Future-Proof Your Vim Plugins:
Strategies for Robust Testing

VimConf2024
2024-11-23

:help Kazuma Inagaki

👾 a.k.a IK

💻 :DeNA

SWET Group2 /Quality Assurance Dept. / IT Unit

vim-airline organization member

VimDevIcons Adds Icons to Your Plugins Collaborators

@get-me-power

@get_me_power

Do you write tests?

What happens if the test isn't written?

- Failing to notice regressions
  - New changes unexpectedly affecting existing features

- Difficulty in reviewing pull requests
  - Without tests, it's hard to verify the behavior during code review

What happens if the test isn't written?

- Failing to notice regressions
    - New changes unexpectedly affecting existing features

The same tasks apply even when making a Vim plugin

    - Without tests, it's hard to verify the behavior during code review

Outline

Outline

1. Introduction to Simple Testing
2. Selection/Us

Explain Using the vim-devicons API as an Example

3. Points to Consider When Writing Tests
4. Efficient Flow for Easing Maintenance Starting from Tests
5. Conclusion

```vim
" A function that returns a specific icon based on the arguments
" a:1 (bufferName), a:2 (isDirectory)
function! WebDevIconsGetFileTypeSymbol(...) abort
endfunction


" Return the buffer icon or a default one.
call WebDevIconsGetFileTypeSymbol()


" Return the Vim icon.
call WebDevIconsGetFileTypeSymbol("hoge.vim")


" Return the folder icon.
call WebDevIconsGetFileTypeSymbol("hoge.vim", 1)
```

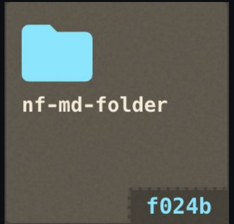
```vim
" A function that returns a specific icon based on the arguments
" a:1 (bufferName), a:2 (isDirectory)
function! WebDevIconsGetFileTypeSymbol(...) abort
endfunction


" Return the buffer icon or a default one.
call WebDevIconsGetFileTypeSymbol()


" Return the Vim icon.
call WebDevIconsGetFileTypeSymbol("hoge.vim")


" Return the folder icon.
call WebDevIconsGetFileTypeSymbol("hoge.vim", 1)
```

nf-custom-default
e612

```vim
" A function that returns a specific icon based on the arguments
" a:1 (bufferName), a:2 (isDirectory)
function! WebDevIconsGetFileTypeSymbol(...) abort
endfunction


" Return the buffer icon or a default one.
call WebDevIconsGetFileTypeSymbol()


" Return the Vim icon.
call WebDevIconsGetFileTypeSymbol("hoge.vim")


" Return the folder icon.
call WebDevIconsGetFileTypeSymbol("hoge.vim", 1)
```

nf-custom-vim
e62b

```vim
" A function that returns a specific icon based on the arguments
" a:1 (bufferName), a:2 (isDirectory)
function! WebDevIconsGetFileTypeSymbol(...) abort
endfunction


" Return the buffer icon or a default one.
call WebDevIconsGetFileTypeSymbol()


" Return the Vim icon.
call WebDevIconsGetFileTypeSymbol("hoge.vim")


" Return the folder icon.
call WebDevIconsGetFileTypeSymbol("hoge.vim", 1)
```



```vim
" A function that returns a specific icon based on the arguments
" a:1 (bufferName), a:2 (isDirectory)
function! WebDevIconsGetFileTypeSymbol(...) abort
endfunction


" Return the buffer icon or a default one.
ca


" Return the Vim icon.
call WebDevIconsGetFileTypeSymbol("hoge.vim")


" Return the folder icon.
call WebDevIconsGetFileTypeSymbol("hoge.vim", 1)
```

Next: Let’s write easy Test

nf-md-folder

f024b

```vim
function! TestWebDevIconsGetFileTypeSymbol()
  let v:errors = []

  call assert_equal('\ue612', WebDevIconsGetFileTypeSymbol())


  call assert_equal('\ue62b', WebDevIconsGetFileTypeSymbol("hoge.vim"))


  call assert_equal('\uf07b', WebDevIconsGetFileTypeSymbol("hoge.vim", 1))


  if len(v:errors) >= 1
    echo v:errors
    return
  endif

  echo 'test success'
endfunction
```

```
function! TestWebDevIconsGetFileTypeSymbol()
  let v:errors = []

  call assert_equal('\ue612', WebDevIconsGetFileTypeSymbol())


  call assert_equal('\ue62b', WebDevIconsGetFileTypeSymbol("hoge.vim"))


  call assert_equal('\uf07b', WebDevIconsGetFileTypeSymbol("hoge.vim", 1))


  if le
    echo
    return
  endif

  echo 'test success'
endfunction
```

What’s asset_equal ?

assert_equal({expected}, {actual} [, {msg}])

When {expected} and {actual} are not equal an error message is added to v:errors and 1 is returned. Otherwise zero is returned. assert-return

The error is in the form "Expected {expected} but got {actual}". When {msg} is present it is prefixed to that, along with the location of the assert when run from a script.

There is no automatic conversion, the String "4" is different from the Number 4. And the number 4 is different from the Float 4.0. The value of 'ignorecase' is not used here, case always matters.

Example:

```
call assert_equal('foo', 'bar', 'baz')
```

Will add the following to v:errors:

```
test.vim line 12: baz: Expected 'foo' but got 'bar'
```

Can also be used as a method, the base is passed as the second argument:

```
mylist->assert_equal([1, 2, 3])
```

Return type: Number

assert_equal({expected}, {actual} [, {msg}])

When {expected} and {actual} are not equal an error message is added to v:errors and 1 is returned. Otherwise zero is returned. assert-return

The error is in the form "Expected {expected} but got {actual}". When {msg} is present it is prefixed to that, along with the location of the assert when run from a script.

There is no automatic conversion, the String "4" is different from the Number 4. And the number 4 is different from the

Next: Explaining the test process

Will add the following to v:errors:

```
test.vim line 12: baz: Expected 'foo' but got 'bar'
```

Can also be used as a method, the base is passed as the second argument:

```
mylist->assert_equal([1, 2, 3])
```

Return type: Number

```
function! TestWebDevIconsGetFileTypeSymbol()
  let v:errors = []

  call assert_equal('\ue612', WebDevIconsGetFileTypeSymbol())


  call assert_equal('\ue62b', WebDevIconsGetFileTypeSymbol("hoge.vim"))


  call assert_equal('\uf07b', WebDevIconsGetFileTypeSymbol("hoge.vim", 1))


  if len(v:errors) >= 1
    echo v:errors
    return
  endif

  echo 'test success'
endfunction
```

Accumulate errors found by the test function.

```vim
function! TestWebDevIconsGetFileTypeSymbol()
  let v:errors = []

  call assert_equal('\ue612', WebDevIconsGetFileTypeSymbol())


  call assert_equal('\ue62b', WebDevIconsGetFileTypeSymbol("hoge.vim"))


  call assert_equal('\uf07b', WebDevIconsGetFileTypeSymbol("hoge.vim", 1))


  if len(v:errors) >= 1
    echo v:errors
    return
  endif

  echo 'test success'
endfunction
```

Run the test.

```
function! TestWebDevIconsGetFileTypeSymbol()
  let v:errors = []

  call assert_equal('\ue612', WebDevIconsGetFileTypeSymbol())


  call assert_equal('\ue62b', WebDevIconsGetFileTypeSymbol("hoge.vim"))


  call assert_equal('\uf07b', WebDevIconsGetFileTypeSymbol("hoge.vim", 1))


  if len(v:errors) >= 1
    echo v:errors
    return
  endif

  echo 'test success'
endfunction
```

Display the test results.

```
function! TestWebDevIconsGetFileTypeSymbol()
  let v:errors = []

  call assert_equal('\ue612', WebDevIconsGetFileTypeSymbol())


  call assert_equal('\ue62b', WebDevIconsGetFileTypeSymbol("hoge.vim"))
```

:call TestWebDevIconsGetFileTypeSymbol()

```
                                                                   ))

  if len(v:errors) >= 1
    echo v:errors
    return
  endif

  echo 'test success'
endfunction
```

Successful test case.

test success

Failed test case.

```
['function TestWebDevIconsGetFileTypeSymbol line 3: Expected "" but got "<98><ab>"',
'function TestWebDevIconsGetFileTypeSymbol line 4: Expected "" but got "<98><ab>"',
'function TestWebDevIconsGetFileTypeSymbol line 5: Expected "" but got "<81><bb>"']
```

テストが成功した場合

test succ

テストが失敗し

- Failure results are unclear.
- Rules or guidelines for writing tests are needed.
- Management of multiple test cases is required.
- Easy separation of test environments is desired.

```
['function TestWebDevIconsGetFileTypeSymbol line 3: Expected "" but got "<98><ab>"',
'function TestWebDevIconsGetFileTypeSymbol line 4: Expected "" but got "<98><ab>"',
'function TestWebDevIconsGetFileTypeSymbol line 5: Expected "" but got "<81><bb>"']
```

テストが成功した場合

test success

- Failure results are unclear.

