cisco Live!

Let's go

#CiscoLiveAPJC

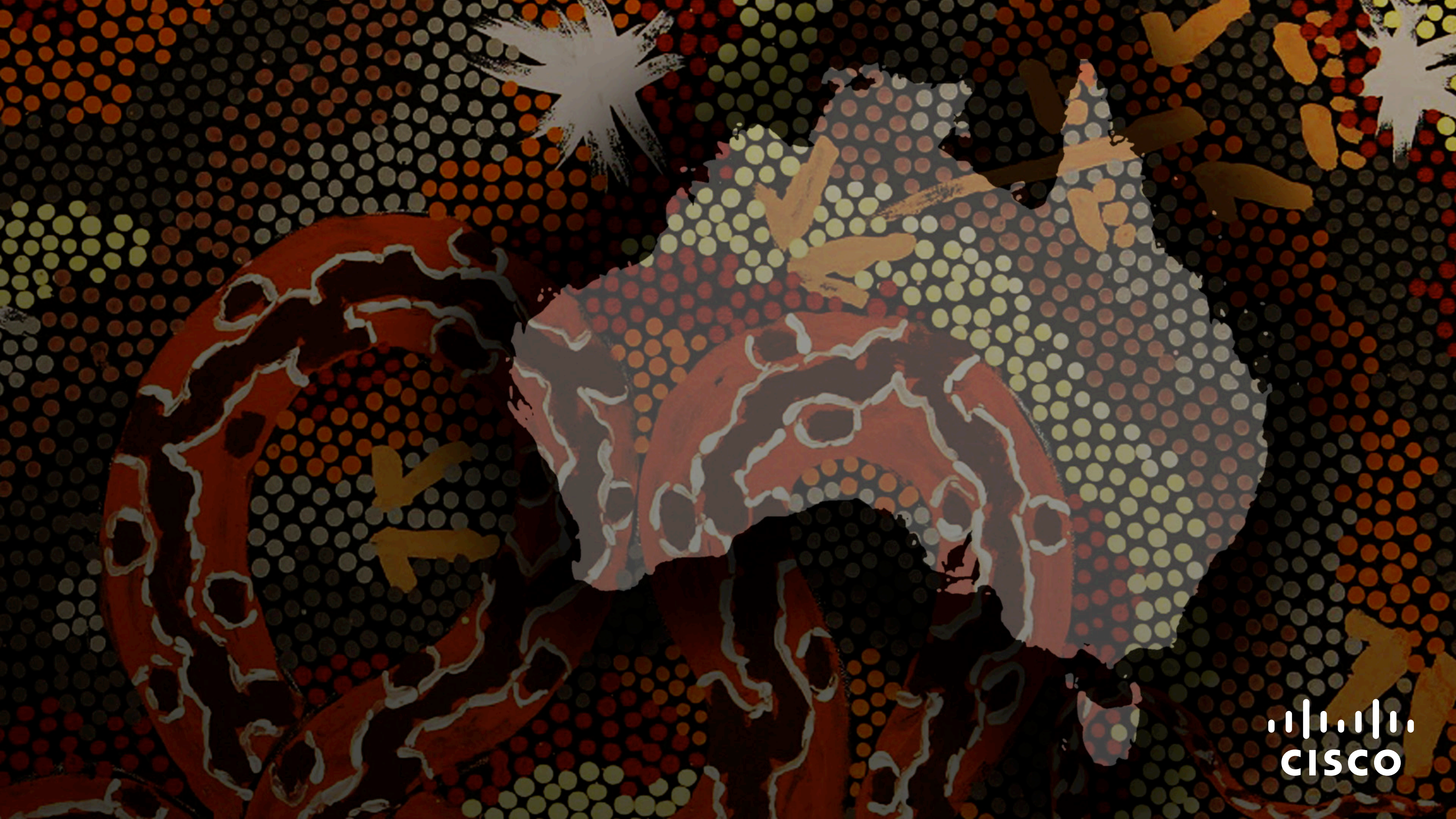
The bridge to possible

Vim: Its Easier Than You Think

How to Exit Without Restarting Your Computer

Quinn Snyder | Technical Advocate, Learning and Certifications
@qsnyder

CISCOU-1815



Agenda

- Introduction
- Why Vim at all?
- Default this!
- Navigation and search
- Modes of operation
- Cut, copy, paste
- Windows, buffers, and tabs
- Conclusion

Let’s Make This Interactive!

https://github.com/qsnyder/ciscou-1815

Contains two configuration files and PDF of slides for those who wish to follow along in real time as I perform the demo

If You’d Rather Watch First…

Cisco U. Tutorial Links

Introduction to Vim

Great for beginners, those just learning

Intermediate Vim

Meant to expand your Vim knowledge, improve workflow

Why Vim?

