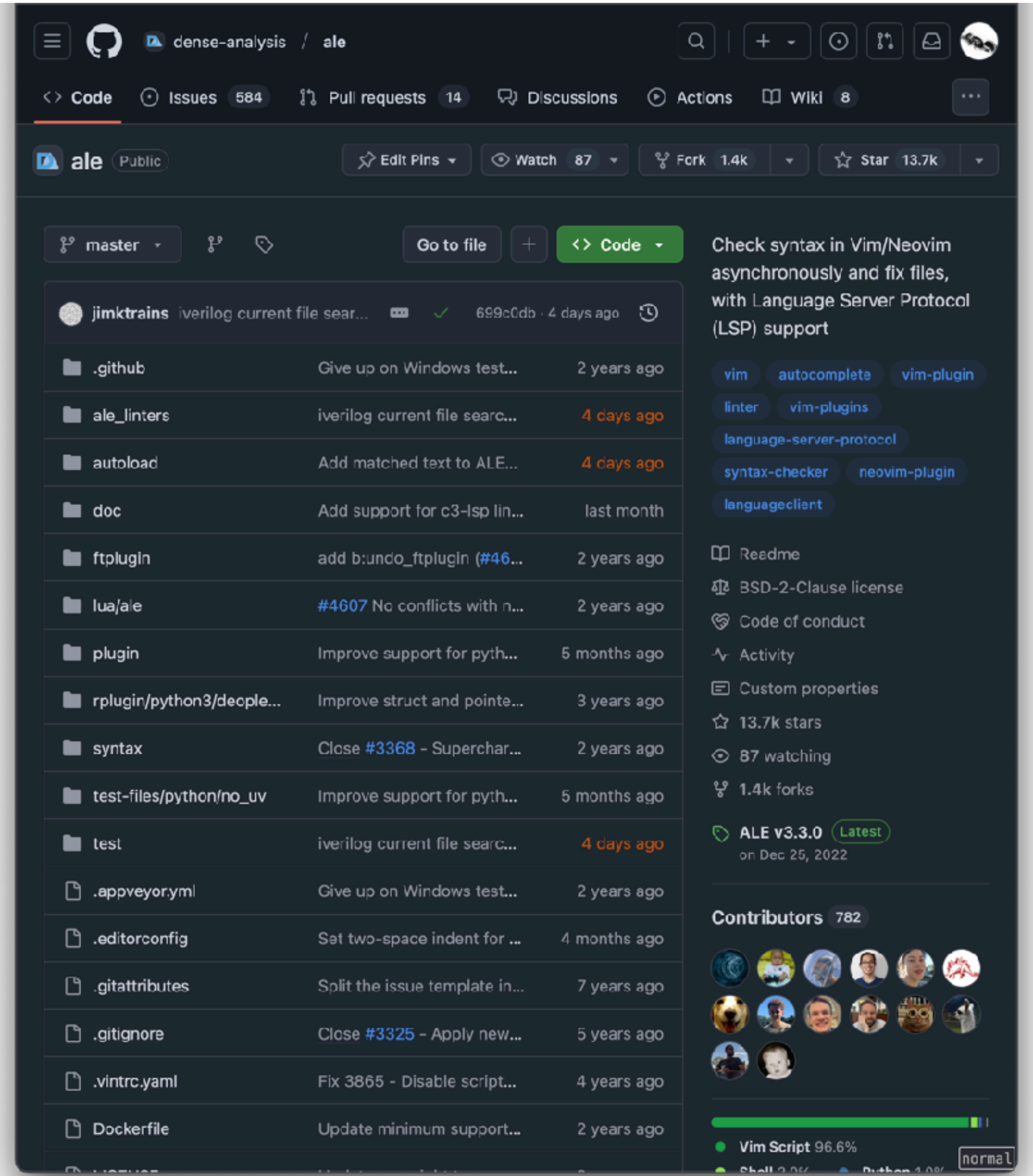
```
❯ bat .github/workflows/pre-commit.yml

File: .github/workflows/pre-commit.yml

name: pre-commit

on:
  pull_request:
  push:
    branches: [main]

jobs:
  pre-commit:
    runs-on: ubuntu-latest
    steps:
    - uses: actions/checkout@v4
    - uses: actions/setup-python@v5
    - uses: pre-commit/action@v3.0.1
```

```
❯ bat .github/workflows/zkg-install.yml

File: .github/workflows/zkg-install.yml

name: zkg-install

on:
  pull_request:
  push:

jobs:
  zkg-install:
    runs-on: ubuntu-latest
    strategy:
      matrix:
        zeek-version:
          - "zeek"
          - "zeek-lts"
    steps:
    - uses: actions/checkout@v4
    - uses: actions/setup-python@v5
    - uses: zeek/action-zkg-install@v2
      with:
        load_packages: true
        zeek_version: ${{ matrix.zeek-version }}
```

