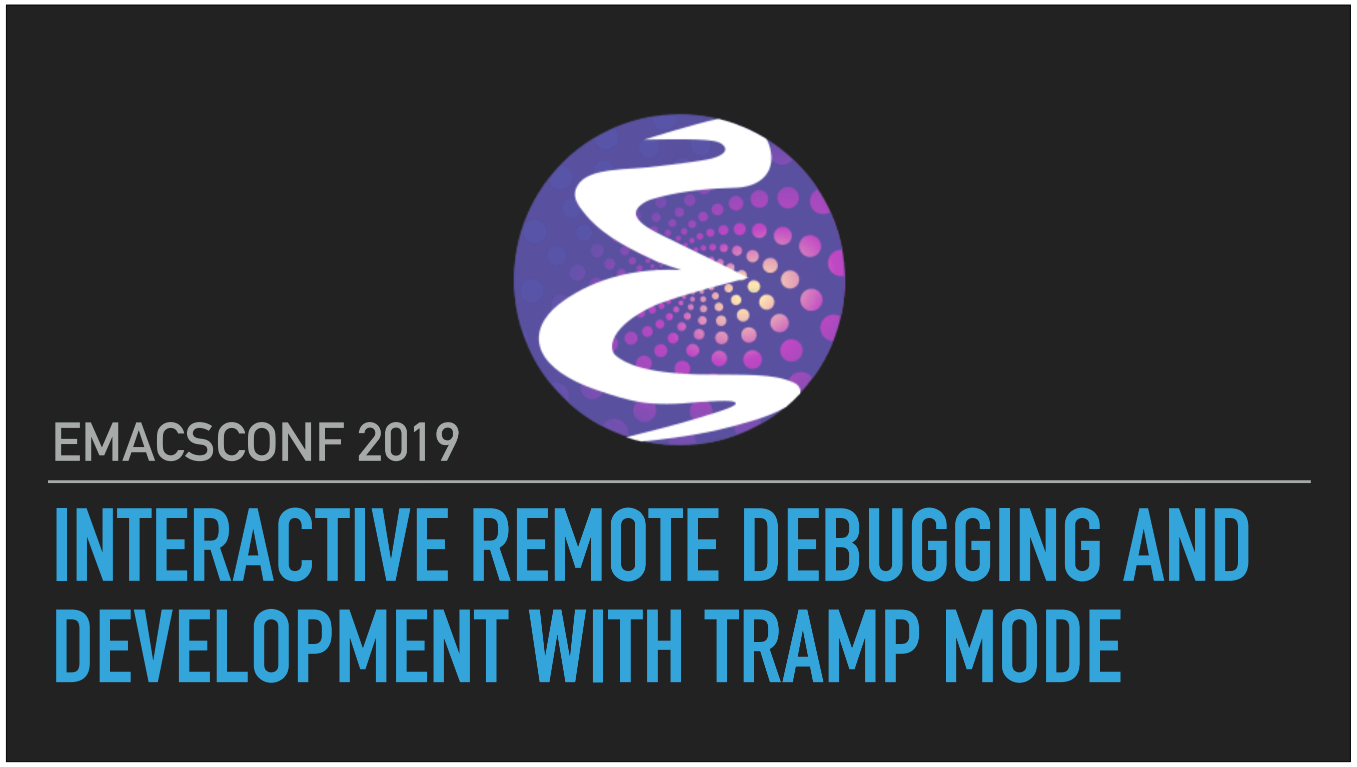
