Haskell IDE Support via LSP

HaskellX 2018

Overview

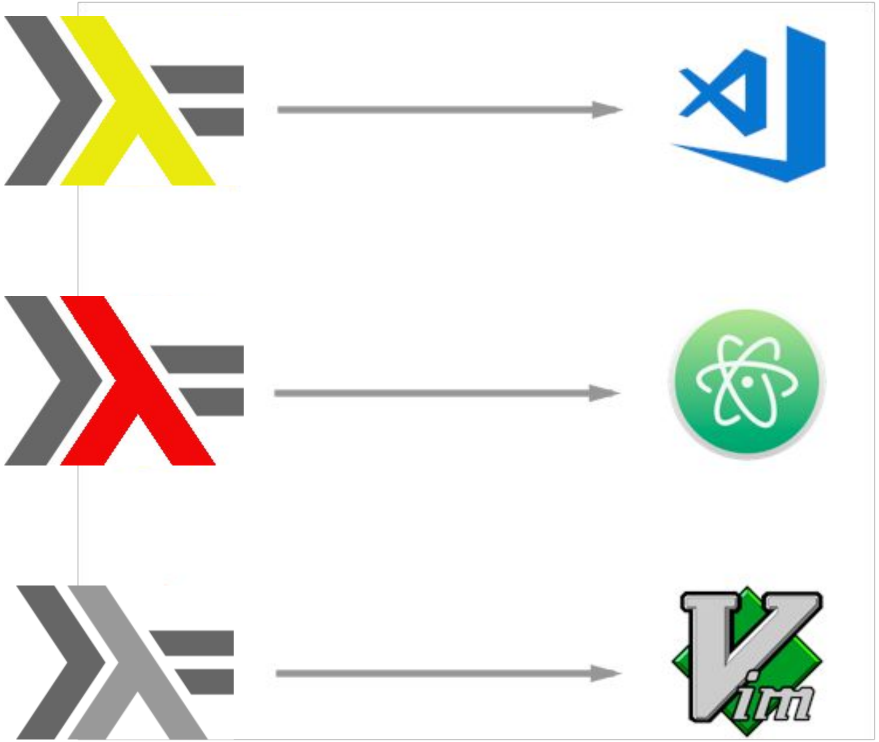
The Problem

How the Language Server Protocol Solves It

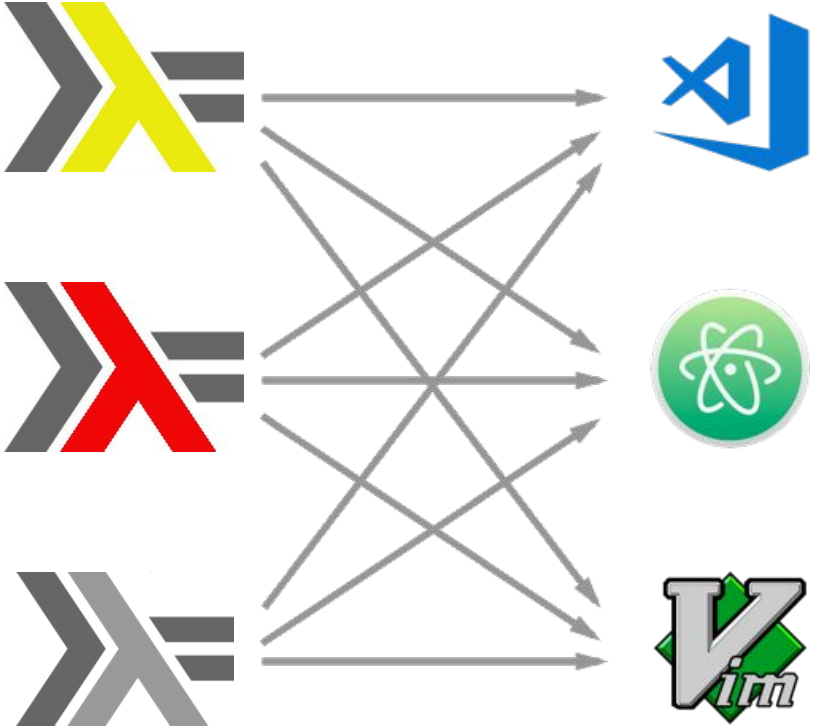
Why it Matters

Call to Action