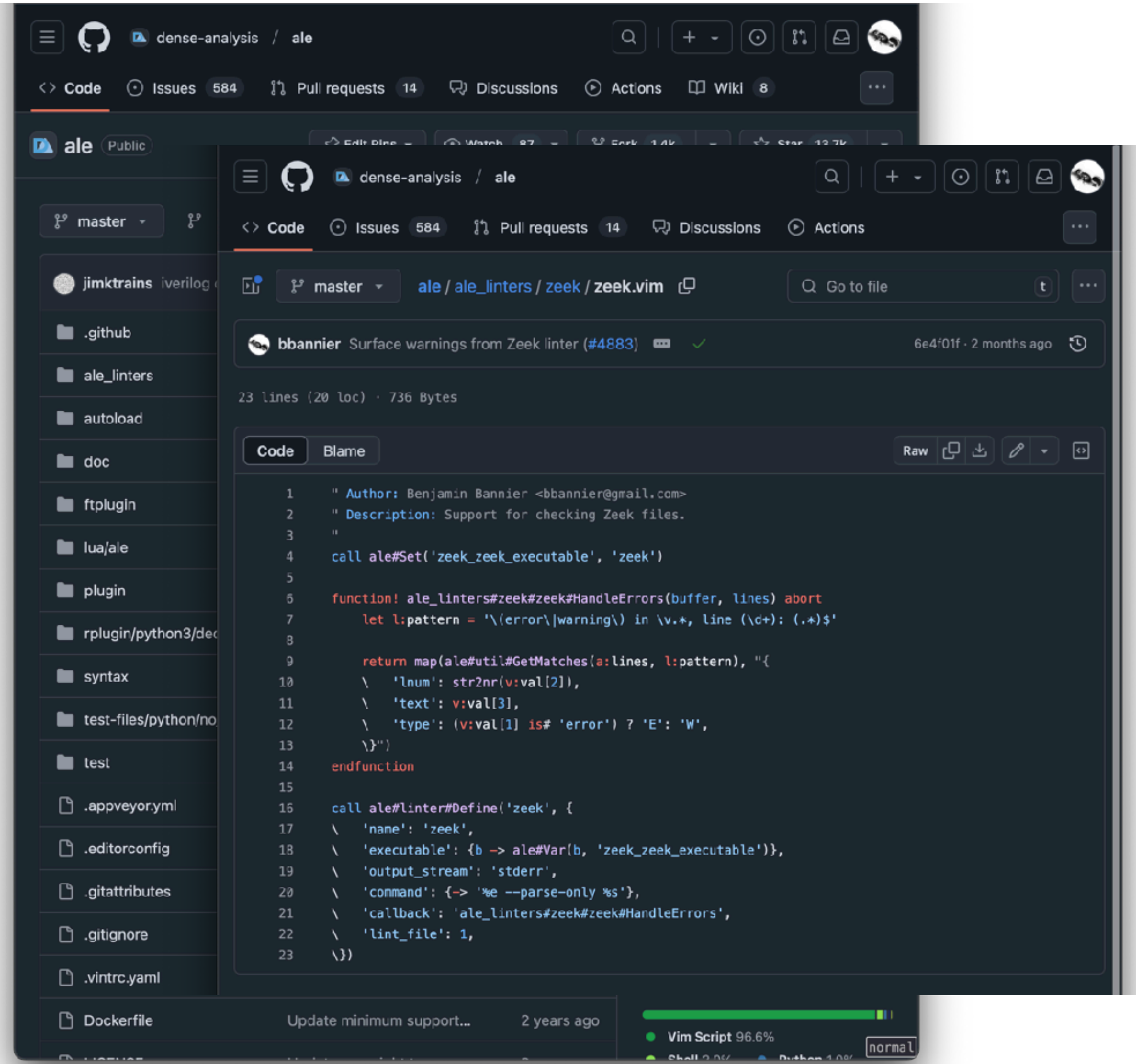


Linting

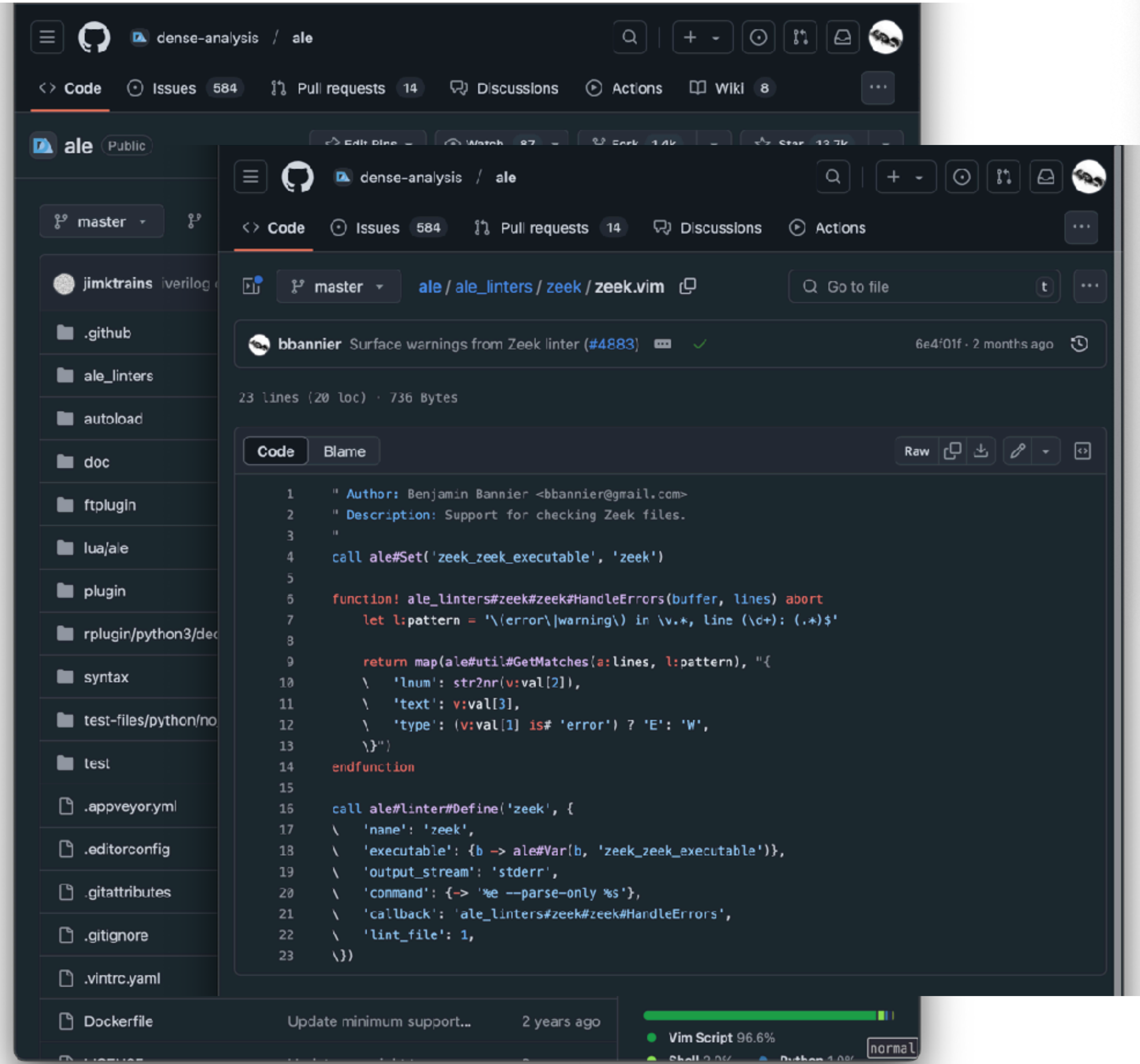
https://github.com/dense-analysis/ale

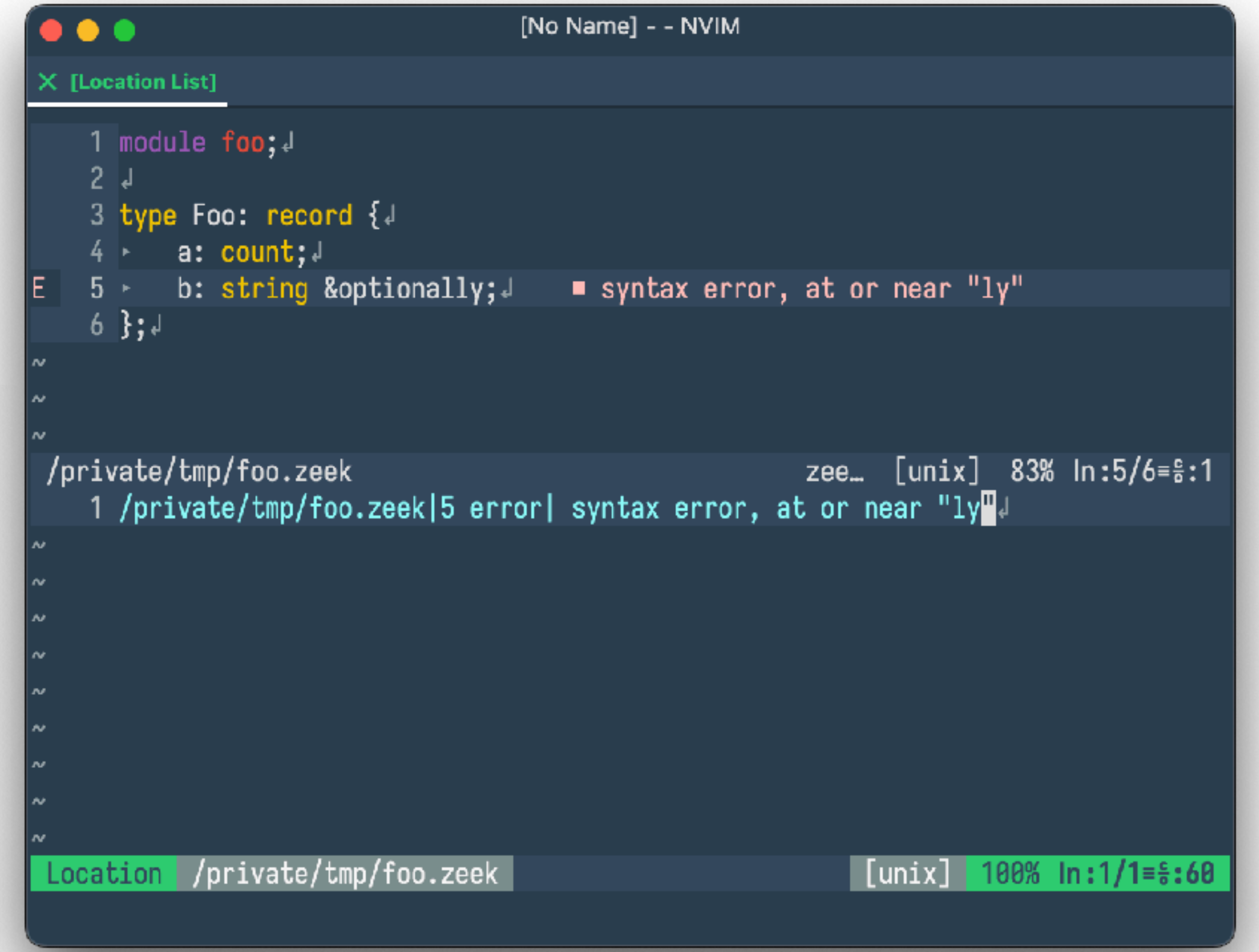


