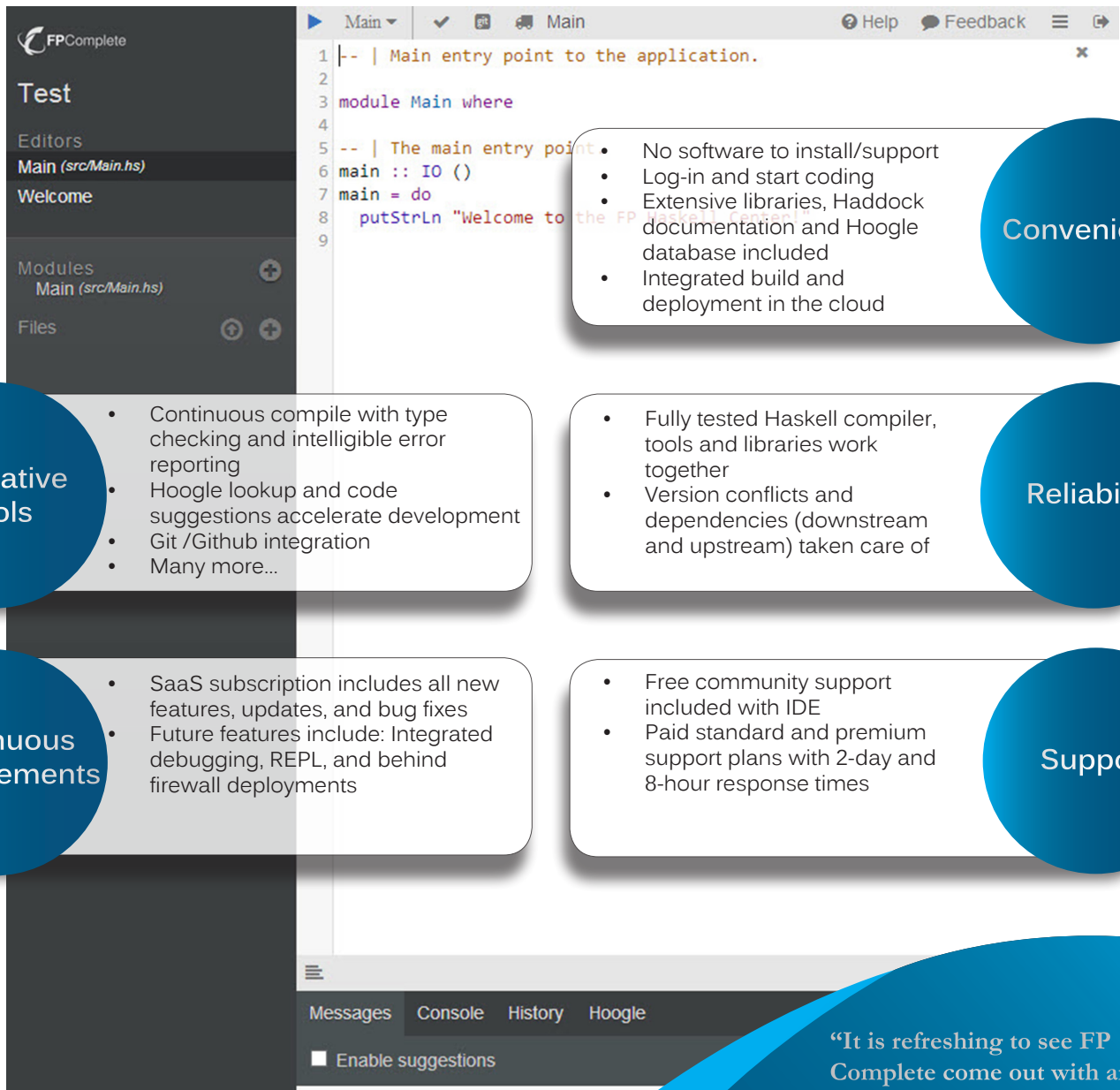


Enhancing the Haskell Experience

FP Haskell Center is the world's first commercial Haskell IDE and deployment platform. It is designed to enable commercial developers, researchers, students, as well as independent developers from beginners to experts to enjoy the immense capabilities of Haskell in the most intuitive and efficient way. It has a full package of benefits that comes with a commercial product that's reliable, supported and continuously enhanced. These benefits are necessary to make FP Haskell Center the platform of choice for Haskell developers and to propel Haskell into a mainstream language.