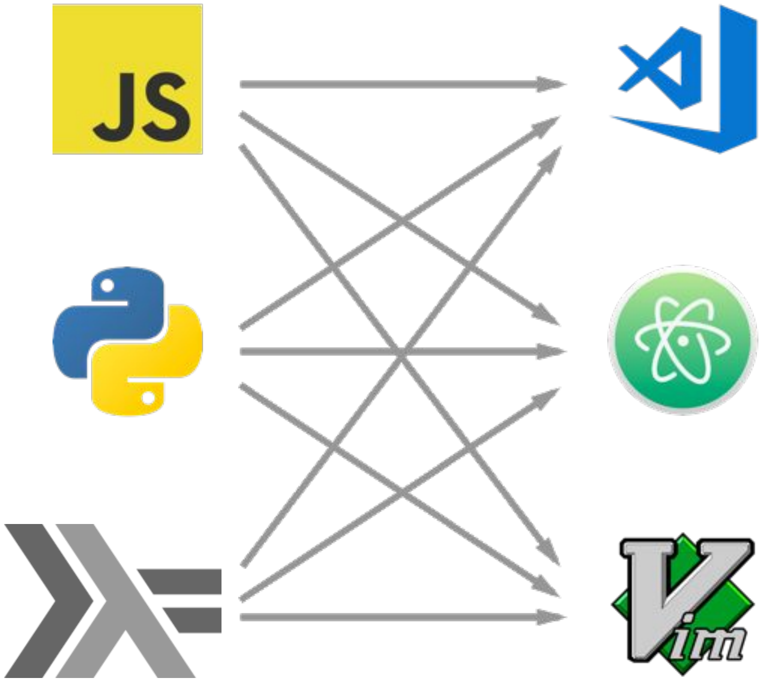
What’s the Problem?



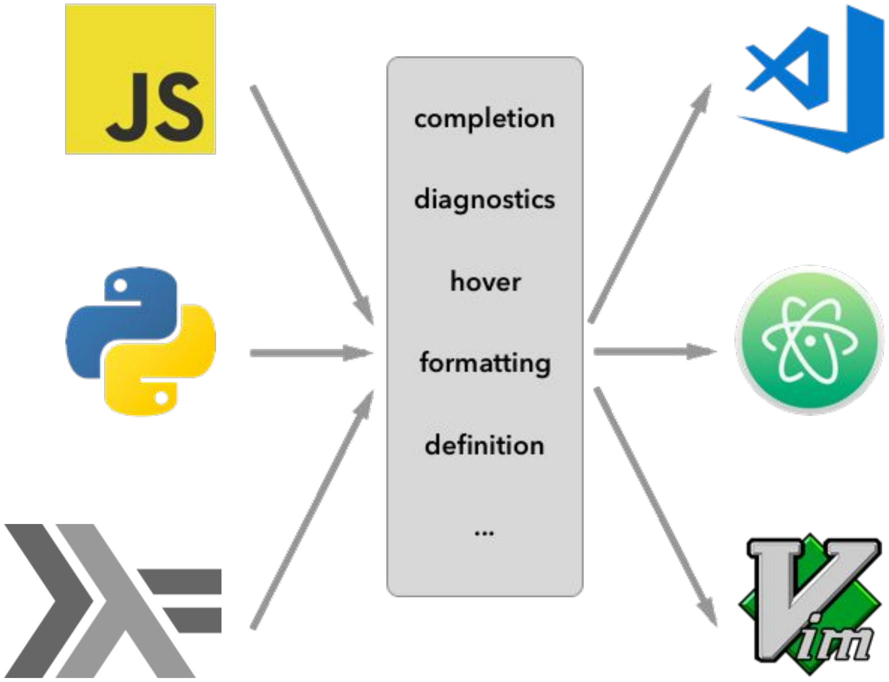
What’s the Problem?



NO LSP



LSP



completion

diagnostics

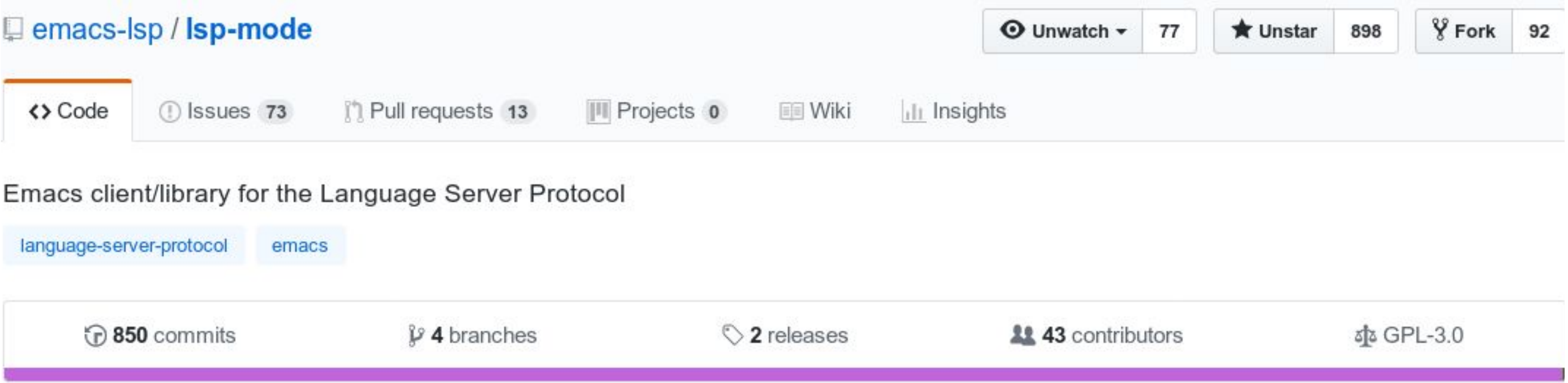
hover

formatting

definition

...