https://github.com/dense-analysis/ale

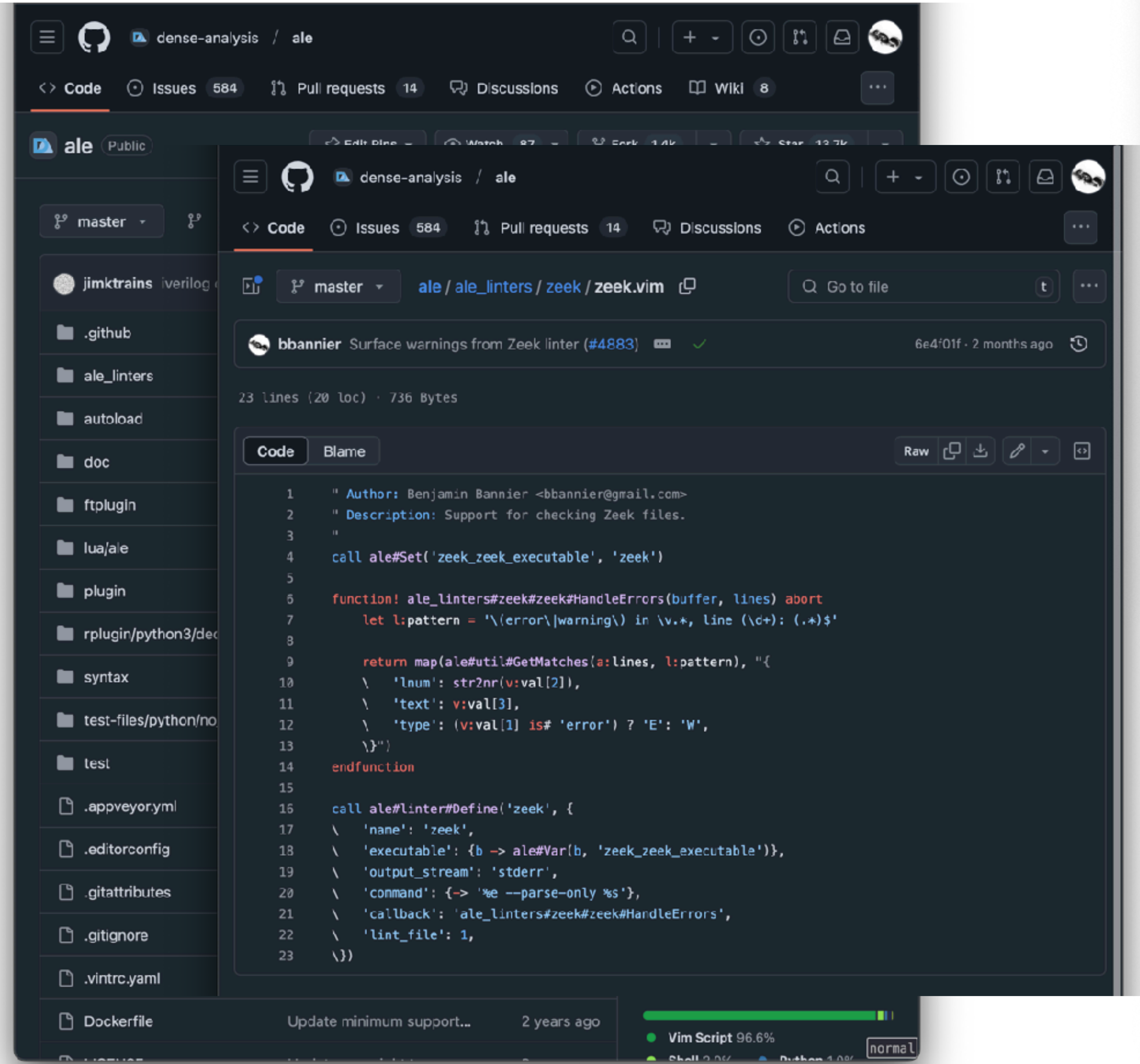


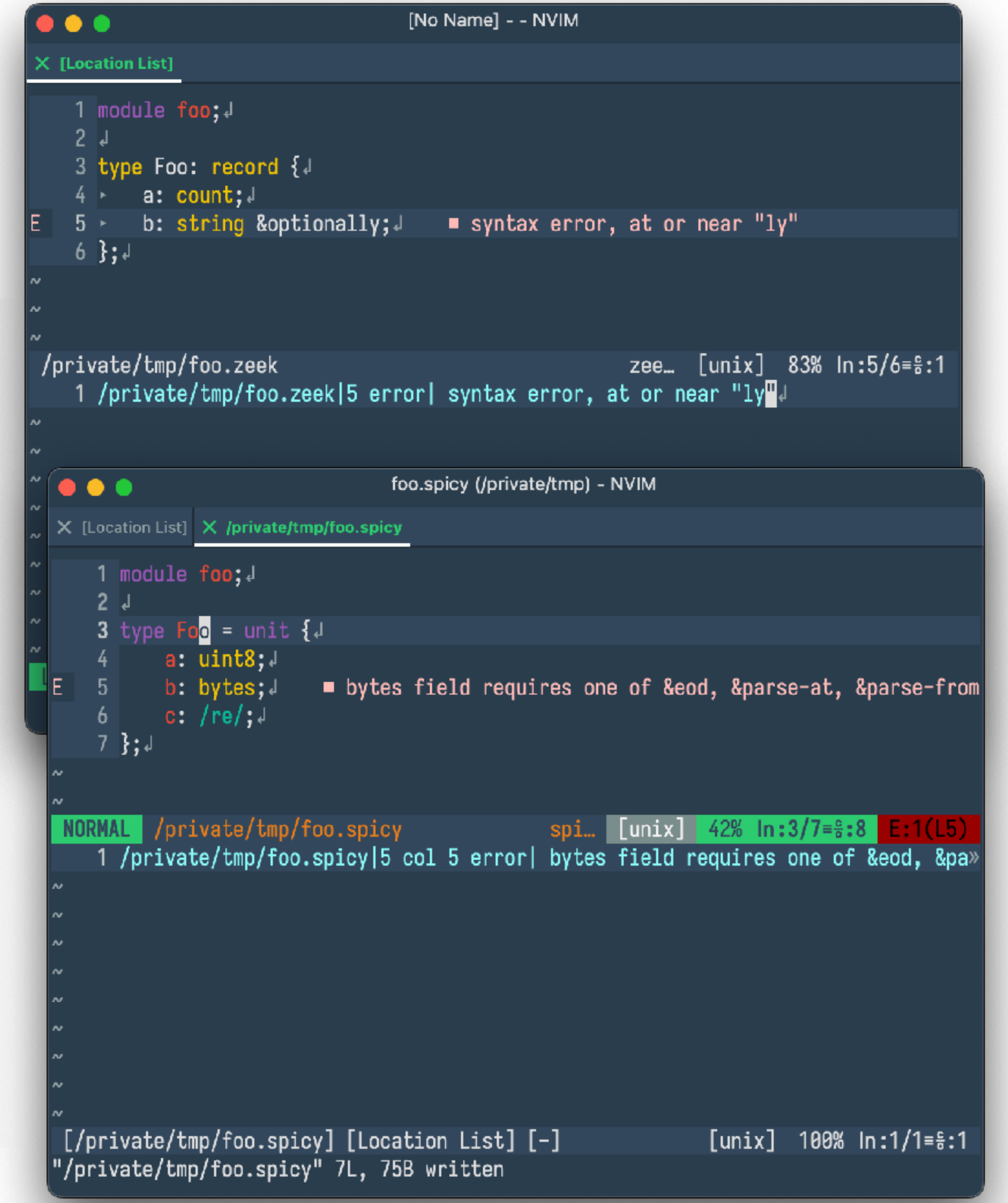