Let's use a testing framework.

テ

```
['function TestWebDevIconsGetFileTypeSymbol line 3: Expected "" but got "<98><ab>"',
'function TestWebDevIconsGetFileTypeSymbol line 4: Expected "" but got "<98><ab>"',
'function TestWebDevIconsGetFileTypeSymbol line 5: Expected "" but got "<81><bb>"']
```

Outline

Benefits of Introducing a Testing Framework

- Test reports are easy to understand.
- Test case writing can be standardized, enhancing maintainability.
- Multiple test cases can be centrally managed by the testing framework.
- Test environments in Vim can be easily separated.

What testing frameworks are available for Vim script?

- thinca/vim-themis
- junegoon/vader.vim
- kana/vim-vspec
- google/vroom

...

What testing frameworks are available for Vim script?

- thinca/vim-themis
- junegoon/vader.vim
- kana/vim-vspec
- google/vroom

…

It's so confusing to figure out which one is better!

Points to Consider When Making a Selection

- Can it be executed from the shell?
  - Want to execute tests in one line for potential CI integration.
- Does it require environments other than Vim?
  - If other environments are needed, it increases the setup effort both locally and in CI.
- Can it manage plugins that the test plugin depends on?
  - Being able to manage them lowers the difficulty of setting up the environment.
  - Test execution be standardized in the local environment

Selection of a Testing Framework

	Can be executed from the shell	「Works with only Vim	Can incorporate dependent plugins during testing
thinca/ vim-themis	◯	◯	◯
junegoon/ vader.vim	✗	◯	✗
kana/ vim-vspec	△	✗	✗
google/ vroom	✗	◯	✗

Selection of a Testing Framework

	Can be executed from the shell	「Works with only Vim	Can incorporate dependent plugins during testing
junegoon/ vader.vim	✗	◯	✗
kana/ vim-vspec	△	✗	✗
google/ vroom	✗	◯	✗

Let's use vim-themis

Outline

Points to consider when writing tests

- Ensure that the results do not change whether executed locally or on CI.

- Prevent assertion roulette.

Points to consider when writing tests

- Ensure that the results do not change whether executed locally or on CI.

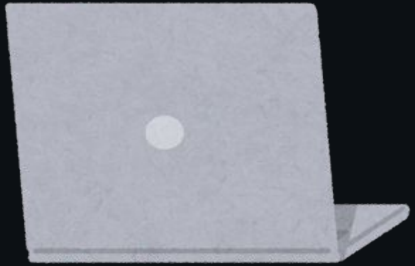
An explanation based on vim-airline's test cases.

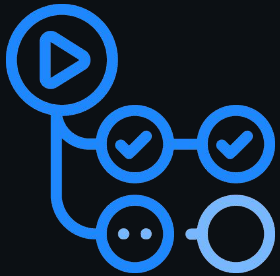
- Prevent assertion roulette.

```
It should extract correct colors
  call airline#highlighter#reset_hlcache()
  highlight Foo ctermfg=1 ctermbg=2
  let colors = airline#themes#get_highlight('Foo')
  Assert Equals(colors[0], 'NONE')
  Assert Equals(colors[1], 'NONE')
  Assert Equals(colors[2], '1')
  Assert Equals(colors[3], '2')
End
```

▼ Execute locally

▼ Execute on CI

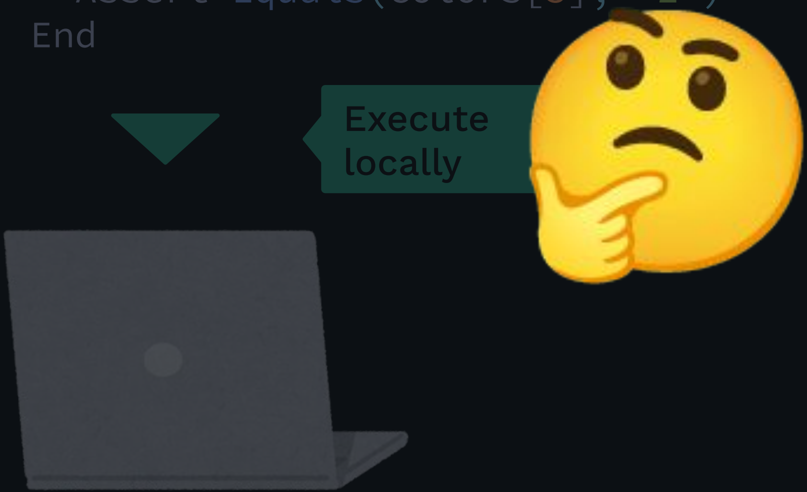


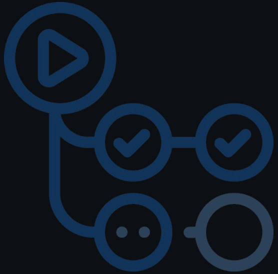


❌ Failed

✅ Success

```
It should extract correct colors
  call airline#highlighter#reset_hlcache()
  highlight Foo ctermfg=1 ctermbg=2
  let colors = airline#themes#get_highlight('Foo')
  Assert Equals(colors[0], 'NONE')
  Assert Equals(colors[1], 'NONE')
  Assert Equals(colors[2], '1')
  Assert Equals(colors[3], '2')
End
```

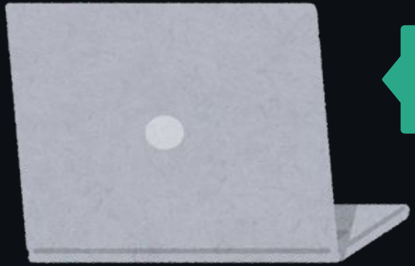




❌ Failed

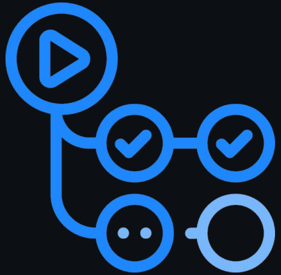
✅ Success

```
It should extract correct colors
  call airline#highlighter#reset_hlcache()
  highlight Foo ctermfg=1 ctermbg=2
  let colors = airline#themes#get_highlight('Foo')
  Assert Equals(colors[0], 'NONE')
  Assert Equals(colors[1], 'NONE')
  Assert Equals(colors[2], '1')
  Assert Equals(colors[3], '2')
End
```



set termguicolor

❌ Failed



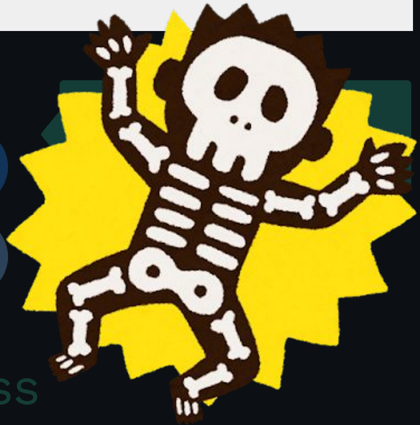
set notermguicolor

✅ Success

```
It should extract correct colors
  call airline#highlighter#reset_hlcache()
  highlight Foo ctermfg=1 ctermbg=2
  let colors = airline#themes#get_highlight('Foo')
  Assert Equals(colors[0], 'NONE')
  Assert Equals(colors[1], 'NONE')
  Assert Equals(colors[2], '1')
  Assert Equals(colors[3], '2')
End
```