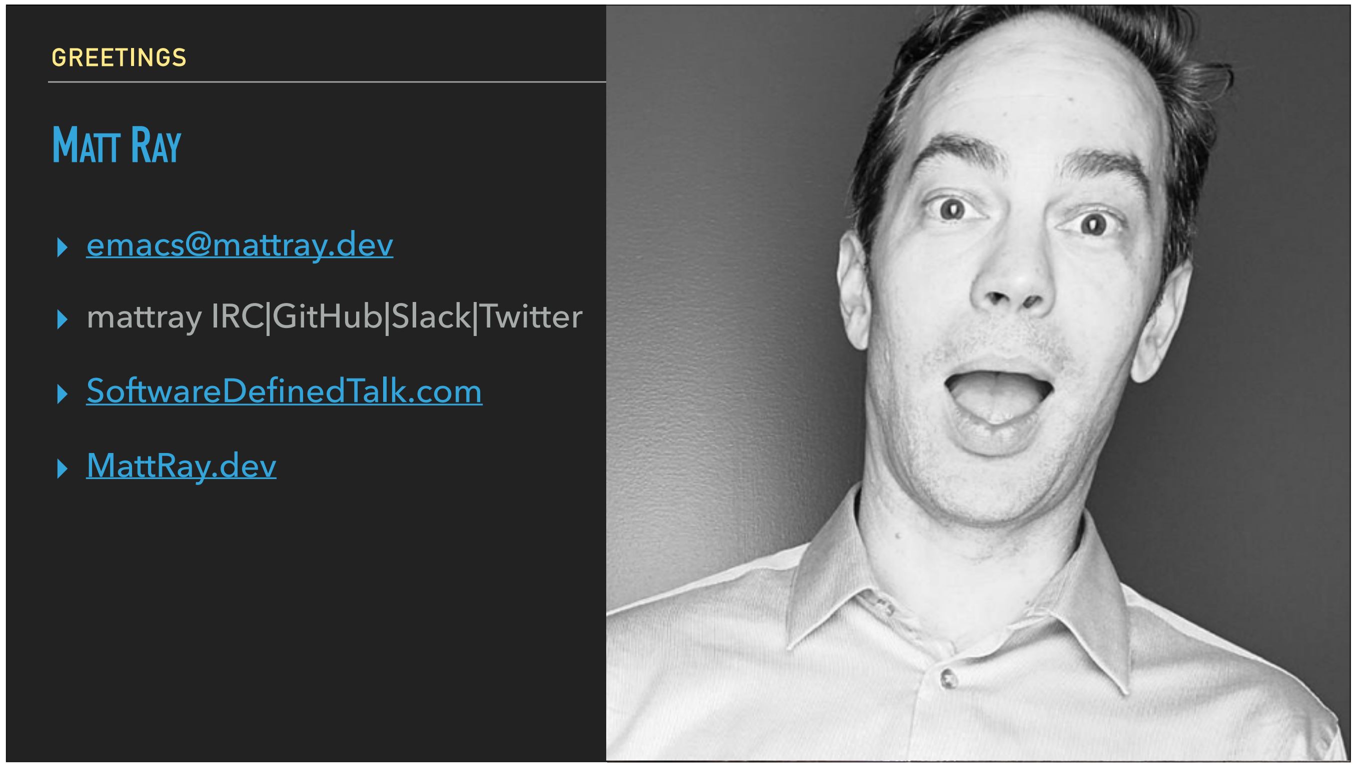


