ardPower Documentation

Release v1.2.0

Anirban Roy Das

Sep 27, 2017

Contents

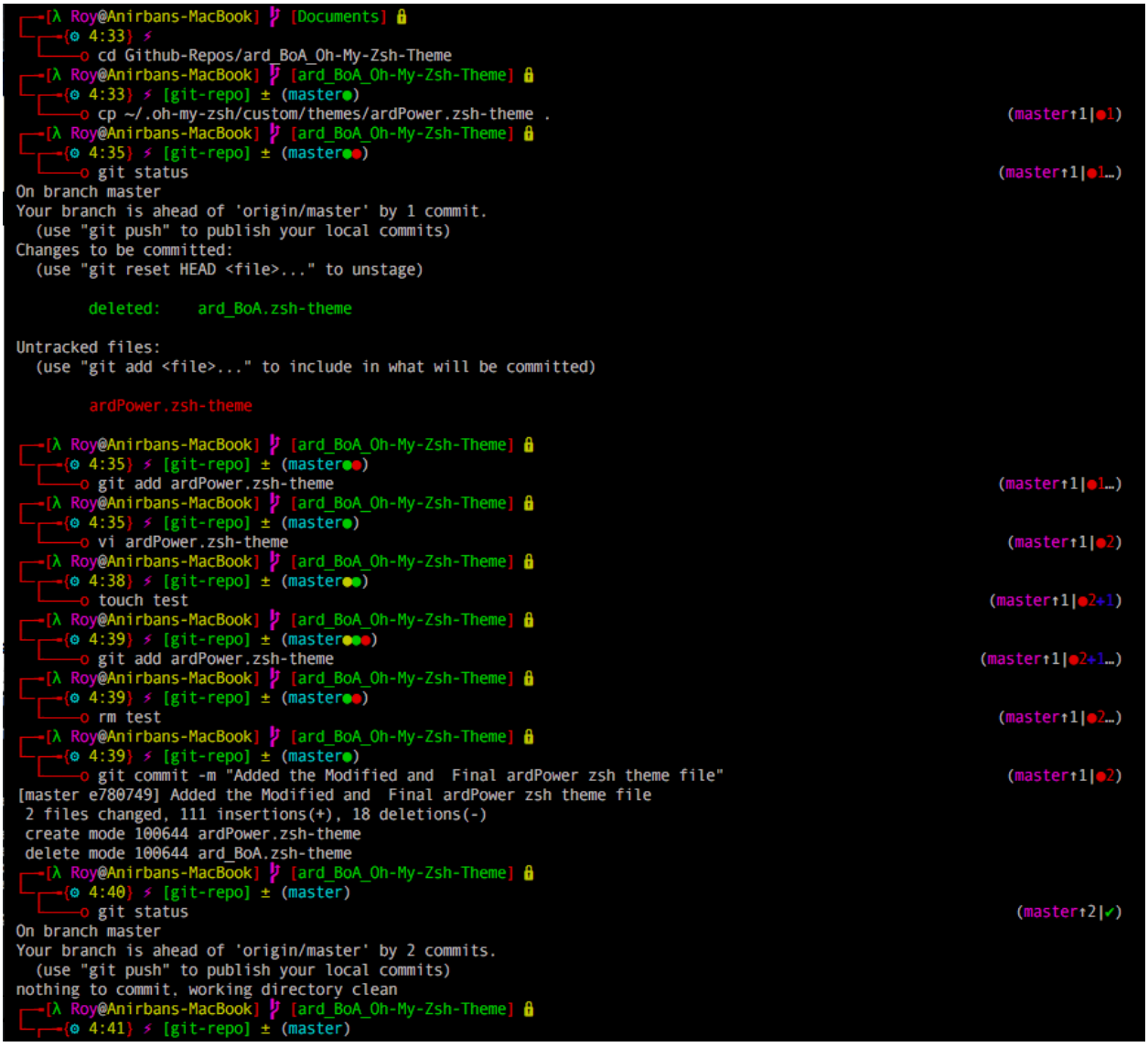
CHAPTER 1

Introduction

Its a custom `Oh-My-Zsh` Theme inspired by many custom themes suited for a perfect ZSH Environment under Byobu with Tmux Backend.

CHAPTER 2

Screenshot



CHAPTER 3

Documentaion

You can also find PDF version of the documentation here.