The test results change depending on Vim options!

set termguicolor



❌ Failed

✅ Success

```
It should extract correct colors with notermguicolors
  set notermguicolors
  call airline#highlighter#reset_hlcache()
  highlight Foo ctermfg=1 ctermbg=2
  let colors = airline#themes#get_highlight('Foo')
  Assert Equals(...)
End

It should extract correct colors with termguicolors
  if !exists("+termguicolors")
    Assert Skip("termguicolors is disable build. Skip this test.")
  endif
  set termguicolors
  call airline#highlighter#reset_hlcache()
  highlight Foo ctermfg=1 ctermbg=2
  let colors = airline#themes#get_highlight('Foo')
  Assert Equals(...)
End
```

```
It should extract correct colors with notermguicolors
  set notermguicolors
  call airline#highlighter#reset_hlcache()
  highlight Foo ctermfg=1 ctermbg=2
  let colors = airline#themes#get_highlight('Foo')
  Assert Equals(...)
End

It should extract correct colors with termguicolors
  if !exists("+termguicolors")
    Assert Skip("termguicolors is disable build. Skip this test.")
  endif
  set termguicolors
  call airline#highlighter#reset_hlcache()
  highlight Foo ctermfg=1 ctermbg=2
  let colors = airline#themes#get_highlight('Foo')
  Assert Equals(...)
End
```

```
It should extract correct colors with notermguicolors
  set notermguicolors
  call airline#highlighter#reset_hlcache()
  highlight Foo ctermfg=1 ctermbg=2
  let colors = airline#themes#get_highlight('Foo')
  Assert Equals(...)
End
```

for notermguicolor method

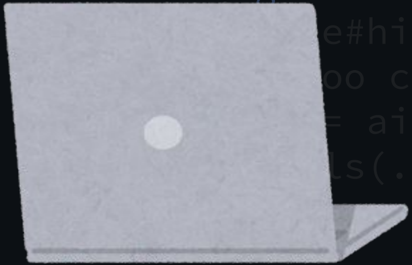
```
                          colors with termguicolors
  if !exists("+termguicolors")
    Assert Skip("termguicolors is disable build. Skip this test.")
  endif
  set termguicolors
  call airline#highlighter#reset_hlcache()
  highlight Foo ctermfg=1 ctermbg=2
  let colors = airline#themes#get_highlight('Foo')
  Assert Equals(...)
End
```

```
It should extract correct colors with notermguicolors
  set notermguicolors
  call airline#highlighter#reset_hlcache()
  highlight Foo ctermfg=1 ctermbg=2
  let colors = airline#themes#get_highlight('Foo')
```

for termguicolors method

```
It should extract correct colors with termguicolors
  if !exists("+termguicolors")
    Assert Skip("termguicolors is disable build. Skip this test.")
  endif
  set termguicolors
  call airline#highlighter#reset_hlcache()
  highlight Foo ctermfg=1 ctermbg=2
  let colors = airline#themes#get_highlight('Foo')
  Assert Equals(...)
End
```

```
It should extract correct colors with notermguicolors
  set notermguicolors
  call airline#highlighter#reset_hlcache()
  highlight Foo ctermfg=1 ctermbg=2
  let colors = airline#themes#get_highlight('Foo')
  Assert Equals(...)
End

It should extract correct colors with termguicolors
  if !exists("+termguicolors")
    Assert Skip("termguicolors is disable build. Skip this test.")
  endif
  set termguicolors
             e#highlighter#reset_hlcache()
             oo ctermfg=1 ctermbg=2
             = airline#themes#get_highlight('Foo')
             s(...)
```



✅ Success

✅ Success

Points to consider when writing tests

- Ensure that the results do not change whether executed locally or on CI.

- Prevent assertion roulette.

An explanation based on vim-devicons' test cases.

```vim
let s:suite = themis#suite('WebDevIconsGetFileTypeSymbol')
let s:assert = themis#helper('assert')

function! s:suite.WebDevIconsGetFileTypeSymbol_testdotvim_returnVimIcon()
  call s:assert.equals(WebDevIconsGetFileTypeSymbol('test.vim'), "\ue62b")
endfunction

function! s:suite.WebDevIconsGetFileTypeSymbol_vimrc_returnVimIcon()
  call s:assert.equals(WebDevIconsGetFileTypeSymbol('vimrc'), "\ue62b")
endfunction

function! s:suite.WebDevIconsGetFileTypeSymbol_gvimrc_returnVimIcon()
  call s:assert.equals(WebDevIconsGetFileTypeSymbol('gvimrc'), "\ue62b")
endfunction
```

```
let s:suite = themis#suite('WebDevIconsGetFileTypeSymbol')
let s:assert = themis#helper('assert')

function! s:suite.WebDevIconsGetFileTypeSymbol_testdotvim_returnVimIcon()
  call s:assert.equals(WebDevIconsGetFileTypeSymbol('test.vim'), "\ue62b")
endfunction

function! s:suite.WebDevIconsGetFileTypeSymbol_vimrc_returnVimIcon()
  call s:assert.equals(WebDevIconsGetFileTypeSymbol('vimrc'), "\ue62b")
endfunction

fur
  c
enc
```

$themis test –reporter spec

```vim
let s:suite = themis#suite('WebDevIconsGetFileTypeSymbol')
let s:assert = themis#helper('assert')

function! s:suite.WebDevIconsGetFileTypeSymbol_testdotvim_returnVimIcon()
  call s:assert.equals(WebDevIconsGetFileTypeSymbol('test.vim'), "\ue62b")
endfunction

function! s:suite.WebDevIconsGetFileTypeSymbol_vimrc_returnVimIcon()
  call s:assert.equals(WebDevIconsGetFileTypeSymbol('vimrc'), "\ue62b")
endfunction

function! s:suite.WebDevIconsGetFileTypeSymbol_gvimrc_returnVimIcon()
  call s:assert.equals(WebDevIconsGetFileTypeSymbol('gvimrc'), "\ue62b")
endfunction
```

```
WebDevIconsGetFileTypeSymbol
  [✓] WebDevIconsGetFileTypeSymbol_testdotvim_returnVimIcon
  [✓] WebDevIconsGetFileTypeSymbol_vimrc_returnVimIcon
  [✓] WebDevIconsGetFileTypeSymbol_gvimrc_returnVimIcon

tests 3
passes 3
```

```
let s:suite = themis#suite('WebDevIconsGetFileTypeSymbol')
let s:assert = themis#helper('assert')

function! s:suite.WebDevIconsGetFileTypeSymbol_testdotvim_returnVimIcon()
  call s:assert.equals(WebDevIconsGetFileTypeSymbol('test.vim'), "\ue62b")
endfunction

function! s:suite.WebDevIconsGetFileTypeSymbol_vimrc_returnVimIcon()
  call s:assert.equals(WebDevIconsGetFileTypeSymbol('vimrc'), "\ue62b")
endfunction

function! s:suite.WebDevIconsGetFileTypeSymbol_gvimrc_returnVimIcon()
  call s:assert.equals(WebDevIconsGetFileTypeSymbol('gvimrc'), "\ue62b")
endfunction
```