Innovative Tools

- Continuous compile with type checking and intelligible error reporting
- Hoogle lookup and code suggestions accelerate development
- Git /Github integration
- Many more...

Continuous Enhancements

- SaaS subscription includes all new features, updates, and bug fixes
- Future features include: Integrated debugging, REPL, and behind firewall deployments

- No software to install/support
- Log-in and start coding
- Extensive libraries, Haddock documentation and Hoogle database included
- Integrated build and deployment in the cloud

Convenience

- Fully tested Haskell compiler, tools and libraries work together
- Version conflicts and dependencies (downstream and upstream) taken care of

Reliability

- Free community support included with IDE
- Paid standard and premium support plans with 2-day and 8-hour response times

Support

“It is refreshing to see FP Complete come out with an IDE that is simple, elegant, and just works.”

Armando Blancas
Tools Engineer, Carrier IQ

Turnkey Haskell

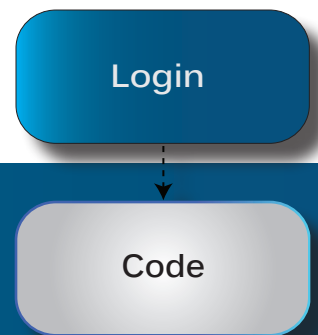
Simply Login and Code

FP Haskell Center™ makes life easier for Haskell programmers and Haskell more accessible to all. It eliminates the complicated and time-consuming process of getting started and delivers Haskell to you ready to work. Your job is to enjoy the great benefits of Haskell with minimal barriers. Our job is to enable that.

Current Haskell Setup



FP Haskell Center



If you are more interested in writing applications than playing with tools, the FP Haskell IDE is a really nice solution.”

Simon Bergot
R & D Engineer, efront



Modules Made Easy

FP Haskell Center™ Module Features

- View projects as a Hierarchy of Haskell Modules. Adding, removing, or moving modules automatically updates the definition of a Haskell project. There are no project files to edit. The IDE gives you GHC's view of the project as a collection of modules, not a collection of files. Modules are imported by their name, not file location.
- Use/import available libraries without first specifying them in the cabal file.
- Add/rename/delete modules and files through drag-and-drop operations.
- Support for multiple main modules and selecting which main is the current target allowing developers to switch between production and test builds of projects.
- Uploading/downloading of binary data and source files.
- Convert existing projects into our IDE and see them in module view.
- Module-based editing so you can easily find, import and manage the modules you need when you need them.

The screenshot displays the FP Complete IDE interface with three overlapping panels:

- Add a Module:** A panel on the left showing the 'yesod-web-service-template' project. It includes a 'Settings' tab, a 'Modules' tree with 'Foundation' and 'Handler' sub-trees, and an 'Import' section for 'Main (src/Main.hs)'. Buttons for 'Save' and 'Move to Files' are visible.
- Project Module Template:** A central panel showing a text editor with a template: 'module MODULE_NAME where'. Below it, the 'Environment' section is set to 'Stable 13.09', and the 'Compiler Extensions' section is empty.
- Collapsible Module Tree:** A panel on the right showing a hierarchical tree of modules. It includes 'Editors', 'Modules', and 'Files' sections. The 'Modules' section is expanded to show 'Foundation', 'Handler', 'Fib', 'Home', 'Markdown', and 'Import Main (src/Main.hs)'.



Innovative Tools

FP Haskell Center™ Features

Haskell developers currently need to cobble together mostly open-source or home-grown tools in order to program in Haskell. FP Haskell Center brings all the components you require into one platform with a full-featured Haskell code editor based on the popular codemirror editor. It also supports both console and web-based applications allowing you to switch between the two. Now, for the first time, you can learn, develop, build and deploy in Haskell from one reliable interface. FP Haskell Center saves you a lot of valuable time, energy and effort *getting to* Haskell so you can focus on using Haskell.

Type Information & Error Messaging

Type information, documentation, and compile time errors at your fingertips. Clear and concise error messages capture code errors and pinpoint their location prior to compiling as you write. Continuous compilation provides live updating of syntax and grammar errors as well as immediate feedback. Hover over variables and see type code.