https://github.com/dense-analysis/ale





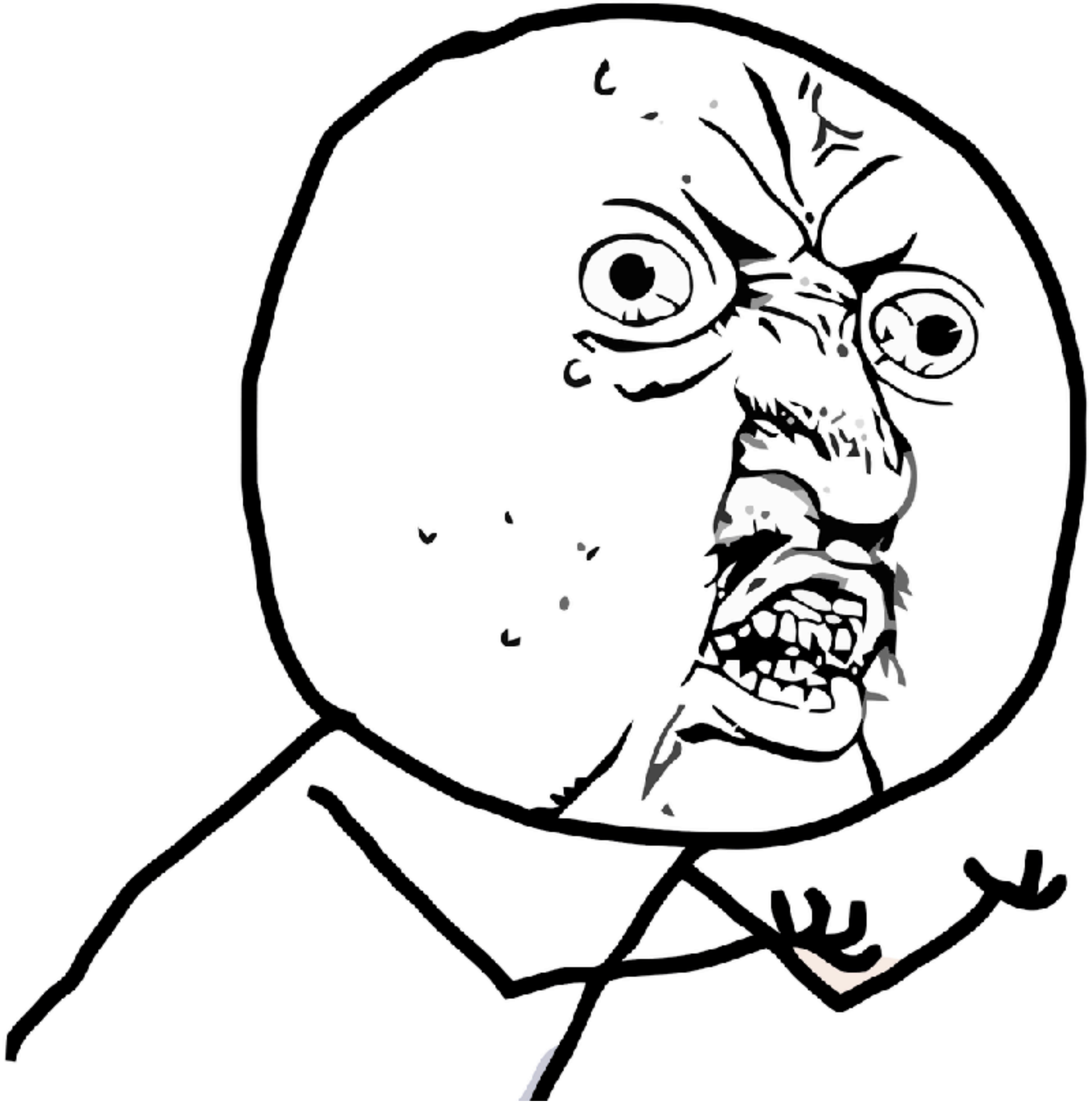
https://github.com/dense-analysis/ale



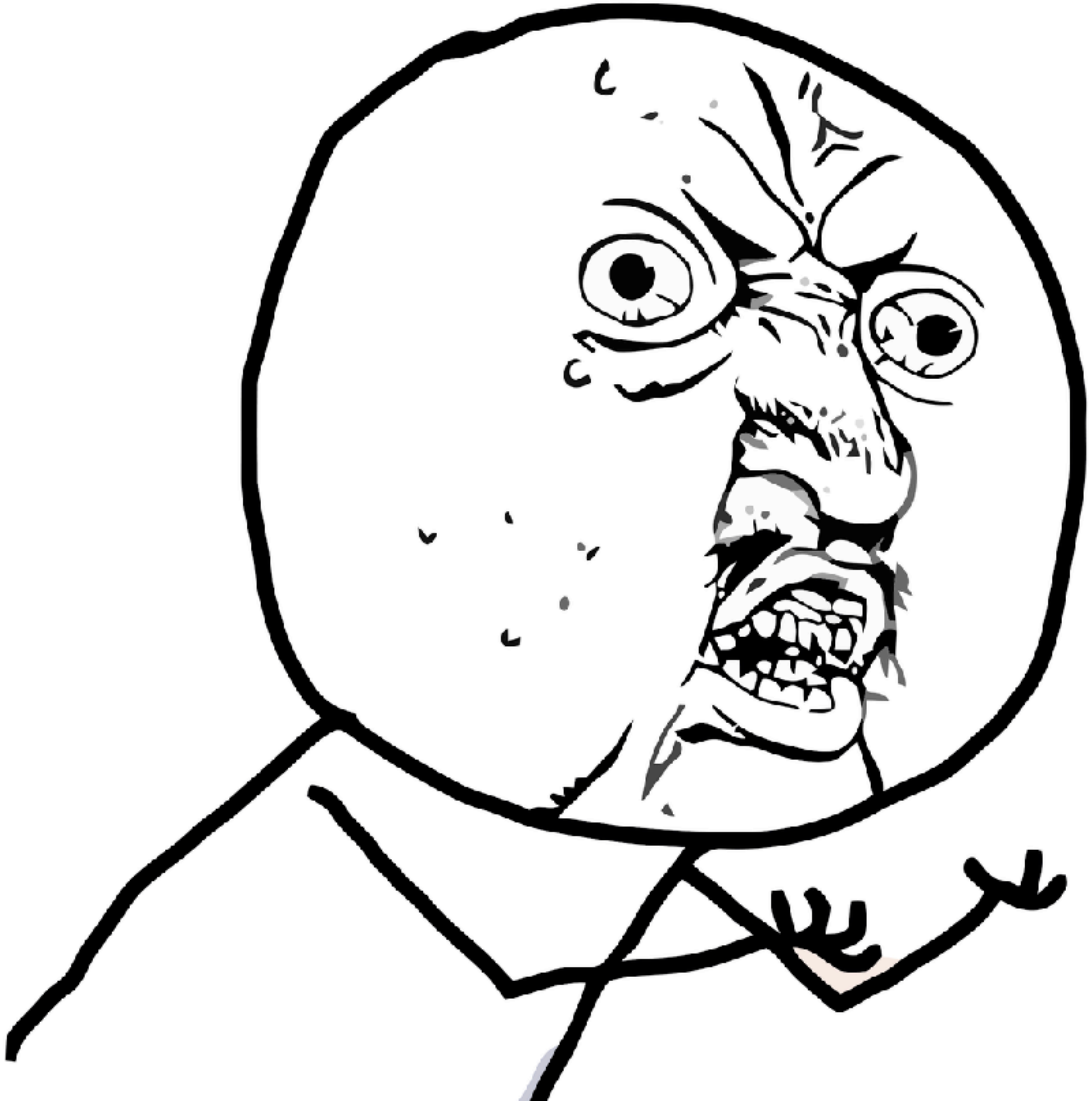


