Elixir Tooling

Dependencies & Packages

@emjii

Mix

Mix

- Generate new projects
- Compile
- Run tests
- Handle dependencies
- Whatever else you can think of

mix.exs

```
defmodule MyProject.Mixfile do
  use Mix.Project
  def project do
    [app: :my_project,
     version: "0.1.0",
     elixir: "~> 1.0"]
  end
end
```

Dependencies

```
defmodule MyProject.Mixfile do
  use Mix.Project
  def project do
    [app: :my_project,
     version: "0.1.0",
     deps: deps]
  end
  defp deps do
    [{:poolboy, github: "devinus/poolboy"},
     {:ecto, "~> 1.2"}]
  end
end
```

Dependencies

- $ mix deps.get / deps.update / ...
- Repeatable builds
- Rebar & Makefile dependencies

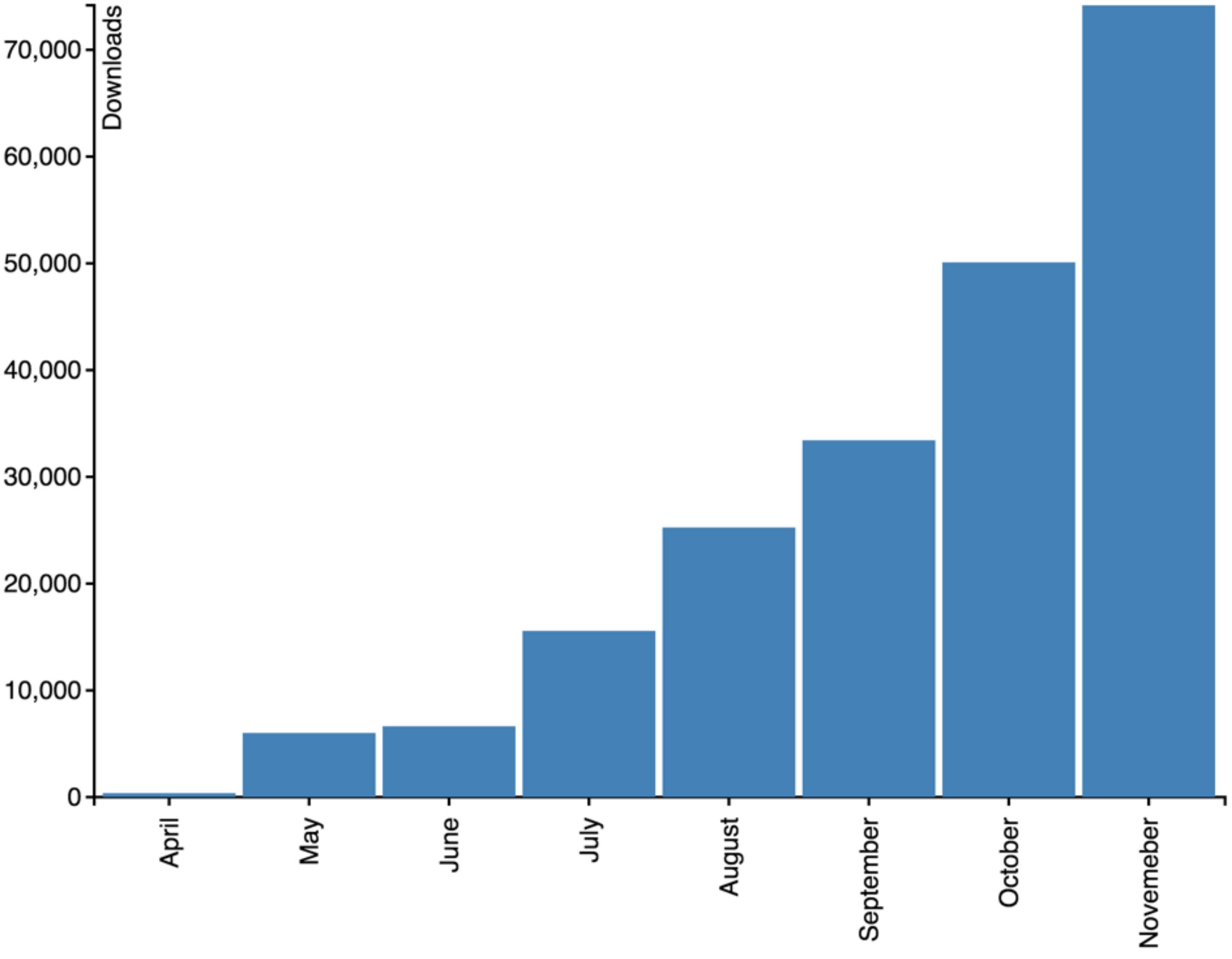
mix.lock

```
%{"cowboy": {:hex, :cowboy, "1.0.0"},
  "cowlib": {:hex, :cowlib, "1.0.0"},
  "plug": {:hex, :plug, "0.8.4"},
  "ranch": {:hex, :ranch, "1.0.0"},
"websocket_client": {
  :git,
  "git://github.com/jeremyong/websocket_client.git",
  "2b8d9805306d36f22330f432ae6472f1f2625c30",
  []}}
```