Emacs decoupling



emacs-lsp / lsp-mode

Emacs client/library for the Language Server Protocol

And there are 16 different language-specific variants too.

How it Works

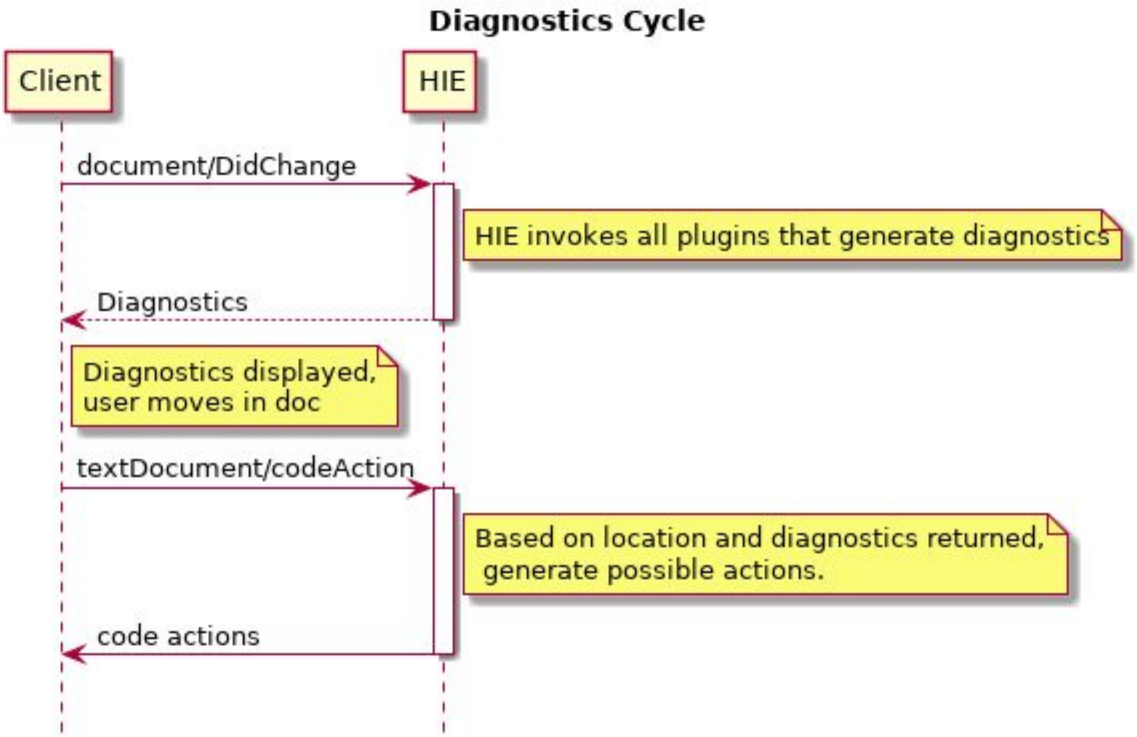
Demo

Operational Flow

Demo

drumroll

Operational Flow



Diagnostics Cycle

Client

HIE

document/DidChange

HIE invokes all plugins that generate diagnostics

Diagnostics

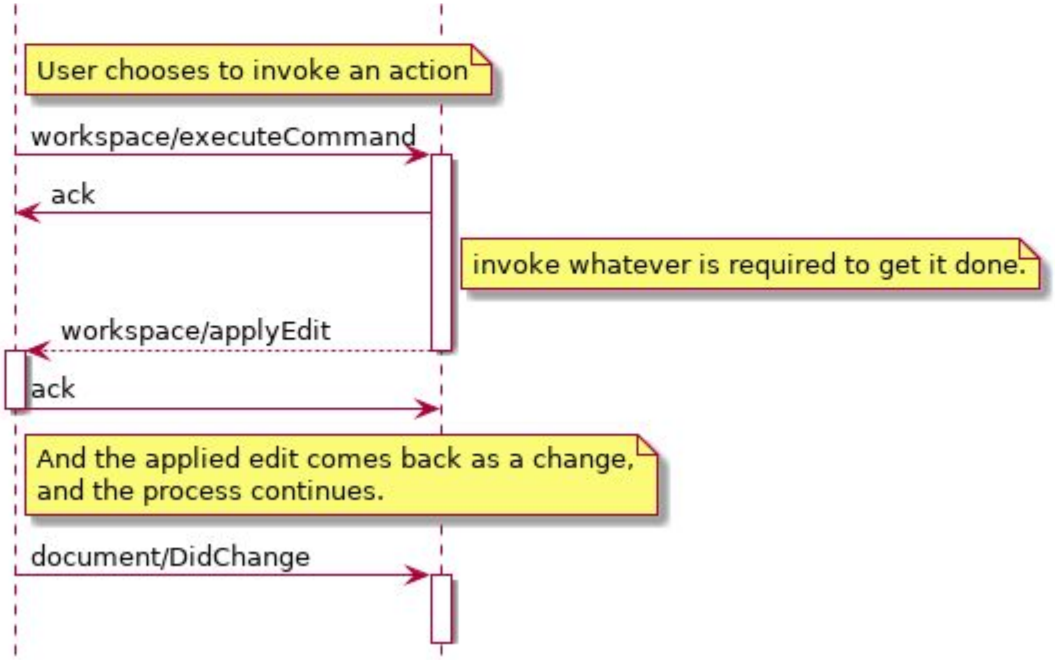
Diagnostics displayed,
user moves in doc