Type Information

The screenshot displays the FP Complete Haskell IDE interface. On the left, a sidebar shows the project structure for 'Yesod web service', including editors like 'Foundation', 'Handler.Fib', and 'Main (src/Main.hs)'. The main editor shows Haskell code for a 'Foundation' module. A tooltip 'Hover over variables' is shown over the variable 'main' in the code, displaying its type signature 'main :: Int'. Below the code editor, a 'Messages' panel shows error messages: 'Right :: b -> Either a b' and two type mismatch errors: 'Main@2:1-2:1 Couldn't match expected type `IO t0' with actual type `Int`' and 'Main@2:8-2:16 Couldn't match expected type `Int' with actual type `[a0]''. A second tooltip 'See type code' is shown over the error messages.

Tutorials and Code Samples

An interactive learning center rich with tools designed to help you create content as well as tutorials which build and run is embedded directly into the editor. Our tutorials are designed to be useful for a broad class of business applications and to help developers solve common problems. You can build a working application, one that runs as-is, or modified for another. Active code examples and exercises are contributed by both FP Complete and the Haskell community.



Innovative Tools

Haddocks Documentation and Hoogle Search

Programming aids display important information about identifiers such as definition, Haddocks documentation and Hoogle search for functions and data types. Built-in Hlint and stylish-Haskell make Haskell programs easier to read.

Haddock Information from Line Item Type

The screenshot shows the Haddock documentation for the `Right` constructor. It lists various instances and type signatures, such as `Typesable2 Either`, `Monad (Either e)`, and `Functor (Either a)`. Below the list, the definition of `either` is shown: `either :: (a -> c) -> (b -> c) -> Either a b -> c`. A brief description follows: "Case analysis for the `Either` type. If the value is `Left` `a`, apply the first function to `a`; if it is `Right` `b`, apply the second function to `b`."

The screenshot shows the Hoogle search interface. The search term is `Right`, and 10 results are displayed. The results include the definition of `Right` and several higher-order arrow applications, such as `RightArrApp` and `RightArrHighApp`. The interface also shows a sidebar with project navigation and a code editor with Haskell code for the `Foundation` module.

Team Management Features

- Team accounts allow developers to share deployment resources and manage users.
- Team Dashboard for managing access control and licenses.
- Shared projects provides fast access to team- shared code and data.
- Online access provides support for distributed and dispersed project teams.
- Screen sharing to enable mentoring.
- Unified billing for all licenses and resources.



Stable Libraries

All developers are familiar with the tedious process of resolving library dependency and version conflicts. Haskell is no exception. Any change in your library set will most likely create a ripple effect of changes downstream and upstream that will require you to spend time and effort to ensure the whole set works as before, until the next change, and on it goes. We eliminate that cycle for you by doing the work ourselves so you can focus on programming.

All libraries within FP Haskell Center are continuously vetted, updated and maintained by our engineers with input from users and Haskell community members.

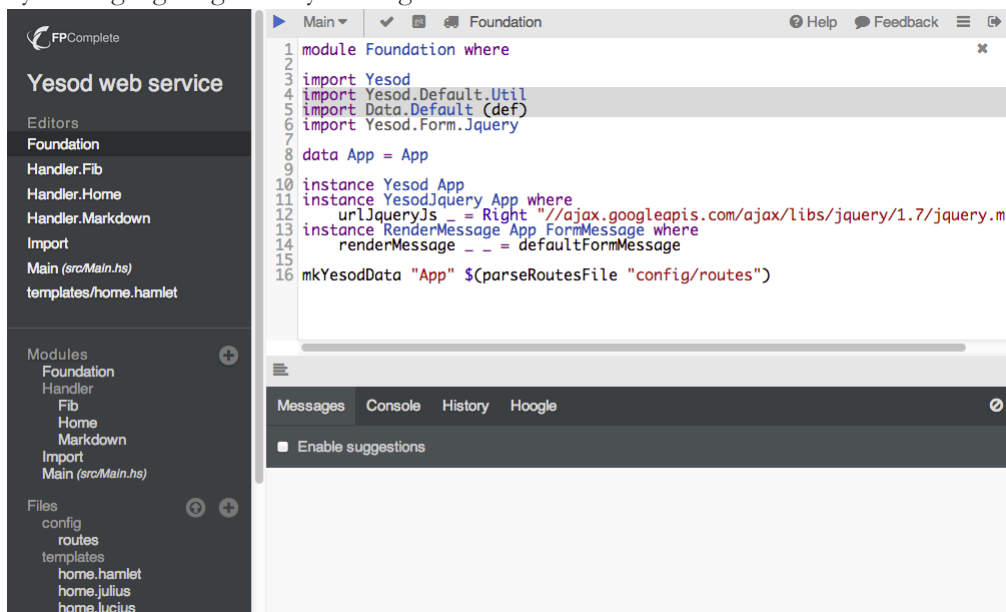
- Includes 400+ and continually growing set of libraries.
- You can import your own libraries.
- Libraries and bug fixes updated on a regular basis.
- Snapshots are taken of each library set and provided as options to users.
- At no point do you have to manually install any packages. System libraries are available, new versions are installed, dependencies and conflicting versions resolved automatically.
- You can write software without having to worry about rewriting code to stay up-to-date with security and bug fixes.
- ALL packages only have one version, and all build and function with each other.