Combine test methods into one

```
let s:suite = themis#suite('WebDevIconsGetFileTypeSymbol')
let s:assert = themis#helper('assert')

function! s:suite.OneArgumet_GetVimIcon()
  call s:assert.equals(WebDevIconsGetFileTypeSymbol('test.vim'), "\ue62b")
  call s:assert.equals(WebDevIconsGetFileTypeSymbol('vimrc'), "\ue62b")
  call s:assert.equals(WebDevIconsGetFileTypeSymbol('gvimrc'), "\ue62b")
endfunction
```

```
let s:suite = themis#suite('WebDevIconsGetFileTypeSymbol')
let s:assert = themis#helper('assert')

function! s:suite.WebDevIconsGetFileTypeSymbol_testdotvim_returnVimIcon()
  call s:assert.equals(WebDevIconsGetFileTypeSymbol('test.vim'), "\ue62b")
endfunction

function! s:suite.WebDevIconsGetFileTypeSymbol_vimrc_returnVimIcon()
  call s:assert.equals(WebDevIconsGetFileTypeSymbol('vimrc'), "\ue62b")
endfunction

function! s:suite.WebDevIconsGetFileTypeSymbol_gvimrc_returnVimIcon()
  call s:assert.equals(WebDevIconsGetFileTypeSymbol('gvimrc'), "\ue62b")
endfunction
```

Simplify using a for loop

```
let s:suite = themis#suite('WebDevIconsGetFileTypeSymbol')
let s:assert = themis#helper('assert')

function! s:suite.OneArgumet_GetVimIcon()
  let targetfilenames = ['test.vim', '.vimrc', 'gvimrc']

  for targetfilename in targetfilenames
    call s:assert.equals(WebDevIconsGetFileTypeSymbol(targetfilename), "\ue62b")
  endfor
endfunction
```

```
let s:suite = themis#suite('WebDevIconsGetFileTypeSymbol')
let s:assert = themis#helper('assert')

function! s:suite.WebDevIconsGetFileTypeSymbol_testdotvim_returnVimIcon()
  call s:assert.equals(WebDevIconsGetFileTyp       'test.vim'), "\ue62b")
endfunction

function! s:suite.WebDevIconsGetFileType                   nVimIcon()
  call s:assert.equals(WebDevIconsGetFil                  '), "\ue62b")
endfunction

function! s:suite.WebDevIconsGetFileTy                      VimIcon()
  call s:assert.equals(WebDevIconsGetFi                    '), "\ue62b")
endfunction
```

ing a for loop

```
let s:suite = themis#suite('WebDevIcons
let s:assert = themis#helper('assert')

function! s:suite.OneArgumet_GetVimIcon(
  let targetfilenames = ['test.vim', '.vim

  for targetfilename in targetfilenames
    call s:assert.equals(WebDevIconsGetFileTypeSymbol(targetfilename), "\ue62b")
  endfor
endfunction
```

What is the issue?

Combine test methods into one

```
let s:suite = themis#suite('WebDevIconsGetFileTypeSymbol')
let s:assert = themis#helper('assert')

function! s:suite.OneArgumet_GetVimIcon()
  call s:assert.equals(WebDevIconsGetFileTypeSymbol('test.vim'), "\ue62b")
  call s:assert.equals(WebDevIconsGetFileTypeSymbol('vimrc'), "\ue62b")
  call s:assert.equals(WebDevIconsGetFileTypeSymbol('gvimrc'), "\ue62b")
endfunction
```

Simplify using a for loop

```
let s:suite = themis#suite('WebDevIconsGetFileTypeSymbol')
let s:assert = themis#helper('assert')

function! s:suite.OneArgumet_GetVimIcon()
  call s:assert.equals(WebDevIconsGetFileTypeSymbol('test.vim'), "\ue62b")
  call s:assert.equals(WebDevIconsGetFileTypeSymbol('vimrc'), "\ue62b")
  call s:assert.equals(WebDevIconsGetFileTypeSymbol('gvimrc'), "\ue62b")
endfunction
```

Combine test methods into one

```
let s:suite = themis#suite('WebDevIconsGetFileTypeSymbol')
let s:assert = themis#helper('assert')

function! s:suite.OneArgumet_GetVimIcon()
  call s:assert.equals(WebDevIconsGetFileTypeSymbol('test.vim'), "\ue62b")
  call s:assert.equals(WebDevIconsGetFileTypeSymbol('vimrc'), "\ue62b")
  call s:assert.equals(WebDevIconsGetFileTypeSymbol('gvimrc'), "\ue62b")
endfunction
```

Si

Intentionally cause it to fail

```
let s:suite = themis#suite('WebDevIconsGetFileTypeSymbol')
let s:assert = themis#helper('assert')

function! s:suite.OneArgumet_GetVimIcon()
  call s:assert.equals(WebDevIconsGetFileTypeSymbol('test.vim'), "\ue62b")
  call s:assert.equals(WebDevIconsGetFileTypeSymbol('vimrc'), "\ue62b")
  call s:assert.equals(WebDevIconsGetFileTypeSymbol('gvimrc'), "\ue62b")
endfunction
```

Combine test methods into one

```
let s:suite = themis#suite('WebDevIconsGetFileTypeSymbol')
let s:assert = themis#helper('assert')

function! s:suite.OneArgumet_GetVimIcon()
  call s:assert.equals(WebDevIconsGetFileTypeSymbol('test.vim'), "")
  call s:assert.equals(WebDevIconsGetFileTypeSymbol('vimrc'), "")
  call s:assert.equals(WebDevIconsGetFileTypeSymbol('gvimrc'), "")
endfunction
```

Simplify using a for loop

```
let s:suite = themis#suite('WebDevIconsGetFileTypeSymbol')
let s:assert = themis#helper('assert')

function! s:suite.OneArgumet_GetVimIcon()
  call s:assert.equals(WebDevIconsGetFileTypeSymbol('test.vim'), "")
  call s:assert.equals(WebDevIconsGetFileTypeSymbol('vimrc'), "")
  call s:assert.equals(WebDevIconsGetFileTypeSymbol('gvimrc'), "")
endfunction
```

Combine test methods into one

```
let s:suite = themis#suite('WebDevIconsGetFileTypeSymbol')
let s:assert = themis#helper('assert')

function! s:suite.OneArgumet_GetVimIcon()
  call s:assert.equals(WebDevIconsGetFileTypeSymbol('test.vim'), "")
  call s:assert.equals(WebDevIconsGetFileTypeSymbol('vimrc'), "")
  call s:assert.equals(WebDevIconsGetFileTypeSymbol('gvimrc'), "")
en
```

$themis test –reporter spec

Si

```
let s:suite = themis#suite('WebDevIconsGetFileTypeSymbol')
let s:assert = themis#helper('assert')

function! s:suite.OneArgumet_GetVimIcon()
  call s:assert.equals(WebDevIconsGetFileTypeSymbol('test.vim'), "")
  call s:assert.equals(WebDevIconsGetFileTypeSymbol('vimrc'), "")
  call s:assert.equals(WebDevIconsGetFileTypeSymbol('gvimrc'), "")
endfunction
```

Combine test methods into one

```
let s:suite = themis#suite('WebDevIconsGetFileTypeSymbol')
let s:assert = themis#helper('assert')

function! s:suite.OneArgumet_GetVimIcon()
  call s:assert.equals(WebDevIconsGetFileTypeSymbol('test.vim'), "")
  call s:assert.equals(WebDevIconsGetFileTypeSymbol('vimrc'), "")
  call s:assert.equals(WebDevIconsGetFileTypeSymbol('gvimrc'), "")
endfunction
```

```
WebDevIconsGetFileTypeSymbol
  [✖] OneArgumet_GetVimIcon
    The equivalent values were expected, but it was not the case.

      expected: ""
           got: "¥ue612"

tests 1
passes 0
fails 1
```

Simplify using a for loop

```
let s:suite = themis#suite('WebDevIconsGetFileTypeSymbol')
let s:assert = themis#helper('assert')

function! s:suite.OneArgumet_GetVimIcon()
  call s:assert.equals(WebDevIconsGetFileTypeSymbol('test.vim'), "")
  call s:assert.equals(WebDevIconsGetFileTypeSymbol('vimrc'), "")
  call s:assert.equals(WebDevIconsGetFileTypeSymbol('gvimrc'), "")
endfunction
```

Combine test methods into one

```vim
let s:suite = themis#suite('WebDevIconsGetFileTypeSymbol')
let s:assert = themis#helper('assert')

function! s:suite.OneArgumet_GetVimIcon()
  call s:assert.equals(WebDevIconsGetFileTypeSymbol('test.vim'), "")
  call s:assert.equals(WebDevIconsGetFileTypeSymbol('vimrc'), "")
  call s:assert.equals(WebDevIconsGetFileTypeSymbol('gvimrc'), "")
endfunction
```

```
WebDevIconsGetFileTypeSymbol
  [✖] OneArgumet_GetVimIcon
    The equivalent values were expected, but it was not the case.

        expected: ""
             got: "¥ue612"

tests 1
passes 0
fails 1
```

Simplify using a for loop

```vim
let s:suite = themis#suite('WebDevIconsGetFileTypeSymbol')
let s:assert = themis#helper('assert')

function! s:suite.OneArgumet_GetVimIcon()
  call s:assert.equals(WebDevIconsGetFileTypeSymbol('test.vim'), "")
  call s:assert.equals(WebDevIconsGetFileTypeSymbol('vimrc'), "")
  call s:assert.equals(WebDevIconsGetFileTypeSymbol('gvimrc'), "")
endfunction
```

!?