textDocument/codeAction

Based on location and diagnostics returned,
generate possible actions.

code actions

Operational Flow



User chooses to invoke an action

workspace/executeCommand

ack

invoke whatever is required to get it done.

workspace/applyEdit

ack

And the applied edit comes back as a change,
and the process continues.

document/DidChange

Why it Matters

It’s all about Context

Haskell IDE Engine architecture

HIE Plugins

It’s all about Context

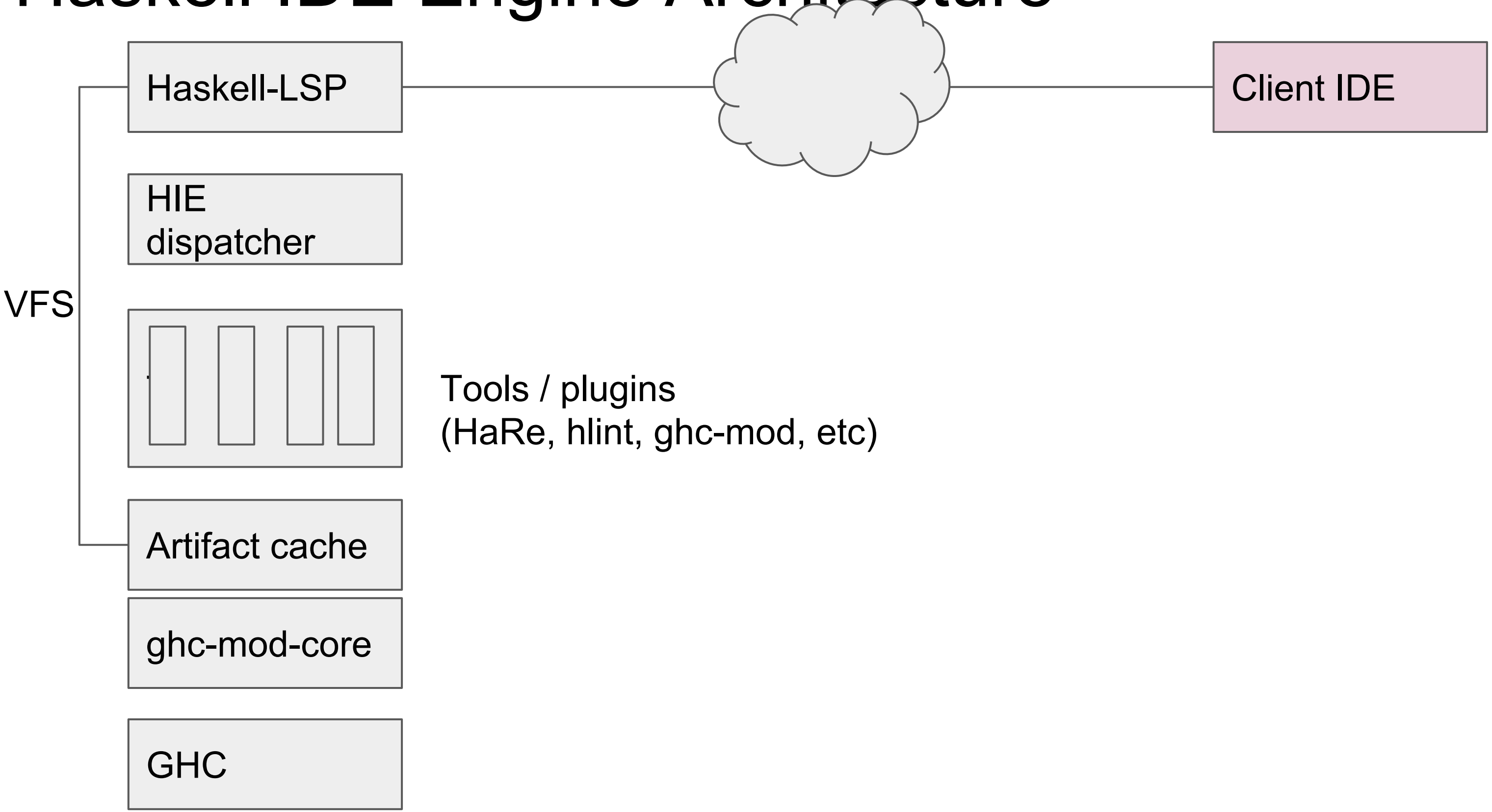
Project type

- Cabal
- Stack
- Nix
- Plain

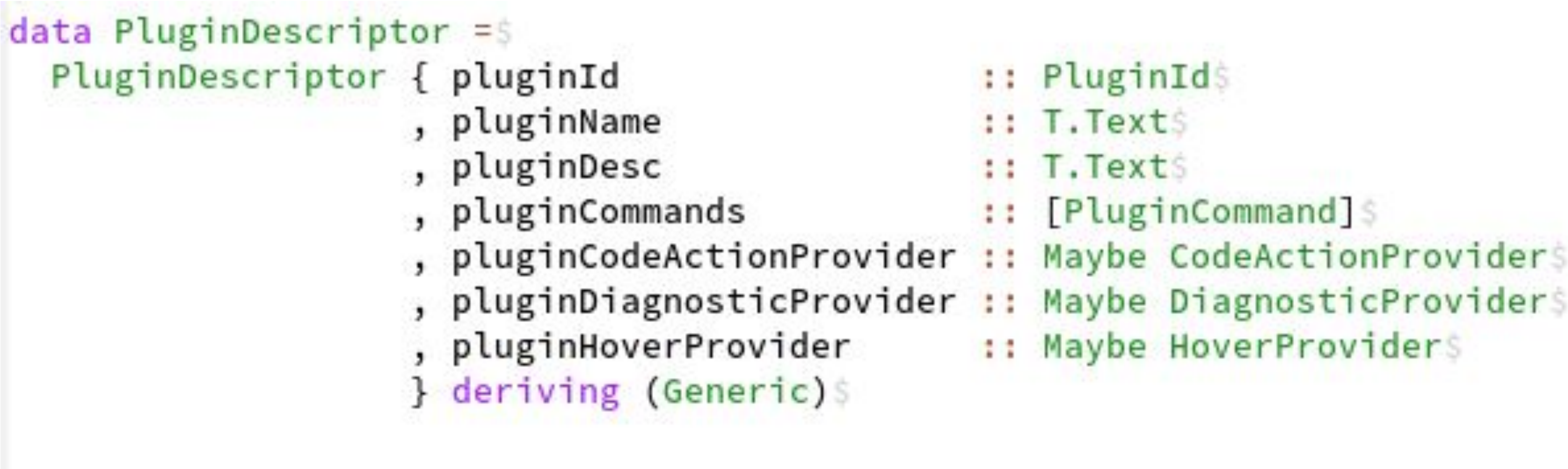
Operating System

- Linux (Debian / Ubuntu / Arch / …)
- Windows