Recommended Solutions

For your convenience, we have selected some libraries as recommended solutions to specific problem domains. The purpose here is to provide new users with a simple path from “I have a problem” to “I have a solution.”

Syntax Highlighting and Key Binding

Syntax highlighting and key bindings for Vim and Emacs.



```
1 module Foundation where
2
3 import Yesod
4 import Yesod.Default.Util
5 import Data.Default (def)
6 import Yesod.Form.Jquery
7
8 data App = App
9
10 instance Yesod App
11 instance YesodjQuery App where
12   urljQueryJs _ = Right "//ajax.googleapis.com/ajax/libs/jquery/1.7/jquery.mi
13 instance RenderMessage App FormMessage where
14   renderMessage _ _ = defaultFormMessage
15
16 mkYesodData "App" $(parseRoutesFile "config/routes")
```



Git/GitHub Integration

Tight integration with git and GitHub provides a quick and convenient local code repository. Edited files are automatically saved locally and can be committed to the local repository in two clicks. Users can view the “commit” and restore the repository to a pre-commit state.

Git Menu

The screenshot shows the FPComplete IDE interface. On the left is a sidebar with a 'Git Menu' containing options like 'Reset', 'Push', 'Pull', 'Publish', 'Unpublish', 'GitHub', 'Remote', and 'Branches'. The main editor area shows a file named 'Handler.Markdown'. Below the editor is a terminal window with the following output:

```
renderHtml :: Html -> Data.Text.Lazy.Internal.Text

Messages Console History Hoogle
[ ] Enable suggestions
bhudak cd /tmp
zsh: correct '/tmp' to '/tmp' [nyae]? y
bhudak git clone git@github.com:mwm/scratchpad.git /tmp
Cloning into 'scratchpad'...
remote: Counting objects: 45, done.
remote: Compressing objects: 100% (29/29), done.
remote: Total 45 (delta 12), reused 42 (delta 9)
Receiving objects: 100% (45/45), 4.71 KiB, done.
Resolving deltas: 100% (12/12), done.
bhudak cd scratchpad
bhudak ls
Application.hs Scratch src
README.md list.hs test.hs
bhudak echo "Test package" > README.md
bhudak git add README.md
bhudak git commit -m master
[master 9917cc8] master
1 file changed, 1 insertion(+), 1 deletion(-)
bhudak git push
warning: push.default is unset: its implicit value is changing in
Git 2.8 from 'matching' to 'simple'. To squelch this message
and maintain the current behavior after the default changes, use:
git config --global push.default simple
To squelch this message and adopt the new behavior now, use:
git config --global push.default simple
See 'git help config' and search for 'push.default' for further information.
(the 'simple' mode was introduced in Git 1.7.11. Use the similar mode
'current' instead of 'simple' if you sometimes use older versions of Git)
Counting objects: 5, done.
Delta compression using up to 12 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 261 bytes, done.
Total 3 (delta 1), reused 0 (delta 0)
To git@github.com:mwm/scratchpad.git
d19b6c8..9917cc8 master -> master
bhudak
```

Git Import

The screenshot shows the 'Git Import' dialog box. It contains several sections:

- Text Parsing:** A template for parsing server log files using the attoparsec package. Includes a 'Clone' button.
- Aeson tweet reader:** Uses OAuth to connect to Twitter's API, and read the last few tweets. It then parses them into a Haskell data structure using the Aeson package. Includes a 'Clone' button.
- Package upload log analyzer:** Includes a 'Clone' button.
- Yesod web service:** Includes a 'Clone' button.

 On the right side, there is a 'URL' field with the value 'https://github.com/fpc/stackage', a 'Warning!' box stating 'The entered URL is not writable by you. You will not be able to push to it. We can only push to SSH URLs.', and a 'Use git@github.com:foo/bar.git' button. Below that are 'Title' (stackage) and 'Branch' (master) fields, and a 'Clone' button. At the bottom, there is a note: 'If you are cloning from a GitHub repository, we recommend authorizing FP Haskell Center on GitHub so we can push to and pull from your repositories.'