Combine test methods into one

```
let s:suite = themis#suite('WebDevIconsGetFileTypeSymbol')
let s:assert = themis#helper('assert')

function! s:suite.OneArgumet_GetVimIcon()
  call s:assert.equals(WebDevIconsGetFileTypeSymbol('test.vim'), "")
  call s:assert.equals(WebDevIconsGetFileTypeSymbol('vimrc'), "")
  call s:assert.equals(WebDevIconsGetFileTypeSymbol('gvimrc'), "")
endfunction
```

Don't know where it failed!

The equivalent values were expected, but it was not the case.

```
let s:suite = themis#suite('WebDevIconsGetFileTypeSymbol')
let s:assert = themis#helper('assert')

function! s:suite.OneArgumet_GetVimIcon()
  call s:assert.equals(WebDevIconsGetFileTypeSymbol('test.vim'), "")
  call s:assert.equals(WebDevIconsGetFileTypeSymbol('vimrc'), "")
  call s:assert.equals(WebDevIconsGetFileTypeSymbol('gvimrc'), "")
endfunction
```

expected: ""
got: "¥ue612"

tests 1
passes 0
fails 1

!?

```vim
let s:suite = themis#suite('WebDevIconsGetFileTypeSymbol')
let s:assert = themis#helper('assert')

function! s:suite.OneArgumet_GetVimIcon()
  let targetfilenames = ['test.vim', '.vimrc', 'gvimrc']

  for targetfilename in targetfilenames
    call s:assert.equals(WebDevIconsGetFileTypeSymbol(targetfilename), "\ue62b")
  endfor
endfunction
```

```
let s:suite = themis#suite('WebDevIconsGetFileTypeSymbol')
let s:assert = themis#helper('assert')

function! s:suite.OneArgumet_GetVimIcon()
  let targetfilenames = ['test.vim', '.vimrc', 'gvimrc']

  for targetfilename in targetfilenames
    call s:assert.equals(WebDevIconsGetFileTypeSymbol(targetfilename), "\ue62b")
  endfor
endfunction
```

```
let s:suite = themis#suite('WebDevIconsGetFileTypeSymbol')
let s:assert = themis#helper('assert')

function! s:Assert(filename, icon)
  call s:assert.equals(WebDevIconsGetFileTypeSymbol(a:filename), a:icon)
endfunction

function! s:suite.__OneArgument_VimIcon__()
  let targetfilenames = ['test.vim', 'vimrc', 'gvimrc']
  let expecticon = "\ue62b"
  let child = themis#suite('OneArgument_VimIcon')

  for targetfilename in targetfilenames
    let child[targetfilename] = funcref('s:Assert', [targetfilename, expecticon])
  endfor
endfunction
```

```vim
let s:suite = themis#suite('WebDevIconsGetFileTypeSymbol')
let s:assert = themis#helper('assert')

function! s:suite.OneArgumet_GetVimIcon()
  let targetfilenames = ['test.vim', '.vimrc', 'gvimrc']

  for targetfilename in targetfilenames
    call s:assert.equals(WebDevIconsGetFileTypeSymbol(targetfilename), "\ue62b")
  endfor
endfunction
```

```vim
let s:suite = themis#suite('WebDevIconsGetFileTypeSymbol')
let s:assert = themis#helper('assert')

function! s:Assert(filename, icon)
  call s:assert.equals(WebDevIconsGetFileTypeSymbol(a:filename), a:icon)
endfunction

function! s:suite.__OneArgument_VimIcon__()
  let targetfilenames = ['test.vim', 'vimrc', 'gvimrc']
  let expecticon = "\ue62b"
  let child = themis#suite('OneArgument_VimIcon')

  for targetfilename in targetfilenames
    let child[targetfilename] = funcref('s:Assert', [targetfilename, expecticon])
  endfor
endfunction
```

```
WebDevIconsGetFileTypeSymbol
  OneArgument_VimIcon
    [✓] test.vim
    [✓] vimrc
    [✓] gvimrc

tests 3
passes 3
```

```
let s:suite = themis#suite('WebDevIconsGetFileTypeSymbol')
let s:assert = themis#helper('assert')

[illegible]VimIcon()
  let targetfilenames = ['test.vim', '.vimrc', 'gvimrc']

  for targetfilename in targetfilenames
    call s:assert.equals(WebDevIconsGetFileTypeSymbol(targetfilename), "\ue62b")
  endfor
endfunction
```

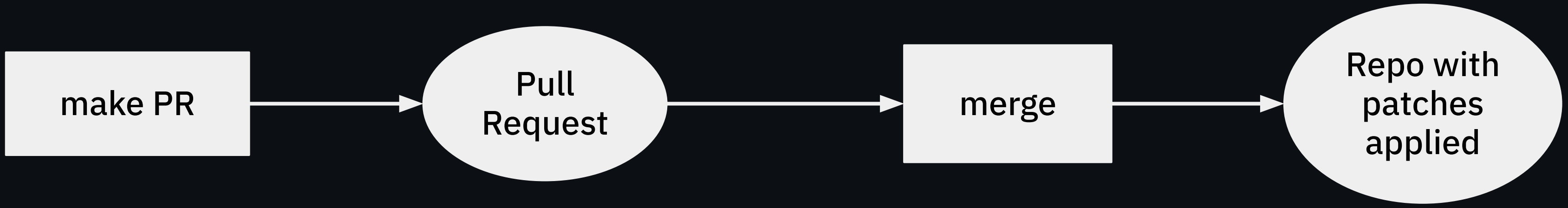
```
WebDevIconsGetFileTypeSymbol
  [✓] OneArgumet_GetVimIcon

tests 1
passes 1
fails 0
```

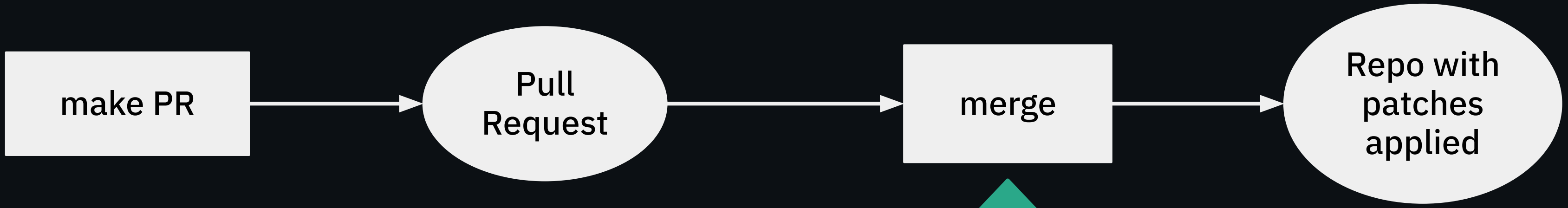
```
let s:suite = themis#suite('WebDevIconsGetFileTypeSymbol')
let s:assert = themis#helper('assert')

function! s:Assert(filename, icon)
  call s:assert.equals(WebDevIconsGetFileTypeSymbol(a:filename), a:icon)
endfunction

function! s:suite.__OneArgument_VimIcon__()
  let targetfilenames = ['test.vim', 'vimrc', 'gvimrc']
  let expecticon = "\ue62b"
  let child = themis#suite('OneArgument_VimIcon')

  for targetfilename in targetfilenames
    let child[targetfilename] = funcref('s:Assert', [targetfilename, expecticon])
  endfor
endfunction
```

```
WebDevIconsGetFileTypeSymbol
  OneArgument_VimIcon
    [✓] test.vim
    [✓] vimrc
    [✓] gvimrc

tests 3
passes 3
```