Its Literally Everywhere

Linux, macOS, and WSL…oh my!

- Vim (Vi iMproved), or at least Vi, is part of the base install for nearly every *nix install
    - Includes most every WSL installation, since they are full-featured OS
- Simple enough to learn for basic editing
- Nano/Pico often require package manager installation
- Remote extensions (VSC, JetBrains) require remote-side download or installation
- Your fingers get to stay on the keyboard!

Let’s Get Started!

Setting the Environment

Establishing some ground rules

By default, Vim leaves a bit to be desired in the editor window

- `:set bg=dark`
  - Useful for dark terminals
- `:set number relativenumber`
  - `number` sets the line numbers on the left side of editor
  - `relativenumber` displays line increments in relation to current line
  - Together, current line is absolute with relative numbers above and below
- `:set cursorcolumn`
  - Vertical alignment column onscreen
- `:set hlsearch`
  - Highlights all search results in the current buffer
- Store in `~/.vimrc` to ensure they are defaults

Basic Navigation

Moving within a file

- Arrow keys are acceptable
- `0` and `$` for front and back of line
- `b`/`w` (or `B`/`W`) to move forward or backward by w/Word,
- `j`, `k` to move down, up lines (classic `h-j-k-l` nav)
    - `–`, `+` can be used, respectively
    - `:<num>` skips right to line number
- Numbers to augment number of "skips"
- (`H`)igh, (`M`)edium, and (`L`)ow within screen
- `Ctrl-b`, `Ctrl-f` for screen movement
- `{` or `}` for paragraph skips
- (`:w`)rite
- (`:q`)uit

Modes of Operation

Why remember one set of keystrokes when you can remember 3?

- When opened, Vim operates in “Normal” mode
  - Nothing indicated onscreen
  - Navigation, text manipulation occurs here
- “Insert” mode allows for adding net-new text
  - (`i`)nsert, (`a`)ppend, (`c`)hange will activate insert mode
  - Denoted on screen
- “Visual” mode used for text selection for manipulation
  - `v` will place you into ”visual” mode
  - `V` will place you into “visual line” mode
  - Both modes indicated on bottom of screen
  - Select text in this mode for cut/copy actions

Finding and Replacing

- Searching for text string within file
  - `/<string>`
  - `n` moves from top to bottom in results
  - `N` moves from bottom to top in results
  - This is where `hlsearch` is handy
  - `/c` to make it case-insensitive

- Finding and replacing text within a file
  - `:s/<search_text>/<replacement>/` - first found instance
  - Appending `/g` makes it file global
  - Appending `/i` makes search case insensitive
  - Appending `/c` makes each change require confirmation
  - Build your own adventure by using what you need

Cut/copy/paste with a side of registers

Now we begin to get strange

- Because Vim has to be weird:
  - Cut doesn’t exist – just (`d`)elete
  - Copy is called (`y`)ank
  - (`p`)aste is normal, though
- Text that is deleted or yanked, is moved to a register
  - `:reg` to view, text placed in unnamed (`""`) register by default
  - `"<x>p` to paste from numbered register
  - Yanked text will always be stored in register `0`, deletes in `1-9` (rolling)
- Text copied from system (outside of Vim), will be stored in the `*` register

Tabs, buffers, and windows – oh my!

Inception intensifies…

- Buffers exist to store text
  - Can be empty, but each new file is opened in a buffer too (think `:w`)
  - If you open multiple files from command line, use `:edit`, or `:new` = new buffer
- Windows are the arrangement of buffers within a current "view"
  - `:split` - `:vsplit` to create, `Ctrl-w+w/W` to move, `Ctrl-w+x` to reverse layout
  - `:terminal` - `:vertical terminal` to invoke terminal in new window
- Tabs are used to collect windows – each tab can have a different window layout
  - `:tabnew` - `:tabnext` - `:tabprev`
  - `:tabnew terminal` - creates new terminal in tab
- `Ctrl-w+N` for terminal normal mode

Wrapping Up

Session Surveys

We would love to know your feedback on this session!

- Complete a minimum of four session and the overall event surveys to claim a Cisco Live merchandise.

Continue your education

- Visit the Cisco Showcase for related demos
- Book your one-on-one Meet the Expert meeting
- Attend the interactive education with DevNet, Capture the Flag, and Walk-in Labs
- Visit the On-Demand Library for more sessions at www.CiscoLive.com/on-demand

cisco Live!

Cisco U.

Tech learning: shaped to you, built for Us.

The bridge to possible

Thank you

#CiscoLiveAPJC

cisco Live!

Let’s go

#CiscoLiveAPJC