The screenshot shows a terminal window with the following output:

```
bhudak cd /tmp
zsh: correct '/tmp' to '/tmp' [nyae]? y
bhudak git clone git@github.com:mwm/scratchpad.git /tmp
Cloning into 'scratchpad'...
remote: Counting objects: 45, done.
remote: Compressing objects: 100% (29/29), done.
remote: Total 45 (delta 12), reused 42 (delta 9)
Receiving objects: 100% (45/45), 4.71 KiB, done.
Resolving deltas: 100% (12/12), done.
bhudak cd scratchpad
bhudak ls
Application.hs Scratch src
README.md list.hs test.hs
bhudak echo "Test package" > README.md
bhudak git add README.md
bhudak git commit -m master
[master 9917cc8] master
1 file changed, 1 insertion(+), 1 deletion(-)
bhudak git push
warning: push.default is unset: its implicit value is changing in
Git 2.8 from 'matching' to 'simple'. To squelch this message
and maintain the current behavior after the default changes, use:
git config --global push.default simple
To squelch this message and adopt the new behavior now, use:
git config --global push.default simple
See 'git help config' and search for 'push.default' for further information.
(the 'simple' mode was introduced in Git 1.7.11. Use the similar mode
'current' instead of 'simple' if you sometimes use older versions of Git)
Counting objects: 5, done.
Delta compression using up to 12 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 261 bytes, done.
Total 3 (delta 1), reused 0 (delta 0)
To git@github.com:mwm/scratchpad.git
d19b6c8..9917cc8 master -> master
bhudak
```

Clone a project from GitHub, modify the file, and push it back

Help Menu

Help menu for common items such as recommended libraries, relevant tutorials, code examples and product documentation.

Remote Version Control

- Cloned projects from remote version control repositories are analyzed as they are loaded and automatically imported into the FP Haskell Center project management system.
- Changes can be pulled from a remote repository branch including manual module conflicts.
- Remote repository users can push local file and module commits into a selected branch.
- Ability to create / rename / switch between local repository branches.
- Ability to clone, view and modify the project code of publicly shared and published projects providing another means to contribute to the community.
- Published projects can be used as code templates for guiding others through common development tasks.



FP Haskell Application Server™

Integrated Application Deployment

The FP Application Server is a virtual server for deploying Haskell applications developed with FP Haskell Center™. It contains everything required for deploying a Haskell-based application into the cloud including a secure application container, required Haskell libraries, operating system, application manager for ensuring the application and support processes are running, control console, and application monitoring.

The FP Application Server works seamlessly with FP Haskell Center to provide a single console where developers can develop, build, and test their application. When ready, they can deploy to a fully functional application server running in the Amazon cloud. Each FP Haskell Center license includes a free, shared FP Application Server instance for testing. Each instance shares an 8-core system and provides 256 MB RAM and 10 GB storage. Dedicated FP Application Servers come in a variety of sizes for supporting small to large scale deployments and includes standard support.

Develop-Build-Test...Deploy

Manage Deployment Configurations

Yesod App 1

Title: Yesod App 1

Empty web app

Title: Empty web app

Hostname: s53fddaf-c508-4c39-8e95-b9216e93acca.app-staging.fc

SSL:

Module: Mainsrc/Main.hs

Arguments: E.g. -foo -bar 123

Environment: Add Variable

Create Server and Launch

Launch application server

License and application server type: Small (Standard support): EC2 - m1.small

Next

Back to application server list

Note: Dedicated FP Application Servers require the purchase of application server licenses.

Launch application server

Name: Yesod Server 1

Disk size (GB): 8

Launch application server

Back to application server list

Deploy Application

Deploy application

Application server: Yesod Server 1 - Ready

Next

Back to application server list

Note: Our console deployment system is currently very rudimentary, but we intend to add more functionality soon. For now, console apps are treated as background jobs which are automatically restarted when the process terminates, and have no access to standard input. If you have specific functionality requests, please direct them to support@fpcomplete.com.