Overview

We will start with understanding the individual components of an entire CLI. Generally we don't put much attention to what we do. We just fire up a terminal or some say sheel and start typing our commands and get the result. That's all. But there are a lot of things that goes behind all this.

Terminal

The very first component is the `Terminal`.

So what is a Terminal?

> A terminal emulator, terminal application, term, or tty for short, is a program that emulates a video terminal within some other display architecture. Though typically synonymous with a shell or text terminal, the term terminal covers all remote terminals, including graphical interfaces.
>
> A terminal emulator inside a graphical user interface is often called a terminal window.A terminal window allows the user access to a text terminal and all its applications such as command-line interfaces (CLI) and text user interface (TUI) applications. These may be running either on the same machine or on a different one via telnet, ssh, or dial-up. On Unix-like operating systems, it is common to have one or more terminal windows connected to the local machine.

Examples:

- *Terminal App* in `Mac OS X`
- *iTerm2* in `Mac OS X`
- *gnome Terminal* in `Ubuntu`

Shell

A Unix shell is a command-line interpreter or shell that provides a traditional Unix-like command line user interface. Users direct the operation of the computer by entering commands as text for a command line interpreter to execute, or by creating text scripts of one or more such commands. Users typically interact with a Unix shell using a terminal emulator, however, direct operation via serial hardware connections, or networking session, are common for server systems. All Unix shells provide filename wildcarding, piping, here documents, command substitution, variables and control structures for condition-testing and iteration.

Examples

- **Bourne Shell**
  - Bourne Again Shell **(bash)**
  - Korn Shell **(ksh)**
  - Z Shell **(zsh)**
- **C Shell**

Zsh

The Z shell (zsh) is a Unix shell that can be used as an interactive login shell and as a powerful command interpreter for shell scripting. Zsh can be thought of as an extended Bourne shell with a large number of improvements, including some features of bash, ksh, and tcsh.

Oh-My-Zsh

Oh-My-Zsh is an open source, community-driven framework for managing your ZSH configuration. It comes bundled with a ton of helpful functions, helpers, plugins, themes, and a few things that make you shout... `Oh My ZSH!`

Terminal Multiplexer

A terminal multiplexer is a software application that can be used to multiplex several virtual consoles, allowing a user to access multiple separate login sessions inside a single terminal window, or detach and reattach sessions from a terminal. It is useful for dealing with multiple programs from a command line interface, and for separating programs from the session of the Unix shell that started the program, particularly so a remote process continues running even when the user is disconnected.

Screen & Tmux

GNU Screen

> GNU Screen is a terminal multiplexer, a software application that can be used to multiplex several virtual consoles, allowing a user to access multiple separate login sessions inside a single terminal window, or detach and reattach sessions from a terminal. It is useful for dealing with multiple programs from a command line interface, and for separating programs from the session of the Unix shell that started the program, particularly so a remote process continues running even when the user is disconnected.

Tmux

> tmux is a software application that can be used to multiplex several virtual consoles, allowing a user to access multiple separate terminal sessions inside a single terminal window or remote terminal session. It is useful for dealing with multiple programs from a command-line interface, and for separating programs

from the Unix shell that started the program.[2] It provides much of the same functionality as GNU Screen, but is distributed under a BSD license.

Byobu

Byobu is an enhancement for the terminal multiplexers GNU Screen or tmux that can be used to provide on screen notification or status as well as tabbed multi window management. It is aimed at providing a better user experience for terminal sessions when connecting to remote servers.

Powerline

Powerline is a statusline plugin for vim, and provides statuslines and prompts for several other applications, including zsh, bash, tmux, IPython, Awesome and Qtile.

Docs: https://powerline.readthedocs.org/en/latest/

Conclusion

So **ardPower** is basically a `Oh-My-Zsh` theme also using the `powerline` patched fonts for vibrant visual graphics and symbols which is used in a `zsh` shell under `Byobu` with `tmux` backend as defualt, can be changed to `screen` as per your wish.

Installation

Prerequisite

- `zsh shell`
- `oh-my-zsh`
- `byobu` (**recommended**) or `tmux` or `Gnu Screen`
- `powerline`, if you want the theme to render perfectly as its shown in the screenshot.