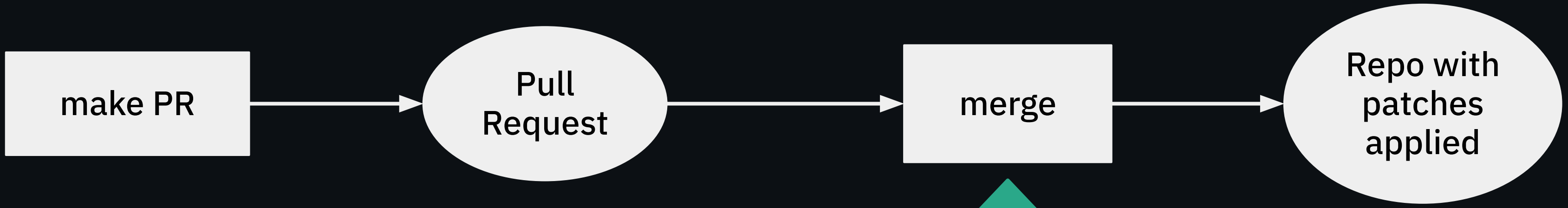
Outline

1. Introduction to Simple Testing
2. Selection/Usage of a Testing Framework
3. Points to Consider When Writing Tests
4. Efficient Flow for Easing Maintenance Starting from Tests
5. Conclusion

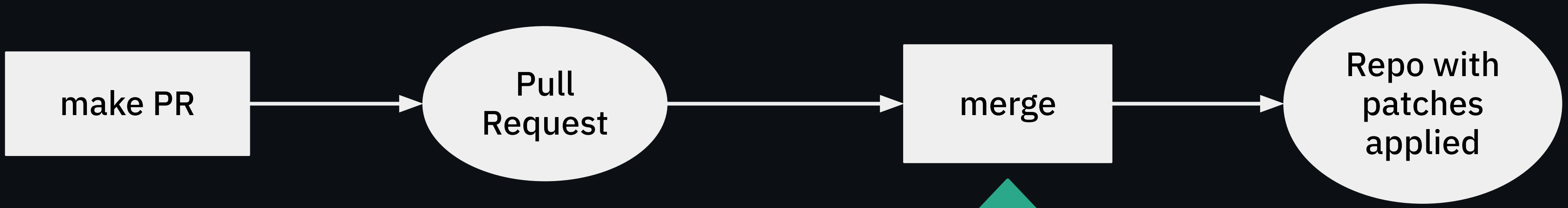

make PR
Pull Request
merge
Repo with patches applied



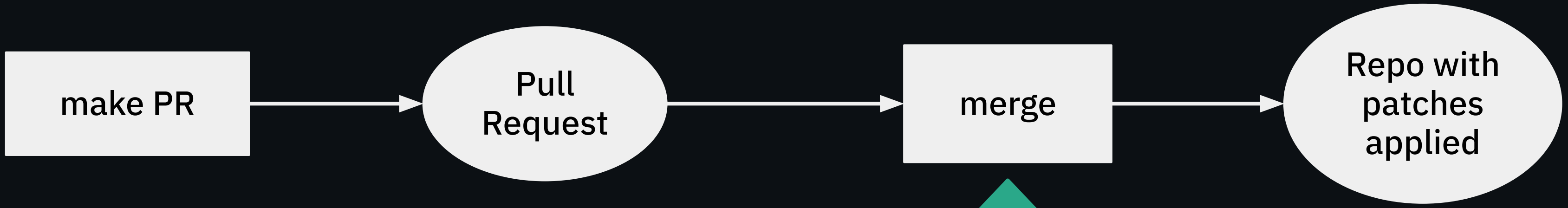
✅ Does the plugin work in the first place?



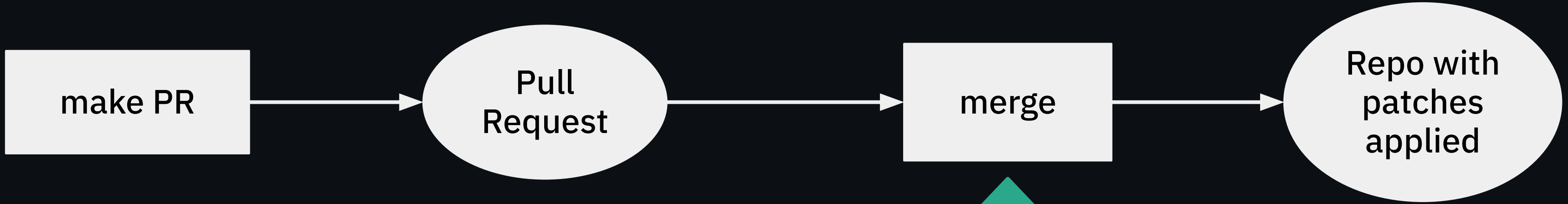
✅ Does the plugin work in the first place?
✅ Is there no regression?



✅ Does the plugin work in the first place?
✅ Is there no regression?
✅ Does it meet the requirements outlined in the PR summary?



✅ Does the plugin work in the first place?
✅ Is there no regression?
✅ Does it meet the requirements outlined in the PR summary?
✅ Is there no negative impact on performance?



✅ Does the plugin work in the first place?
✅ Is there no regression?
✅ Does it meet the requirements outlined in the PR summary?
✅ Is there no negative impact on performance?

maintainer

Do we have to do all of this manually!?!?

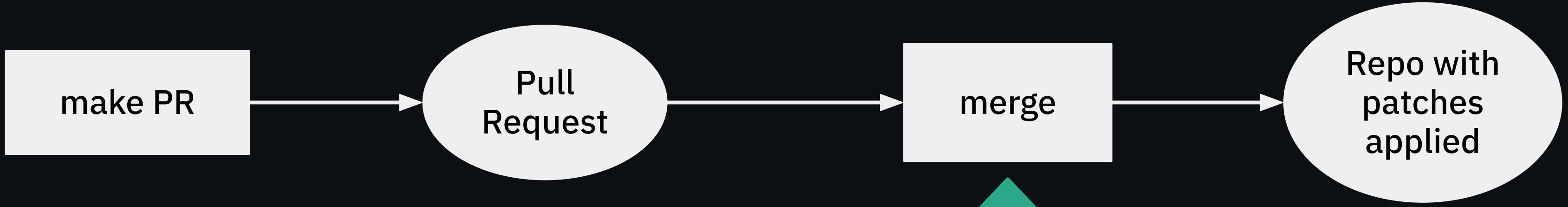
YOU DIED

This is not something a person should do

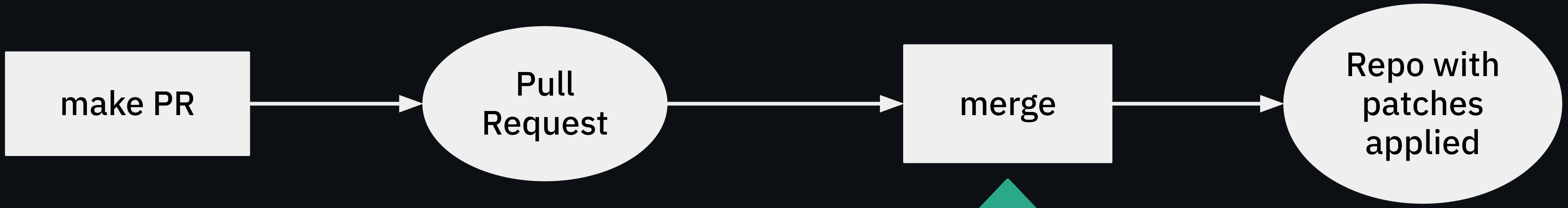
This is not something a person should do