Deploy application

App name: Yesod App 1

Binary URL: https://s3.amazonaws.com/temp.fpcomplete.com/project-staging/dkiphw

Open TCP ports (space separated): 80 443

Open UDP ports (space separated):

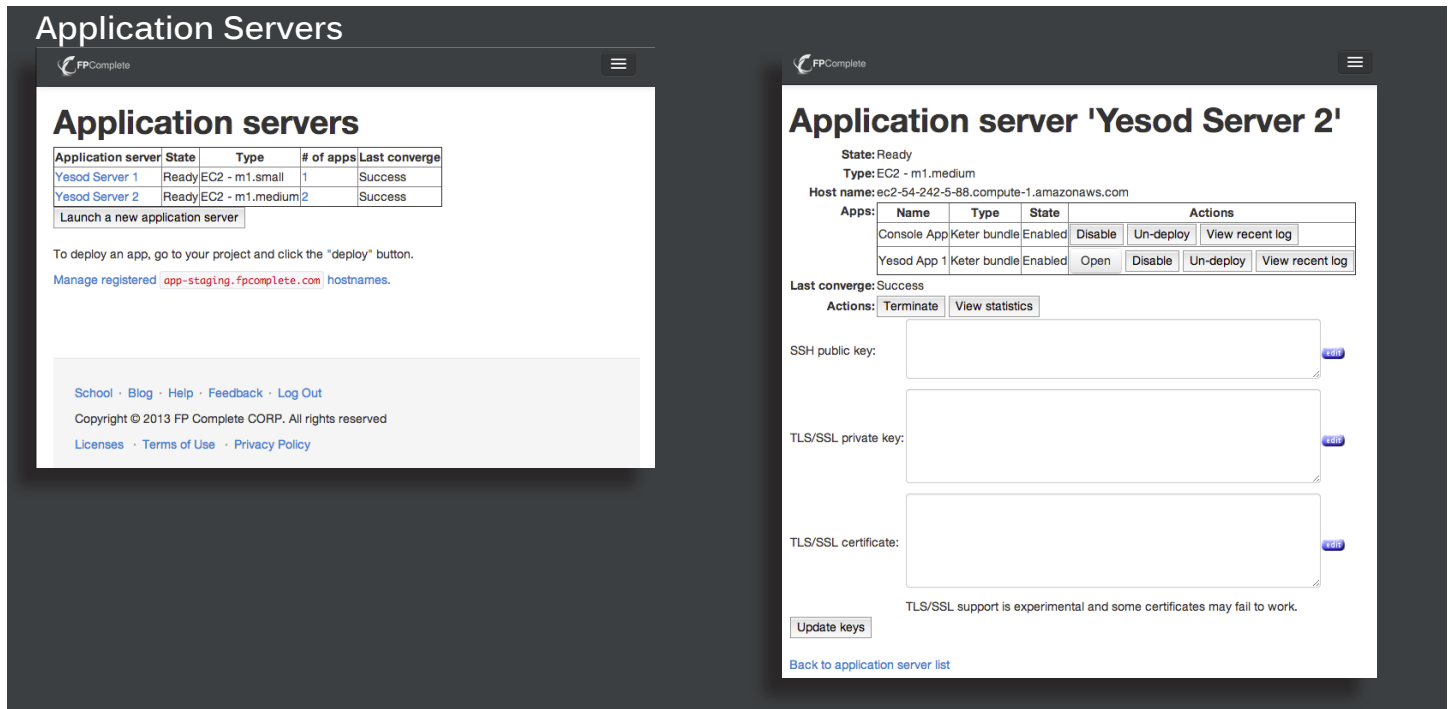
Keter bundle config:

```
plugins: {}
stanzas:
- ssl: false
  env: {}
  hosts:
  - 17df1471-f05d-4898-945d-c7cb36a8b279.app-staging.fpcomplete.com
  type: webapp
  args: []
  exec: ../Main
```

Deploy app

Back to application server list

FP Haskell Application Server™



FP Haskell Application Server Features

- Integrated dashboard for managing and running background jobs (daemons), web applications and services.
- Choice of domain, built-in fpcenter.io or user-provided.
- Fully-powered deployment manager that allows viewing and management of deployed apps and servers as well as viewing of logs and statistics.
- Sharing FP Application Servers across team accounts.
- Some future features planned include FP Application Servers behind the firewall and provide real-time performance profiling and postmortem debugging.

Dedicated FP Haskell Application Servers

Dedicated FP Haskell Application Servers are available for a monthly fee. They are installed on EC2 instances at Amazon Web Services. You can configure your servers securely through remote execution tools such as SSH.

EC2 instance	Small	Medium	Large	X Large
ECU	1	2	4	8
Memory (GB)	1.7	3.75	7.5	15
Storage (GB)	1x160	1x410	2x420	4x420