Dependencies

- $ mix deps.get / deps.update / ...
- Repeatable builds
- Rebar & Makefile dependencies

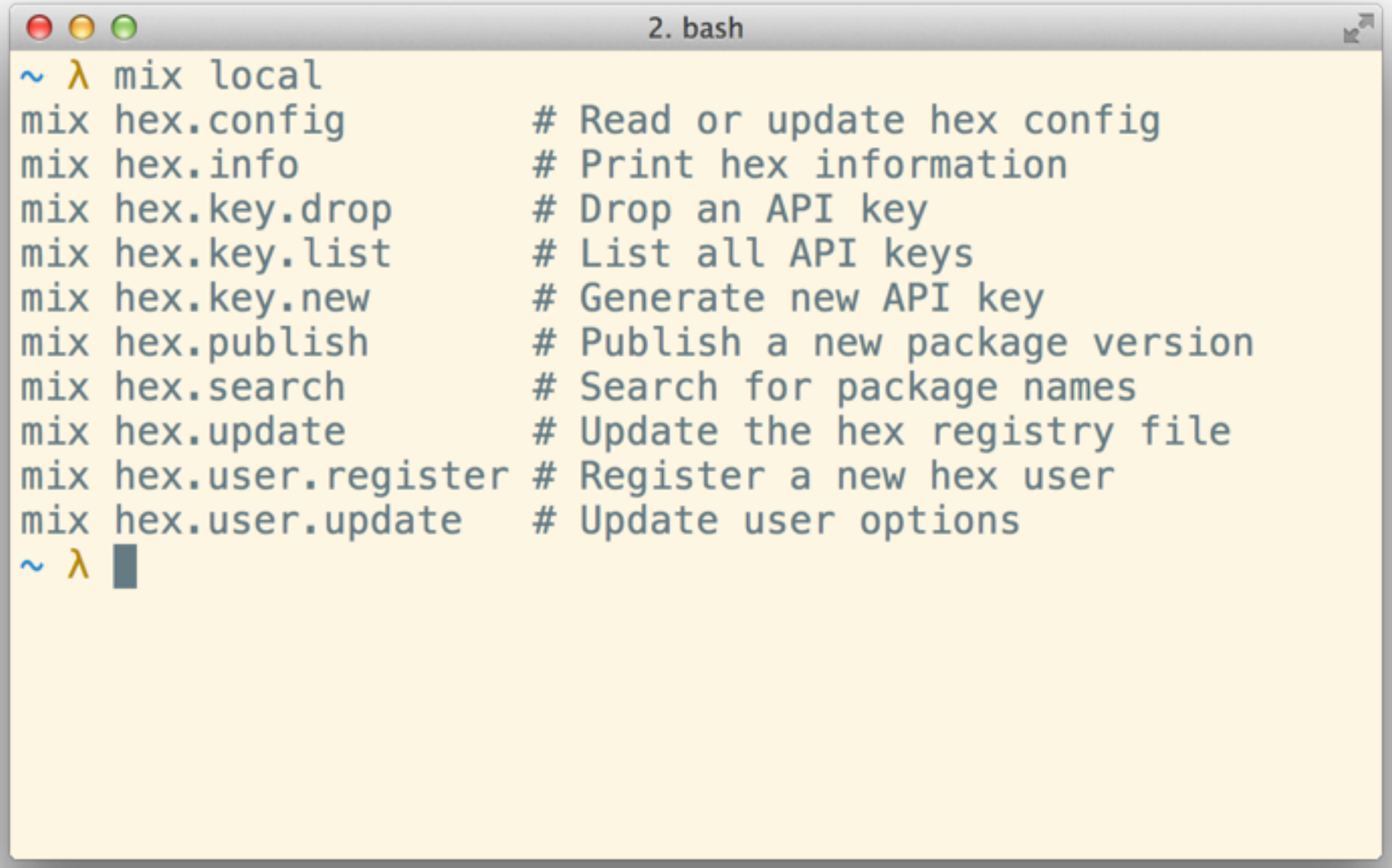
Hex


Downloads
70,000
60,000
50,000
40,000
30,000
20,000
10,000
0
April
May
June
July
August
September
October
Novemeber

Mix integration

- Mix tasks
- Archives
- Updating
- Remote converger

Hex tasks


2. bash
~ λ mix local
mix hex.config # Read or update hex config
mix hex.info # Print hex information
mix hex.key.drop # Drop an API key
mix hex.key.list # List all API keys
mix hex.key.new # Generate new API key
mix hex.publish # Publish a new package version
mix hex.search # Search for package names
mix hex.update # Update the hex registry file
mix hex.user.register # Register a new hex user
mix hex.user.update # Update user options
~ λ

Mix tasks

```
defmodule Mix.Tasks.MyTask do
  use Mix.Task

  def run(args) do
    IO.puts "Hello world!"
  end
end

$ mix my_task
Hello world!
```

Extending Mix

- ecto (github.com/elixir-lang/ecto)
  - $ mix ecto.gen.migration
  - $ mix ecto.migrate
- exrm (github.com/bitwalker/exrm)
  - $ mix release