- Mac
- Nixos
- ...

Haskell IDE Engine Architecture



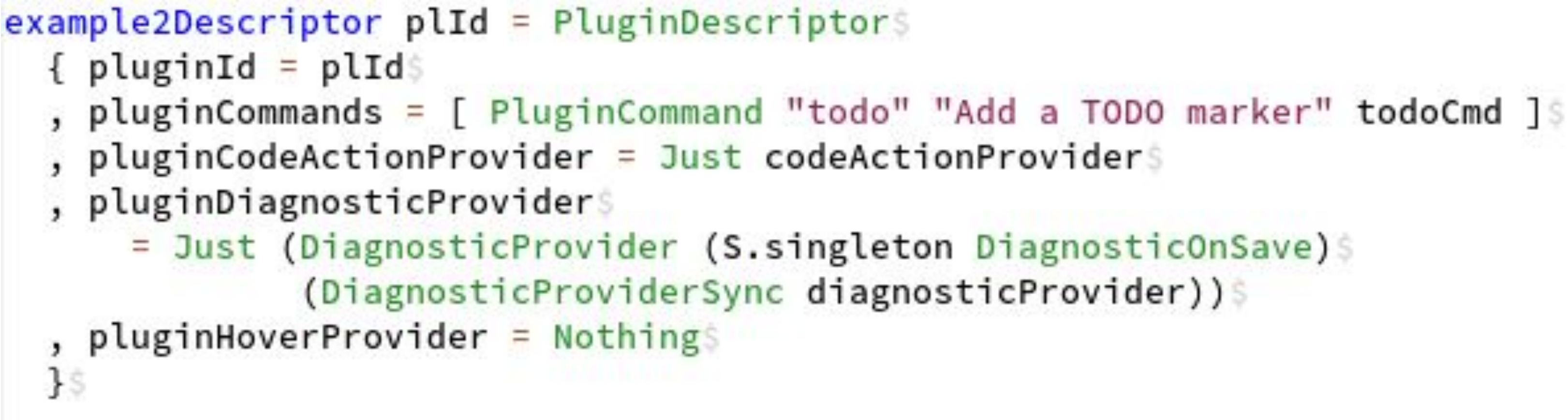
HIE Plugins



```haskell
data PluginDescriptor =$
  PluginDescriptor { pluginId                 :: PluginId$
                   , pluginName               :: T.Text$
                   , pluginDesc               :: T.Text$
                   , pluginCommands           :: [PluginCommand]$
                   , pluginCodeActionProvider :: Maybe CodeActionProvider$
                   , pluginDiagnosticProvider :: Maybe DiagnosticProvider$
                   , pluginHoverProvider      :: Maybe HoverProvider$
                   } deriving (Generic)$
```