Stopping here on linting for now, but if interested, ask me about linters for code style later ;)

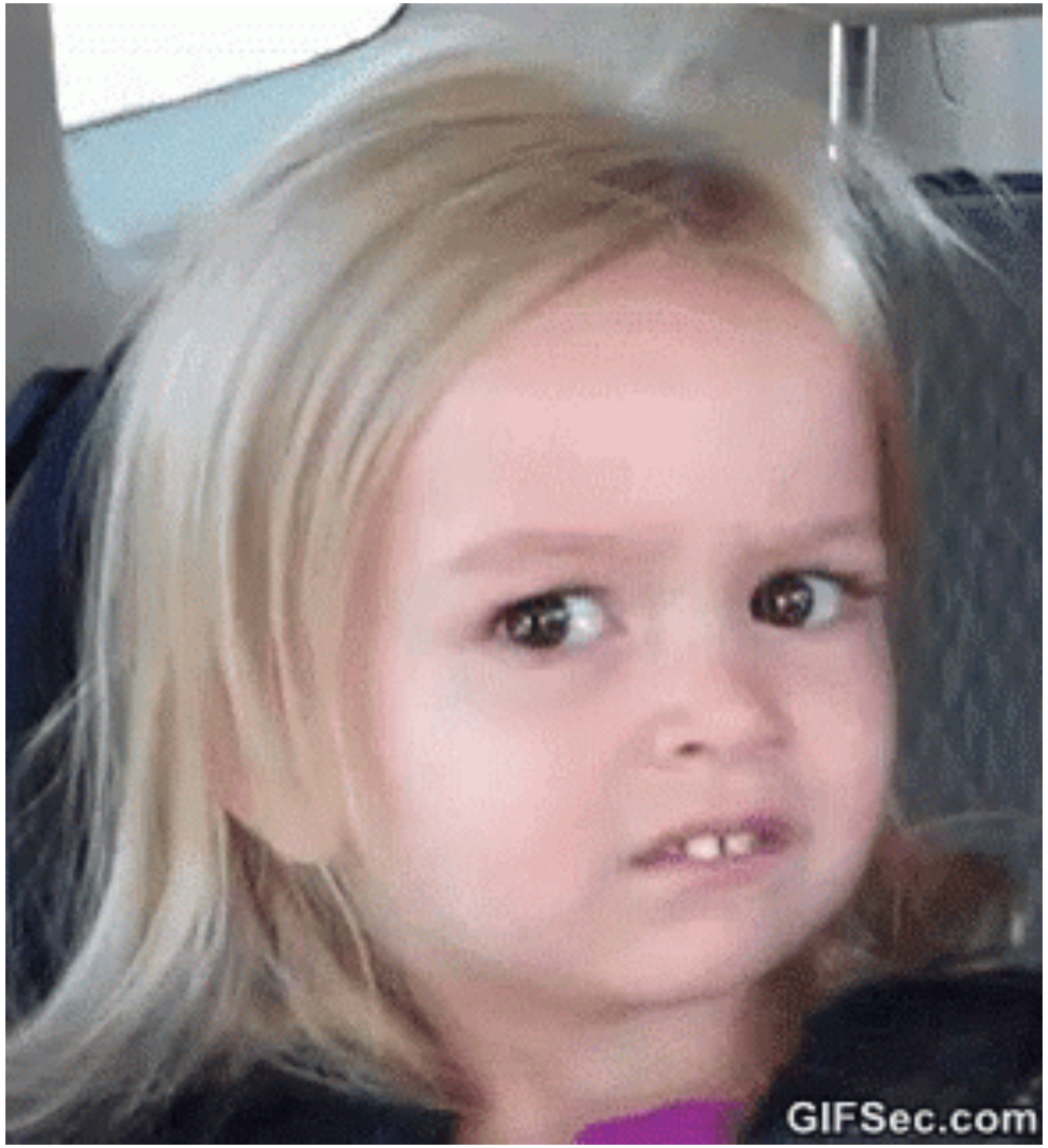
Poll time!