▼

Let's automate it

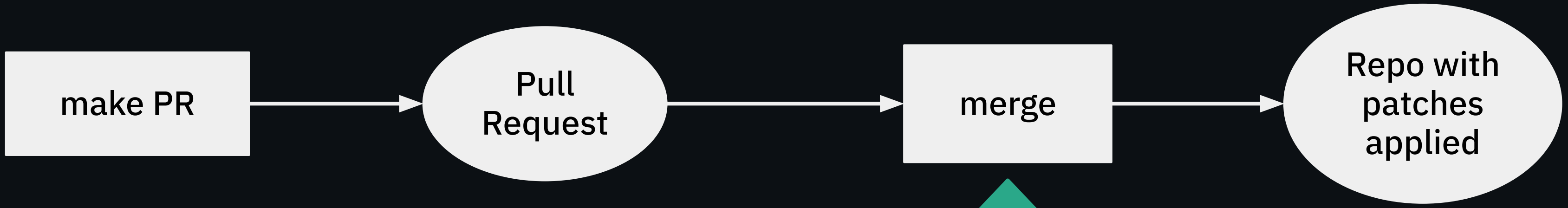


✅ Does the plugin work in the first place?
✅ Is there no regression?
✅ Does it meet the requirements outlined in the PR summary?
✅ Is there no negative impact on performance?



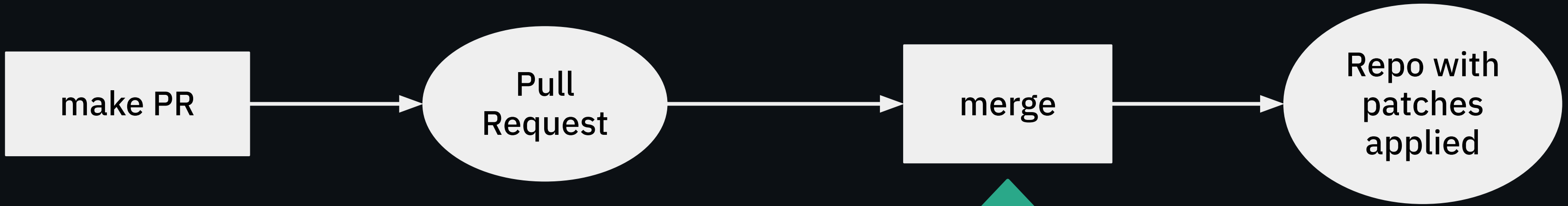
It seems automatable

- ✅ Does the plugin work in the first place?
- ✅ Is there no regression?
- ~~✅ Does it meet the requirements outlined in the PR summary?~~
- ~~✅ Is there no negative impact on performance?~~