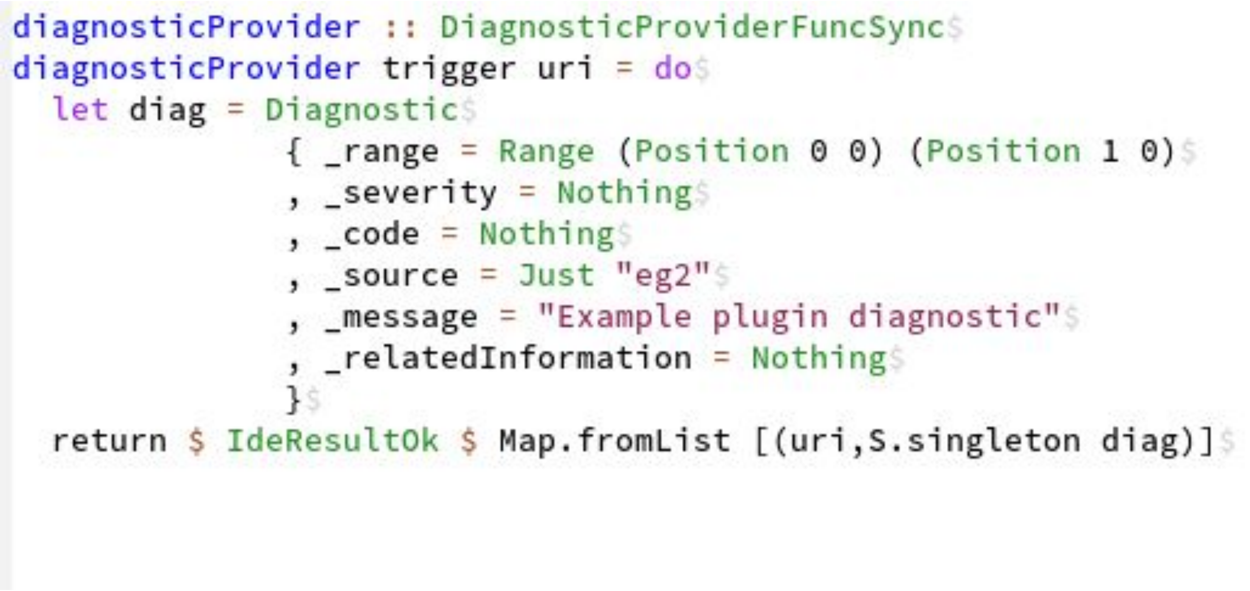
Example2 Demo

Example2 Plugin



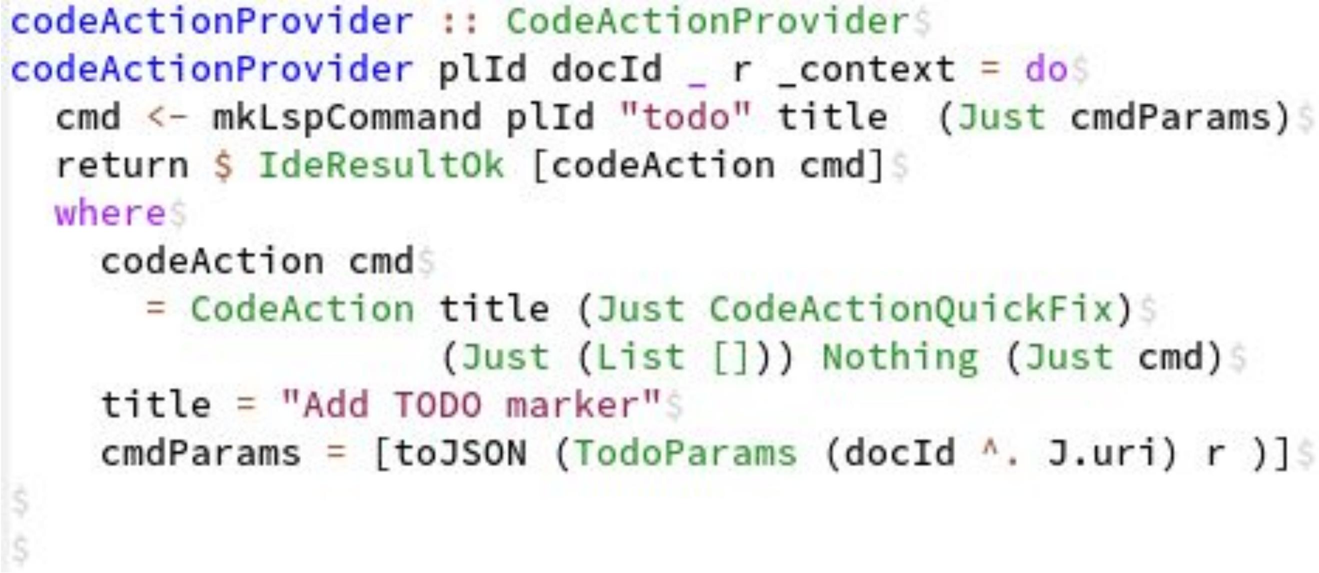
```
example2Descriptor plId = PluginDescriptor
  { pluginId = plId
  , pluginCommands = [ PluginCommand "todo" "Add a TODO marker" todoCmd ]
  , pluginCodeActionProvider = Just codeActionProvider
  , pluginDiagnosticProvider
      = Just (DiagnosticProvider (S.singleton DiagnosticOnSave)
                (DiagnosticProviderSync diagnosticProvider))
  , pluginHoverProvider = Nothing
  }
```