Step 1 - zsh shell

Confirm the Initial Zsh Version

```
$ zsh --version
zsh 5.0.5 (x86_64-apple-darwin14.0)
```

Confirm the zsh location:

```
$ which zsh
/bin/zsh
```

Confirm the present Shell that's set for you or User:

```
$ dscl . -read /Users/$USER UserShell
UserShell: /bin/bash
```

The . is short for localhost, and the $USER variable expands to your username.

Upgrade zsh with brew:

```
$ brew install zsh zsh-completions
```

Use the brew zsh:

```
$ sudo dscl . -create /Users/$USER UserShell /usr/local/bin/zsh
Password: *********
```

After that, restart your Terminal to have it take effect. You can also use System Preferences. Open Users & Groups, ctrl-click your username, then select "Advanced Options". You can select your shell in there.

Now if you run which again, you'll see the system is recognizing the one you installed:

```
$ which zsh
/usr/local/bin/zsh
```

Confirm You're Running Brew zsh:

```
$ dscl . -read /Users/$USER UserShell
UserShell: /usr/local/bin/zsh
```

That's the most precise way to confirm. Next, try echoing an environment variable (case matters):

```
$ echo $SHELL
/usr/local/bin/zsh
```

Handling Upgrades

There are a couple of considerations to keep in mind any time you upgrade OS X.First, your shell might get reset, so check it to be sure.

Secondly, OS X El Capitan (v 10.11) has a new security system called "System Integrity Protection", which is set up to be stricter with the security of /usr/local, among other things. Since this is where brew keeps its files, you'll likely need to reset security on it by running the following command:

```
sudo chown -R $(whoami):admin /usr/local
```

Step 2 - Superchargin Zsh - Oh-My-Zsh

Install oh-my-zsh

Install oh-my-zsh on top of zsh to getting additional functionality, we can use curl:

```
$ sh -c "$(curl -fsSL https://raw.github.com/robbyrussell/oh-my-zsh/master/tools/
↪install.sh)"
```

or wget:

```
$ sh -c "$(wget https://raw.github.com/robbyrussell/oh-my-zsh/master/tools/install.sh␣
↪-O -)"
```

Configure oh-my-zsh

You can configure oh-my-zsh settings just by opening ~/.zshrc in a text editor. It should look like this.

```
# Path to your oh-my-zsh installation.
export ZSH=/Users/username/.oh-my-zsh

# Set name of the theme to load.
# Look in ~/.oh-my-zsh/themes/
# Optionally, if you set this to "random", it'll load a random theme each
# time that oh-my-zsh is loaded.
ZSH_THEME="robbyrussell"

# Uncomment the following line to use case-sensitive completion.
# CASE_SENSITIVE="true"

# Uncomment the following line to use hyphen-insensitive completion. Case
# sensitive completion must be off. _ and - will be interchangeable.
# HYPHEN_INSENSITIVE="true"

# Uncomment the following line to disable bi-weekly auto-update checks.
# DISABLE_AUTO_UPDATE="true"

# Uncomment the following line to change how often to auto-update (in days).
# export UPDATE_ZSH_DAYS=13

# Uncomment the following line to disable colors in ls.
# DISABLE_LS_COLORS="true"

# Uncomment the following line to disable auto-setting terminal title.
# DISABLE_AUTO_TITLE="true"

# Uncomment the following line to enable command auto-correction.
# ENABLE_CORRECTION="true"

# Uncomment the following line to display red dots whilst waiting for completion.
# COMPLETION_WAITING_DOTS="true"

# Uncomment the following line if you want to disable marking untracked files
# under VCS as dirty. This makes repository status check for large repositories
# much, much faster.
# DISABLE_UNTRACKED_FILES_DIRTY="true"

# Uncomment the following line if you want to change the command execution time
# stamp shown in the history command output.
# The optional three formats: "mm/dd/yyyy"|"dd.mm.yyyy"|"yyyy-mm-dd"
# HIST_STAMPS="mm/dd/yyyy"

# Would you like to use another custom folder than $ZSH/custom?
# ZSH_CUSTOM=/path/to/new-custom-folder

# Which plugins would you like to load? (plugins can be found in ~/.oh-my-zsh/plugins/
↪*)
# Custom plugins may be added to ~/.oh-my-zsh/custom/plugins/
# Example format: plugins=(rails git textmate ruby lighthouse)
# Add wisely, as too many plugins slow down shell startup.
plugins=(git)

# User configuration

export PATH="/usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin"
# export MANPATH="/usr/local/man:$MANPATH"
```

```
source $ZSH/oh-my-zsh.sh

# You may need to manually set your language environment
# export LANG=en_US.UTF-8

# Preferred editor for local and remote sessions
# if [[ -n $SSH_CONNECTION ]]; then
#   export EDITOR='vim'
# else
#   export EDITOR='mvim'
# fi

# Compilation flags
# export ARCHFLAGS="-arch x86_64"

# ssh
# export SSH_KEY_PATH="~/.ssh/dsa_id"

# Set personal aliases, overriding those provided by oh-my-zsh libs,
# plugins, and themes. Aliases can be placed here, though oh-my-zsh
# users are encouraged to define aliases within the ZSH_CUSTOM folder.
# For a full list of active aliases, run `alias`.

alias zshconfig="sublime ~/.zshrc"
alias ohmyzsh="sublime ~/.oh-my-zsh"
```