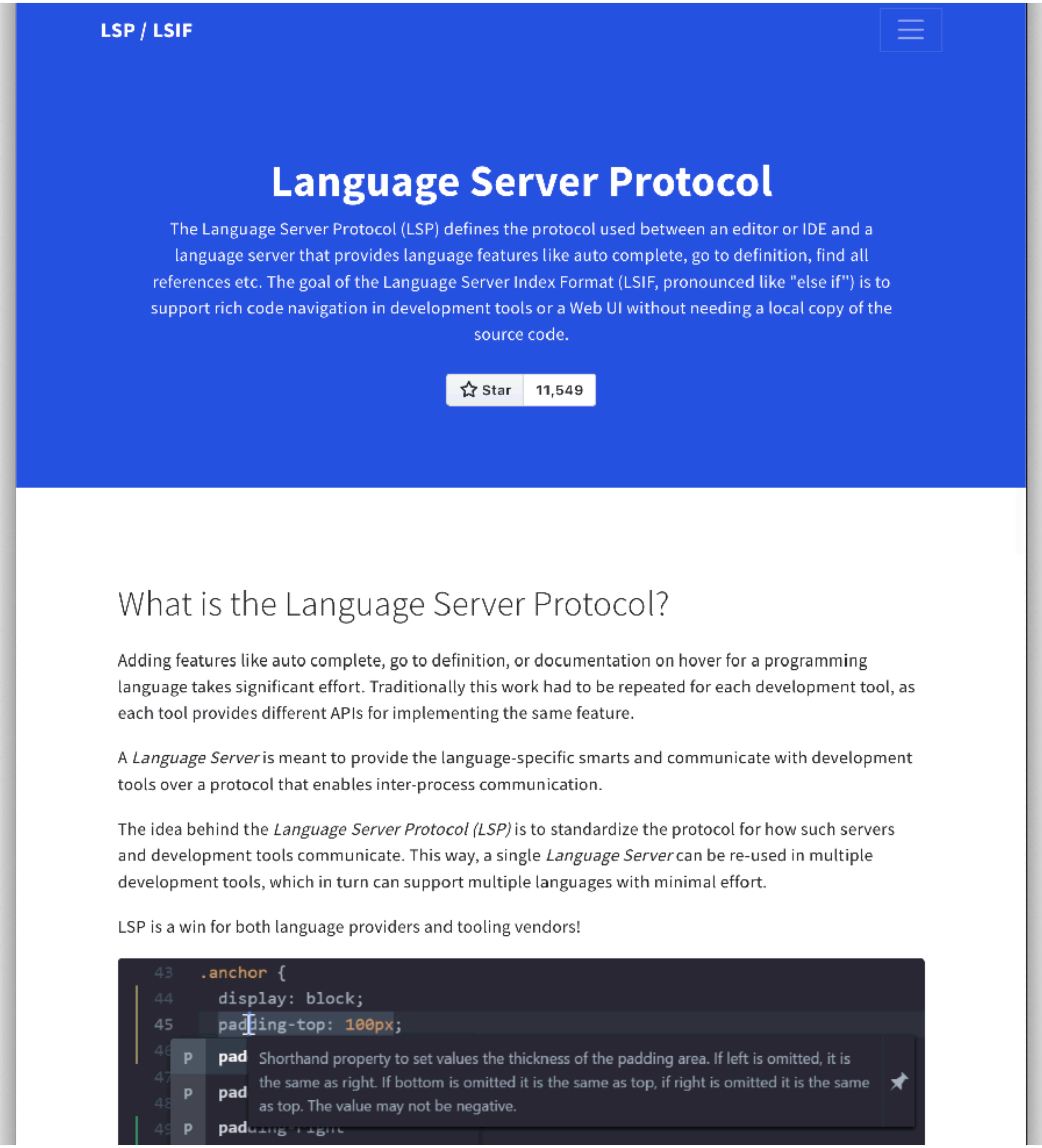
Users



Users



Maintainers



The problem: "The Matrix"

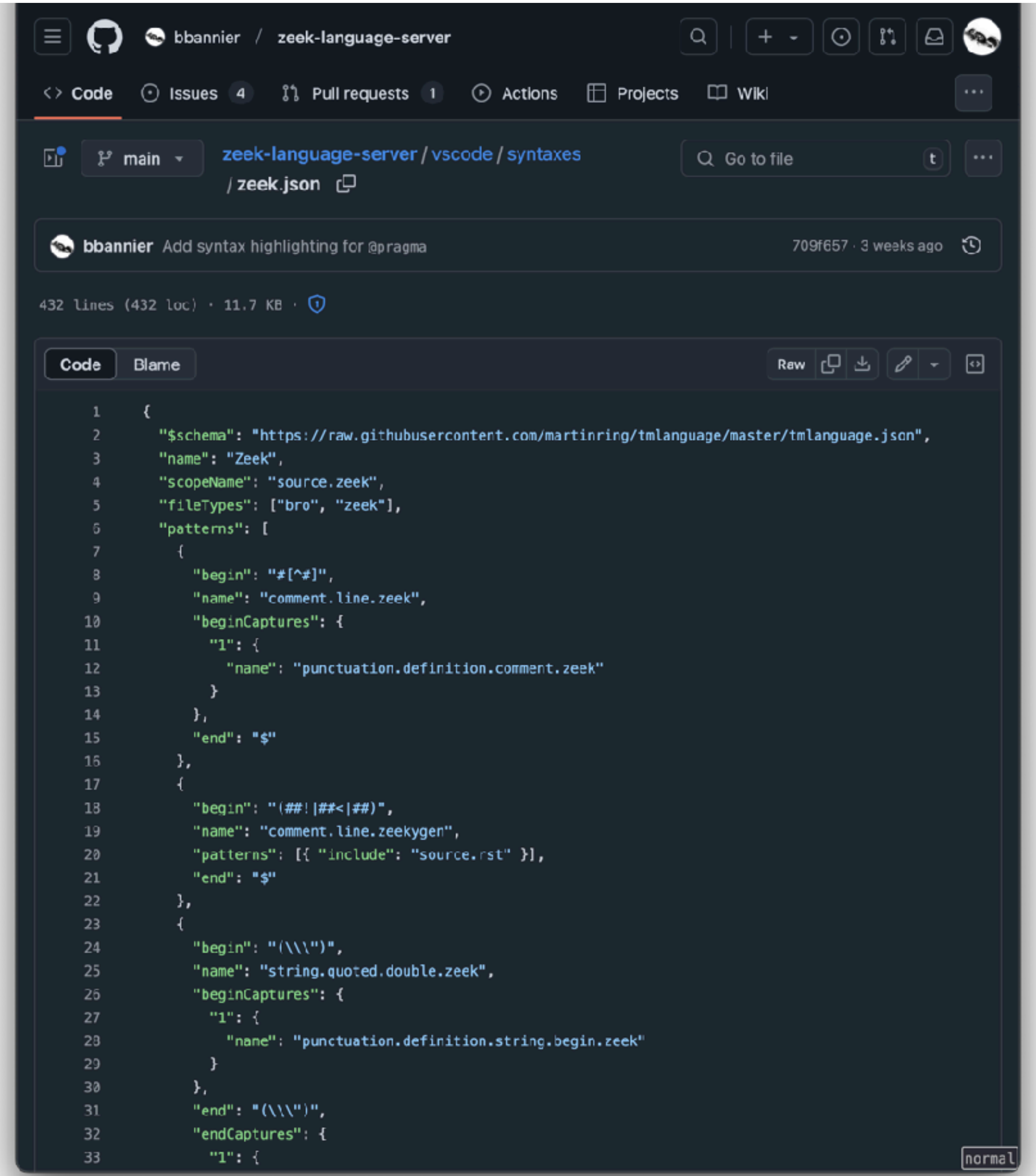
	Go	Java	TypeScript	...
Emacs				
Vim				
VSCode				
...				