Example2 Plugin diagnosticProvider



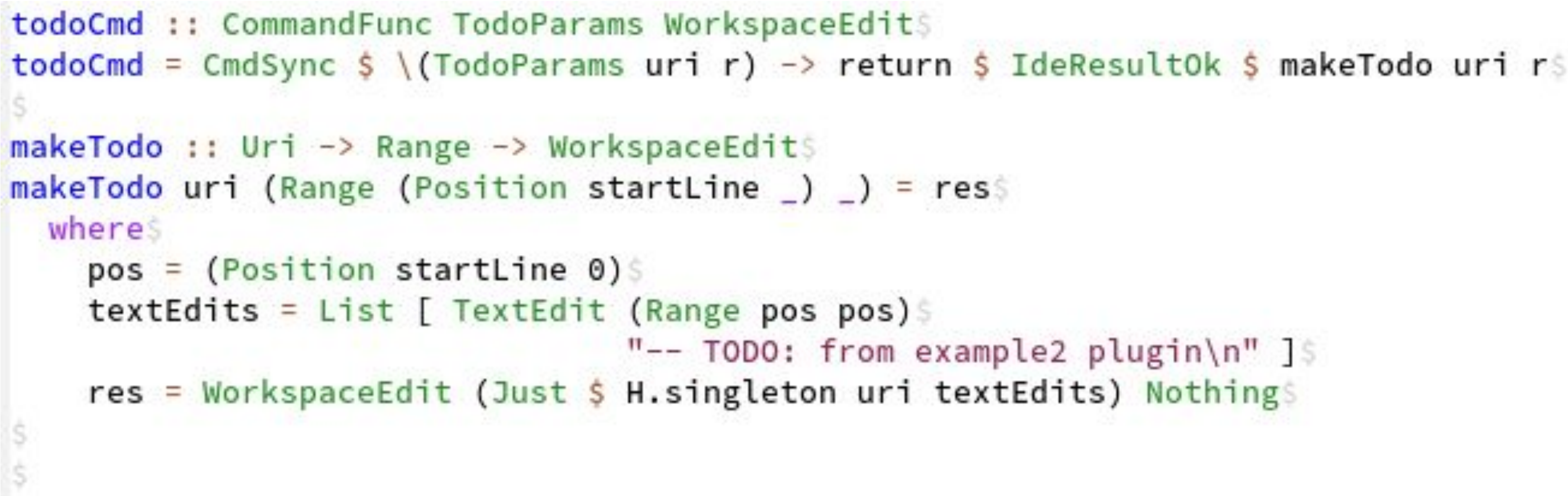
```
diagnosticProvider :: DiagnosticProviderFuncSync
diagnosticProvider trigger uri = do
  let diag = Diagnostic
              { _range = Range (Position 0 0) (Position 1 0)
              , _severity = Nothing
              , _code = Nothing
              , _source = Just "eg2"
              , _message = "Example plugin diagnostic"
              , _relatedInformation = Nothing
              }
  return $ IdeResultOk $ Map.fromList [(uri,S.singleton diag)]
```

Example2 Plugin codeActionProvider



```haskell
codeActionProvider :: CodeActionProvider
codeActionProvider plId docId _ r _context = do
  cmd <- mkLspCommand plId "todo" title  (Just cmdParams)
  return $ IdeResultOk [codeAction cmd]
  where
    codeAction cmd
      = CodeAction title (Just CodeActionQuickFix)
                   (Just (List [])) Nothing (Just cmd)
    title = "Add TODO marker"
    cmdParams = [toJSON (TodoParams (docId ^. J.uri) r )]
```