Oh-my-zsh offers a lot of themes and plugins you can use to customize your shell experience. You can find everything on these links:

```
# Which plugins would you like to load? (plugins can be found in ~/.oh-my-zsh/plugins/
↪*)
# Custom plugins may be added to ~/.oh-my-zsh/custom/plugins/
# Example format: plugins=(rails git extmate ruby lighthouse)
# Add wisely, as too many plugins slow down shell startup.

plugins=(git git-extras git-flow colored-man colorize github vagrant virtualenv␣
↪virtualenvwrapper pip python brew osx zsh-syntax-highlighting npm docker django␣
↪bower celery node sublime sudo supervisor web-search)
```

Step 3 - Install byobu

Install via brew:

```
$ brew install byobu
```

Configure byobu

Start the Shell and initiate byobu by:

```
$ byobu-enable
```

Disable byobu:

```
$ byobu-disable
```

Start tmux as byobu backend:

```
$ byobu-tmux
```

Start screen as byobu-backend:

```
$ byobu-screen
```

Select backend permanently:

```
$ byobu-select-backend
```

Change key bindings and evironment

Go to `~/.byobu/` and start changing the config files for tmux. You can also make changes in the `/usr/local/byobu/` directory for advanced control.

Step 4 - Installing powerline

Pip installation

Due to a naming conflict with an unrelated project powerline is available on PyPI under the powerline-status name:

```
pip install powerline-status
```

Patched fonts installation

If you don't install the patched fonts, then the theme will not render properly. This method is the fallback method and works for every terminal.

Download the font from powerline-fonts. If preferred font can't be found in the **powerline-fonts** repo, then patching the preferred font is needed instead.

After downloading this font refer to platform-specific instructions.

Guide to platform specific instructions

Please follow the official docs where the process is better explained, click here

Install

This section will tell you how to install ardPower into your system.

1. Download the **ardPower.zsh-theme** file from the repo or copy it from the fork of this repo or clone of this repo.
2. Paste the file into `~/.oh-my-zsh/custum/themes/` directory, if the directory does not exist, create it.
3. Edit the `~/.zshrc` file and update the line `ZSH_THEME="ardPower"` and source the file:

   ```
   $ source ~/.zshrc
   ```

4. Restart Terminal for better performance.

Usage

Use Case

You can use this theme to have better control of your development environment. Specially when you are doing a lot of git development.

This theme provides control over your git repos, by specifying whenver you are in a git repo. It searches for a .git folder in your workind directory and changes the shell view accordingly.

Moreover, it shows you in which branch of your git repo you are presently in. Also, it indicates if you have untracked files, tracked but modified files, tracked and staged files waiting to be commited via different color dots like a traffic signal. It also shows the number of additions in terms of files and edits.

It has a very vibrant look which used many Powerline-status symbols and glyphs. To activate them follow the Installation documentation.

If powerline is not installed, the theme may not render properly in your system.

How to Use

The theme is very vibrant and easy to understand.

1. The first line shows you the `user@host` info along with the current working directory you are in .
2. The second line shows the time the command was invoked along with whether you are in a git repo or not. It also shows you the present branch of the git repo you are in. It also shows the status of your branch, likely untracked files, unstaged files, staged files, removed files, etc.
3. The 3rd and last line gives you the space to type you command along with a right prompt if you are in a git repo with the branch and edit informations.

CHAPTER 4

Indices and tables

- genindex
- modindex
- search