The solution: lang servers and clients

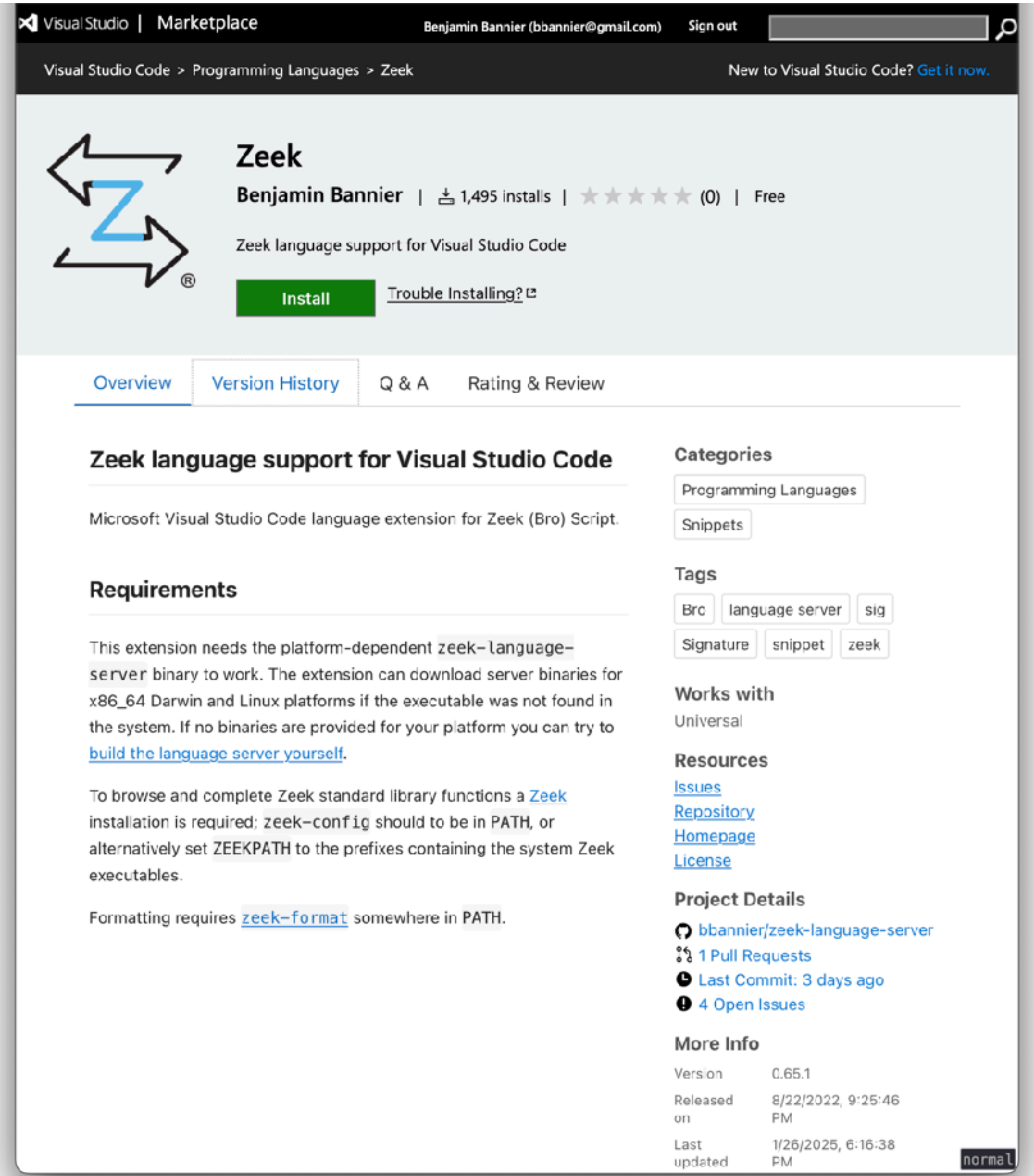
Go	✔
Java	✔
TypeScript	✔
...	

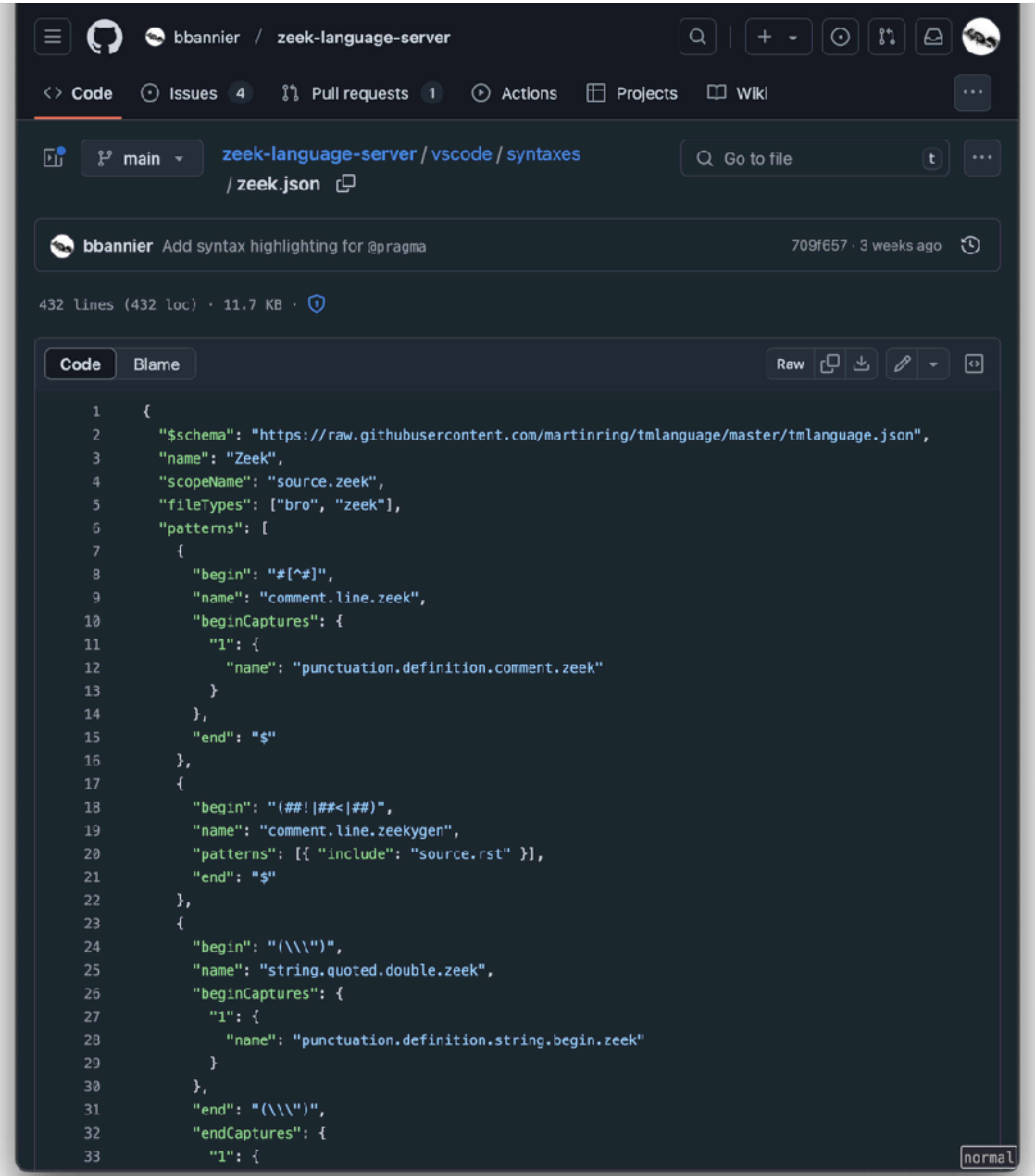
Emacs	✔
Vim	✔
VSCode	✔
...	