Example2 todo command



```
todoCmd :: CommandFunc TodoParams WorkspaceEdit$
todoCmd = CmdSync $ \(TodoParams uri r) -> return $ IdeResultOk $ makeTodo uri r$
$
makeTodo :: Uri -> Range -> WorkspaceEdit$
makeTodo uri (Range (Position startLine _) _) = res$
  where$
    pos = (Position startLine 0)$
    textEdits = List [ TextEdit (Range pos pos)$
                                "-- TODO: from example2 plugin\n" ]$
    res = WorkspaceEdit (Just $ H.singleton uri textEdits) Nothing$
$
$
```

Plugins

GHC Session

Deferred

With Cached

Async

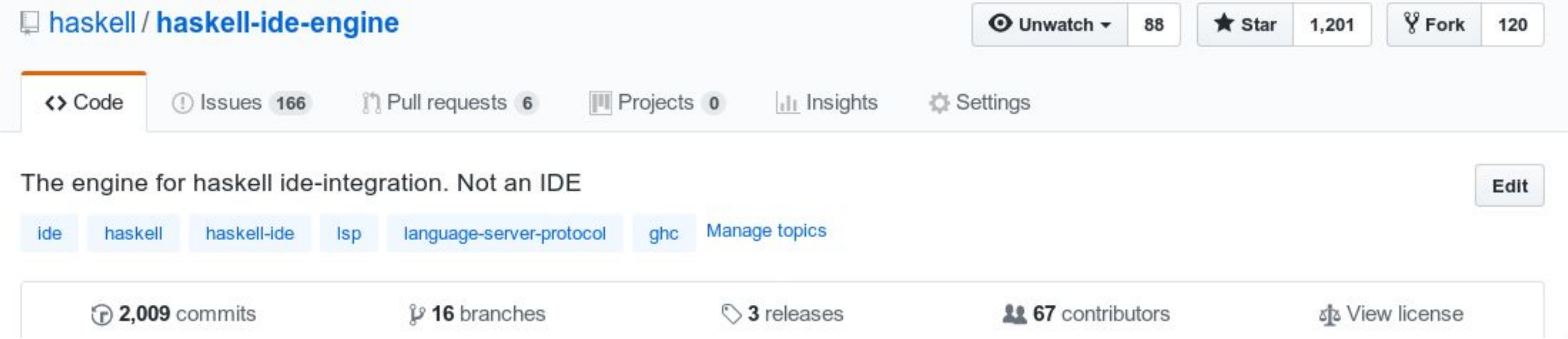
Demo : liquid haskell plugin

Call To Action

Plugins are Easy

Impact is High

Impact is High



haskell / haskell-ide-engine

The engine for haskell ide-integration. Not an IDE

2,009 commits

16 branches

3 releases

67 contributors

View license

Impact is High

GSOC 2018

Luke Lau - haskell-lsp-test

Zubin Duggal - Making GHC Tooling Friendly

Simon Jakobi - Hi Haddock

Shayan Najd - TTG

HSOC 2017

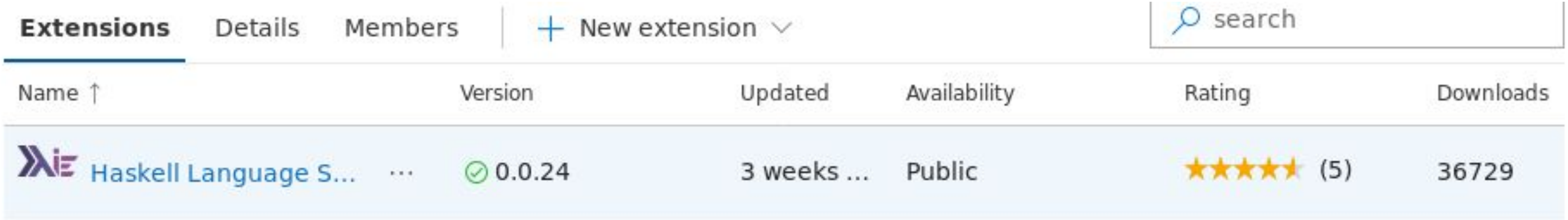
Zubin Duggal - Haskell IDE Engine

Impact is High

README install instructions for

VS Code
Sublime Text
Vim/Neovim
Atom
Oni
Emacs
Spacemacs

Impact is High



Extensions Details Members + New extension search

Name ↑	Version	Updated	Availability	Rating	Downloads
Haskell Language S... ···	✓ 0.0.24	3 weeks ...	Public	★★★★★ (5)	36729

Not all Rosy

No REPL

GHC Session Memory Leak

Not optimised for memory usage

No “new build” support

Not on hackage / stackage

Plugins are Easy Bonus

http://www.haskellforall.com/2018/10/detailed-walkthrough-for-beginner.html

Wrapup

Image credits:

https://code.visualstudio.com/docs/extensions/example-language-server

https://commons.wikimedia.org/w/index.php?curid=8479507

Links

Project home

https://github.com/haskell/haskell-ide-engine

Slides will be at

https://github.com/haskell/haskell-ide-engine/tree/master/docs/HaskellX2018.pdf

Video via SkillsMatter at

https://skillsmatter.com/conferences/10237-haskell-exchange-2018#skillscasts