EMACSCONF 2019

INTERACTIVE REMOTE DEBUGGING AND DEVELOPMENT WITH TRAMP MODE

Greetings EmacsConf!
My name is Matt Ray, welcome to my session on Interactive Remote Debugging and Development with TRAMP Mode

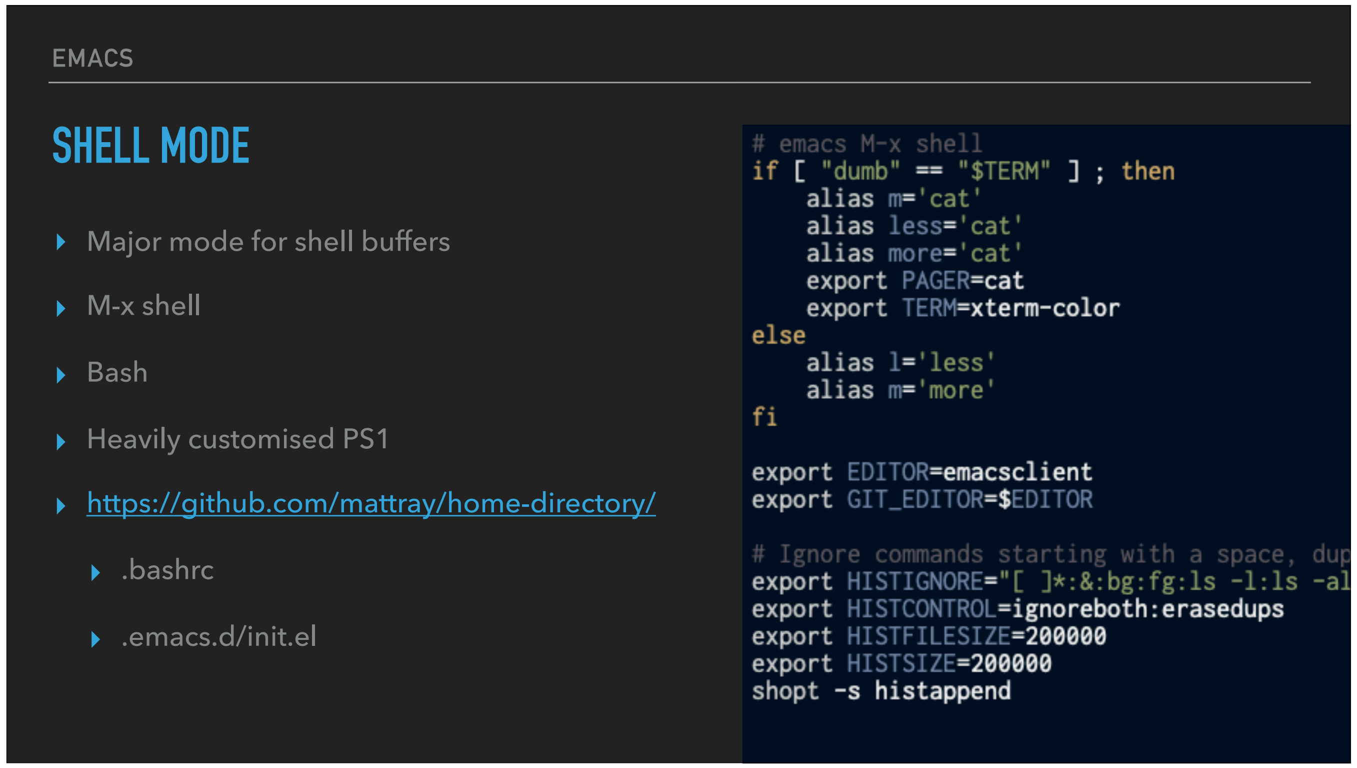


GREETINGS

Matt Ray

- emacs@mattray.dev
- mattray IRC|GitHub|Slack|Twitter
- SoftwareDefinedTalk.com
- MattRay.dev

First, I'd like to thank the organizers of EmacsConf for allowing me to present.
My name is Matt Ray and I've been using Emacs for about 20 years and figured it would be fun to share some of the ways I use Emacs in my day to day role.
I've been a developer and had various roles over the years.
I'm currently based in Sydney Australia, you can tell by my accent. I moved here about 3 years ago.
I'm 'mattray' on Twitter and Slack and GitHub and I co-host a rambling podcast called Software Defined Talk.
I blog from time to time about random tech stuff at MattRay.dev



EMACS

SHELL MODE

- Major mode for shell buffers
- M-x shell
- Bash
- Heavily customised PS1
- https://github.com/mattray/home-directory/
  - .bashrc
  - .emacs.d/init.el

```
# emacs M-x shell
if [ "dumb" == "$TERM" ] ; then
    alias m='cat'
    alias less='cat'
    alias more='cat'
    export PAGER=cat
    export TERM=xterm-color
else
    alias l='less'
    alias m='more'
fi

export EDITOR=emacsclient
export GIT_EDITOR=$EDITOR

# Ignore commands starting with a space, dup
export HISTIGNORE="[ ]*:&:bg:fg:ls -l:ls -al
export HISTCONTROL=ignoreboth:erasedups
export HISTFILESIZE=200000
export HISTSIZE=200000
shopt -s histappend
```

I've been using M-x shell mode for probably 20 years, it's always been one of the killer features for me within Emacs.
The ability to capture everything I do in an editable shell has proven invaluable time and time again and frequently been a great way to get get people interested in Emacs.
I'm still using Bash, newer shells like Zsh haven't really caught my eye because I've always gotten what I needed out of Shell mode.
<demo screen>
Within my .bashrc I set a couple of configuration settings, including setting more and less to cat so there's no weird pagination issues within shell mode.
My editor and git_editor shell variables are set for any commands that need to spawn an editor, and I use the emacs server to manage all of my emacs sessions within a single window.
Shell history settings are there to optimise what's tracked, and I frequently grep my .bash_history file for anything not in the current shell session.