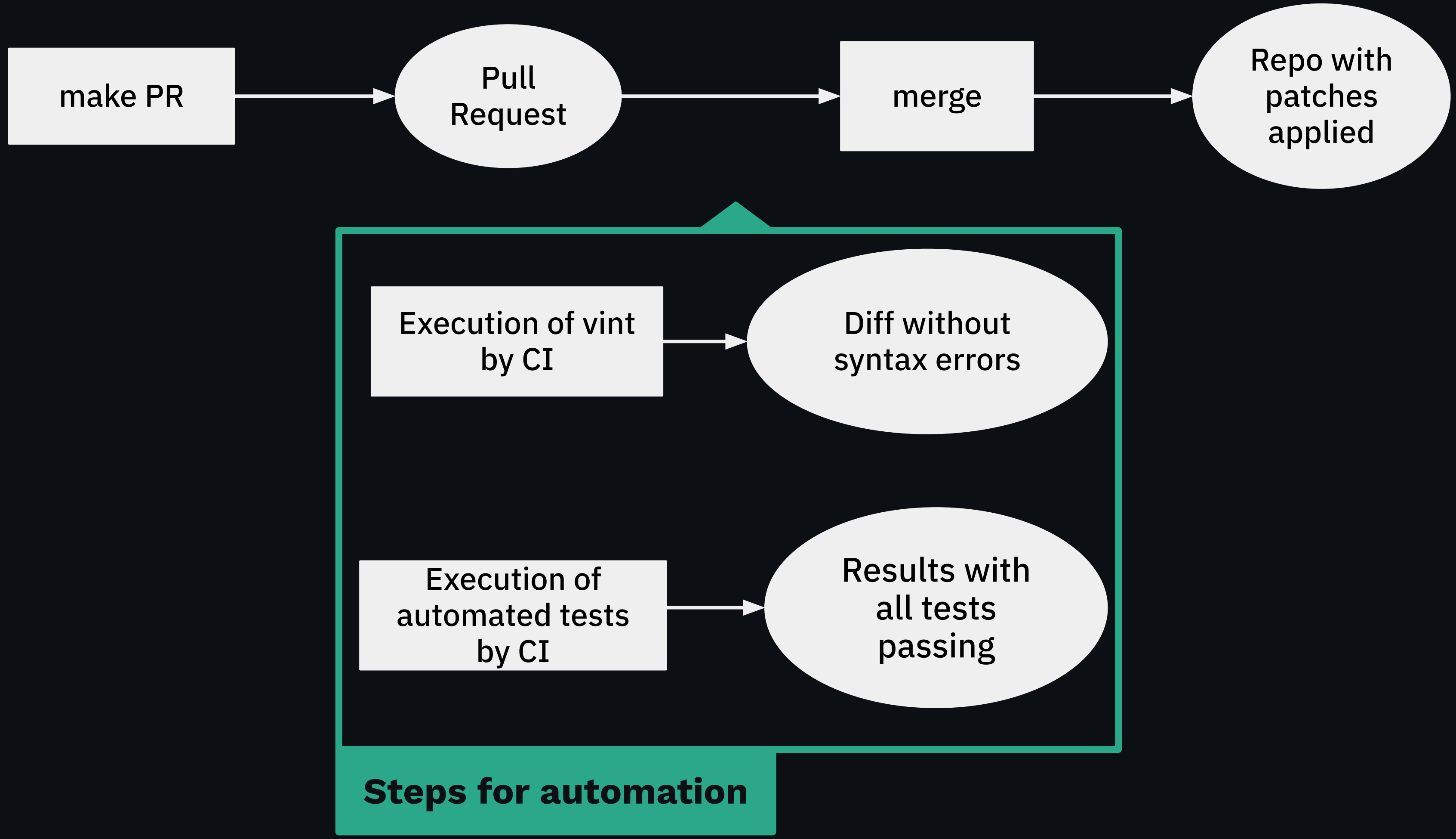
Static analysis is effective

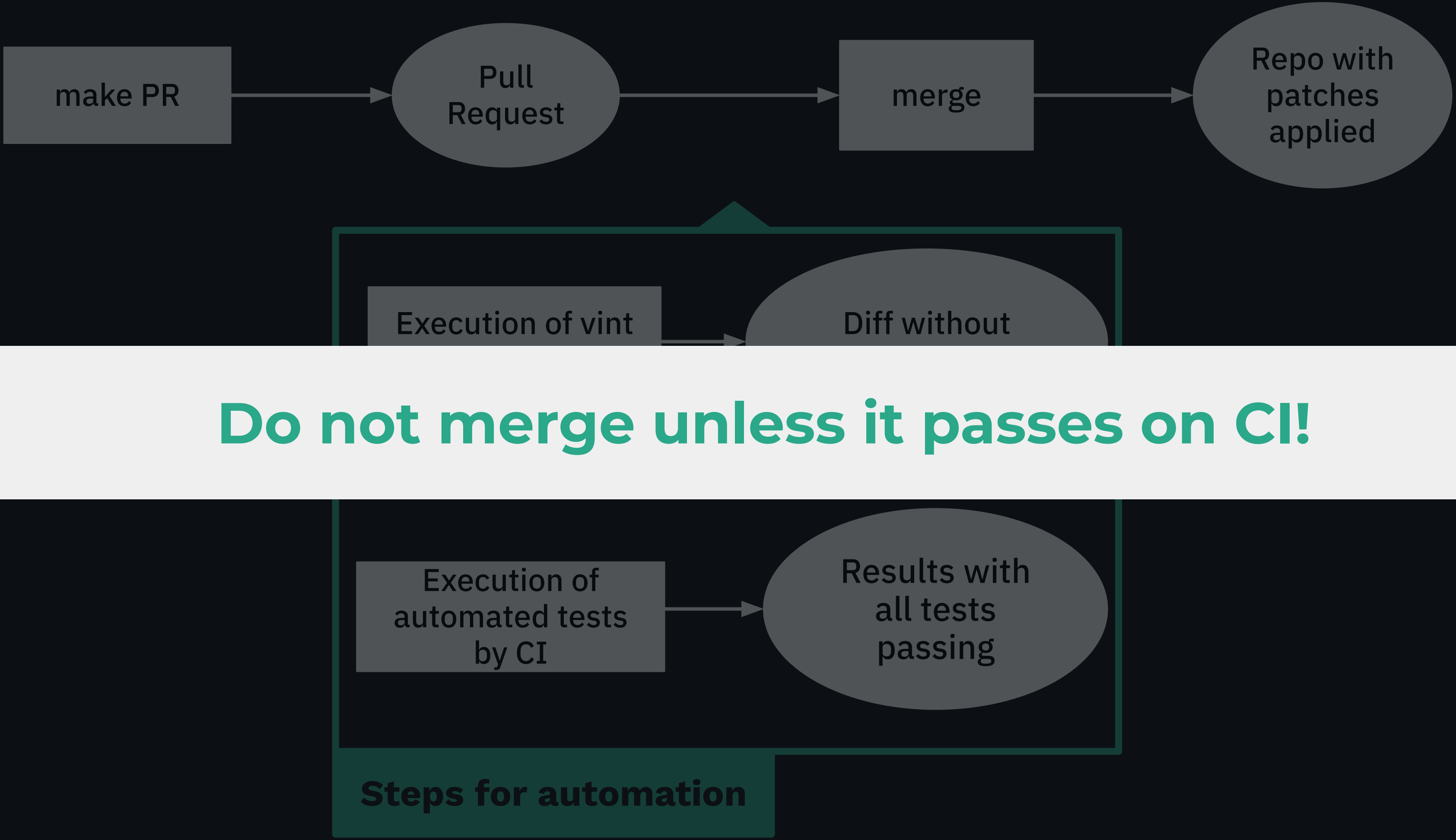
- ✅ Does the plugin work in the first place?
- ✅ Is there no regression?
- ~~✅ Does it meet the requirements outlined in the PR summary?~~
- ~~✅ Is there no negative impact on performance?~~



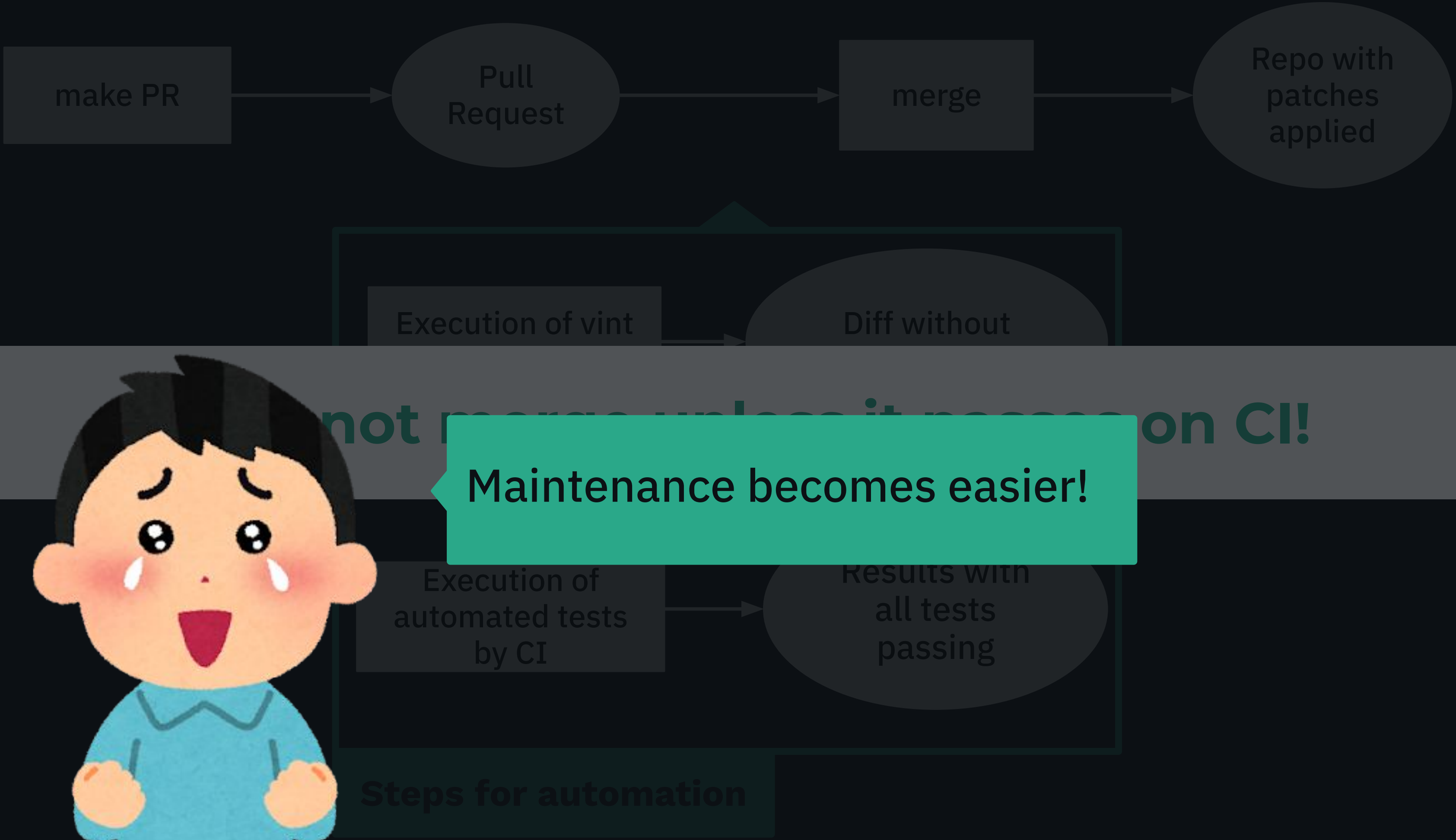
Executing automated tests is effective

- ✅ Does the plugin work in the first place?
- ✅ Is there no regression?
- ✅ ~~Does it meet the requirements outlined in the PR summary?~~
- ✅ ~~Is there no negative impact on performance?~~


make PR
Pull Request
merge
Repo with patches applied
Execution of vint by CI
Diff without syntax errors
Execution of automated tests by CI
Results with all tests passing
Steps for automation


make PR
Pull Request
merge
Repo with patches applied
Execution of vint
Diff without
Execution of automated tests by CI
Results with all tests passing
Steps for automation

Do not merge unless it passes on CI!


make PR
Pull Request
merge
Repo with patches applied
Execution of vint
Diff without
Execution of automated tests by CI
Results with all tests passing
Steps for automation

not merge unless it passes on CI!

Maintenance becomes easier!

Outline

Conclusion

- Nowadays, there are various ways to create Vim plugins
  - Deno, Lua, Vim script, Vim9 script, etc...
- Although Vim script is used as the example here, the basic concepts are the same for all.
- You can contribute to OSS through testing as well.

Conclusion

- Nowadays, there are various ways to create Vim plugins

I encourage you to try writing test!

concepts are the same for all.

- You can contribute to OSS through testing as well.

終

:DeNA

DeNA × AI Day || DeNA TechCon 2025

2025年2月5日 オンラインにて開催します!!

イベントサイトは12月上旬公開予定!!
X 公式アカウントにて告知しますのでフォローお願いします。

@DeNAxTech