Mix integration

- Mix tasks
- Archives
- Updating
- Remote converger

Code archives

- ZIP-file of .beam and .app files
- Supported by Erlang's code loader
- Auto-loaded from ~/.mix/archives

Code archive

hex-1.0.0.ez

- hex.app
- Elixir.Hex.beam
- Elixir.Mix.Tasks.Hex.Info.beam

Code archives

- ZIP-file of .beam and .app files
- Supported by Erlang's code loader
- Auto-loaded from ~/.mix/archives

Mix integration

- Mix tasks
- Archives
- Updating
- Remote converger

Mix integration

- Mix tasks
- Archives
- Updating
- Remote converger

mix deps.get

1. Run converger
2. Run remote converger
3. Fetch packages

Mix converger

- Traverse tree breadth-first
- Converge if possible
- Error on diverge
- Sort dependencies

mix deps.get

1. Run converger
2. Run remote converger
3. Fetch packages

Remote converger

- Hooks into converger
- Update registry
- Run dependency resolver

Registry

- Single ETS file

```
{"ecto",     [["0.1.0", "0.1.1", "0.1.2", …]]}
{"postgrex", [["0.1.0", "0.2.0", "0.2.1", …]]}

{{"ecto", "0.1.0"}, [[["postgrex", "~> 0.1.0"],
                      ["poolboy", "~> 1.1.0”],
                      …]]}
```

Dependency resolution

1. Add deps from mix.exs to pending requests
2. Take next pending, find latest matching release
    2a. Compare against activated package
    2b. If no matching, backtrack
3. Activate the package
4. Add children to pending
5. Save state for backtracking
6. Goto 2

mix deps.get

1. Run converger
2. Run remote converger
3. Fetch packages

Fetching packages

- Conditional HTTP requests
- Parallel fetching
- Cached locally

Package tarballs

ecto-1.0.0.tar

VERSION

metadata.exs

CHECKSUM

contents.tar.gz

Using Hex with Erlang

- Standalone Mix
- Use Mix for your Erlang projects
- Integration with Erlang tooling?

Hex.pm

Hex is a package manager for the Erlang ecosystem.

Using with Elixir

Simply specify your Mix dependencies as two-item tuples like `{:ecto, "~> 0.1.0"}` and Elixir will ask if you want to install Hex if you haven't already. After installed, you can run `$ mix local` to see all available Hex tasks and `$ mix help TASK` for more information about a specific task.

Using with Erlang

Download a standalone Mix, put it in your `PATH` and give it executable permissions. Now you can install Hex with `$ mix local.hex` and run all of its tasks.

Statistics

329	packages
1 198	package versions
3 950	downloads yesterday
24 613	downloads last seven days
215 545	downloads all time

Most downloaded

15 187	plug
14 947	cowboy
13 822	cowlib
11 667	ranch
9 785	poison
9 732	linguist
8 847	conform
8 754	ex_conf
7 843	decimal
6 870	jsx

Hosted documentation

- Uses ExUnit to generate
- Just run `$ mix hex.docs`
- Hosted on hexdocs.pm

▸ Plug
▸ Plug.Adapters.Cowboy
▸ Plug.Adapters.Translator
▸ Plug.Builder
▸ Plug.Conn
▸ Plug.Conn.Adapter
▸ Plug.Conn.Cookies
▸ Plug.Conn.Query
▸ Plug.Conn.Status
Plug.Conn.Unfetched
▸ Plug.Conn.Utils
▸ Plug.Crypto
▸ Plug.Crypto.KeyGenerator
▸ Plug.Crypto.MessageEncryptor
▸ Plug.Crypto.MessageVerifier
▸ Plug.Debugger
▸ Plug.Head
▸ Plug.Logger
▸ Plug.MIME
▸ Plug.MethodOverride
▸ Plug.Parsers
▸ Plug.Parsers.JSON
▸ Plug.Parsers.MULTIPART
▸ Plug.Parsers.URLENCODED

Plug

Plug is:

1. A specification for composable modules in between web applications
2. Connection adapters for different web servers in the Erlang VM

Documentation for Plug is available online.

Hello world

```
defmodule MyPlug do
  import Plug.Conn

  def init(options) do
    # initialize options

    options
  end

  def call(conn, _opts) do
    conn
    |> put_resp_content_type("text/plain")
    |> send_resp(200, "Hello world")
  end
end

Plug.Adapters.Cowboy.http MyPlug, []
IO.puts "Running MyPlug with Cowboy on http://localhost:4000"
```

The snippet above shows a very simple example on how to use Plug. Save that snippet to a file and run it inside the plug application with:

```
mix run --no-halt path/to/file.exs
```

Access "http://localhost:4000" and we are done!

Installation

You can use plug in your projects in two steps:

escripts

```
{:my_dep, git: ”git://github.com/ericmj/my_dep.git”}

{:my_dep, ”0.1.0”}
```

```
$ mix escript.install git
  git://github.com/ericmj/my_escript.git

$ mix escript.install my_escript 0.1.0
```

?

@emjii