http://github.com/bbannier/zeek-language-server

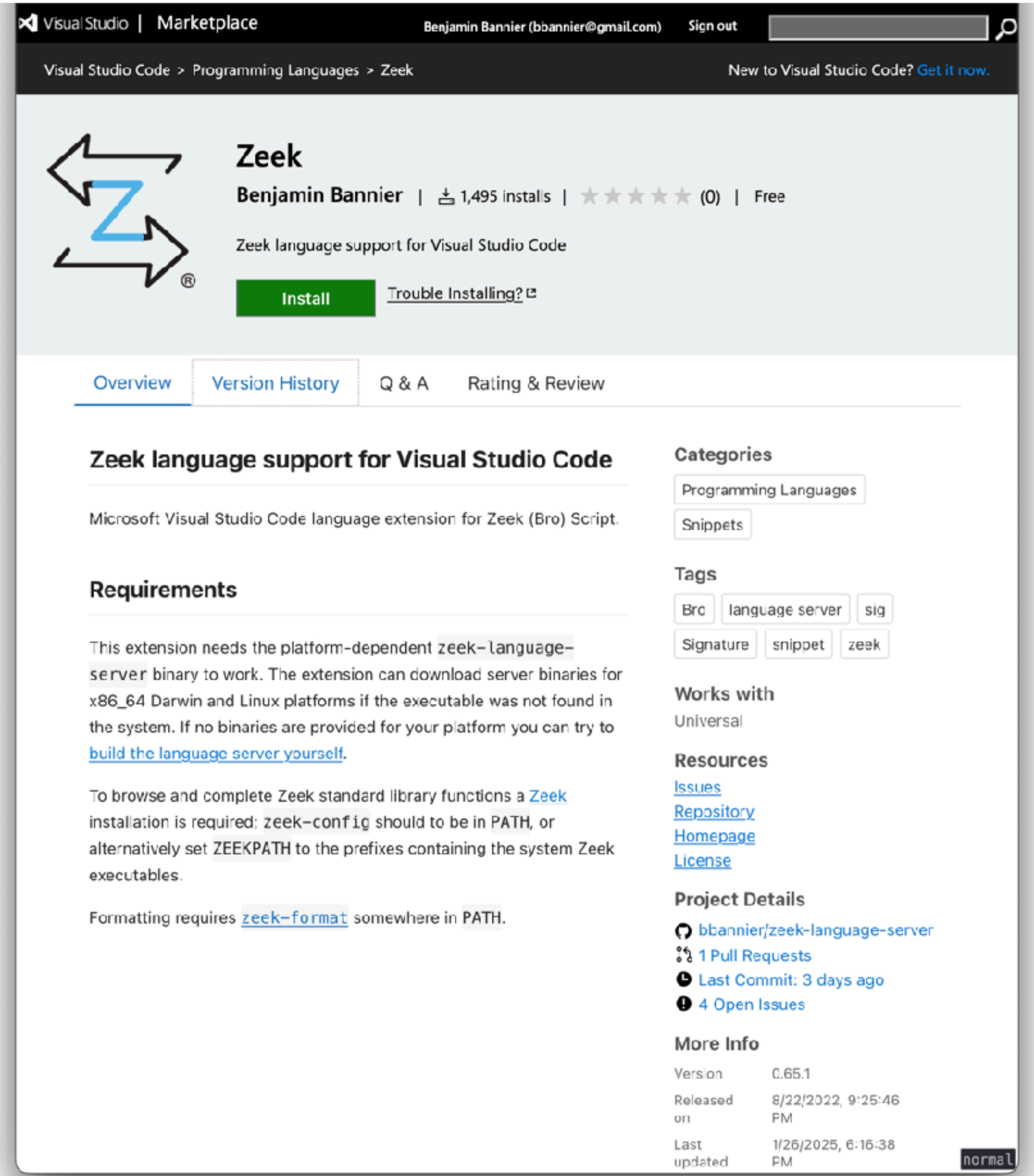


bbannier.zeek-language-server http://github.com/bbannier/zeek-language-server





bbannier.zeek-language-server



Visual Studio | Marketplace   Benjamin Bannier (bbannier@gmail.com)   Sign out

Visual Studio Code > Programming Languages > Zeek   New to Visual Studio Code? Get it now.

Zeek

Benjamin Bannier | 1,495 installs | (0) | Free

Zeek language support for Visual Studio Code

Install   Trouble Installing?

Overview   Version History   Q & A   Rating & Review

Zeek language support for Visual Studio Code

Microsoft Visual Studio Code language extension for Zeek (Bro) Script.

Requirements

This extension needs the platform-dependent `zeek-language-server` binary to work. The extension can download server binaries for x86_64 Darwin and Linux platforms if the executable was not found in the system. If no binaries are provided for your platform you can try to build the language server yourself.

To browse and complete Zeek standard library functions a Zeek installation is required; `zeek-config` should to be in `PATH`, or alternatively set `ZEEKPATH` to the prefixes containing the system Zeek executables.

Formatting requires `zeek-format` somewhere in `PATH`.

Categories

Programming Languages

Snippets

Tags

Bro   language server   sig

Signature   snippet   zeek

Works with

Universal

Resources

Issues

Repository

Homepage

License

Project Details

bbannier/zeek-language-server

1 Pull Requests

Last Commit: 3 days ago

4 Open Issues

More Info

Version	0.65.1
Released on	8/22/2022, 9:25:46 PM
Last updated	1/26/2025, 6:16:38 PM

normal

Demo

http://github.com/bbannier/zeek-playground