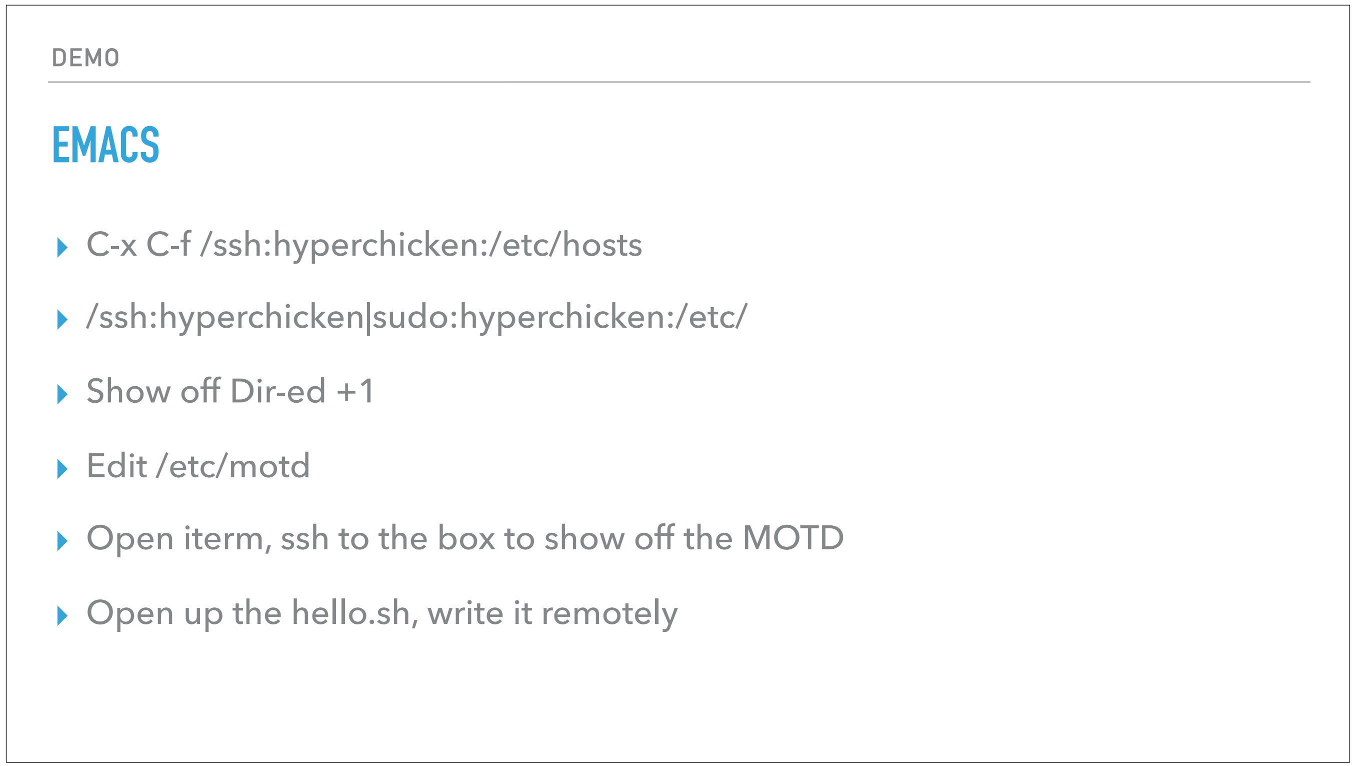
DEMO

M-X SHELL DEMO

- Start emacs
- M-x shell
- C-x 1
- M-+ 3x
- Cd emacsconf
- C-x C-f hello.sh
- Bash hello.sh

TRAMP MODE

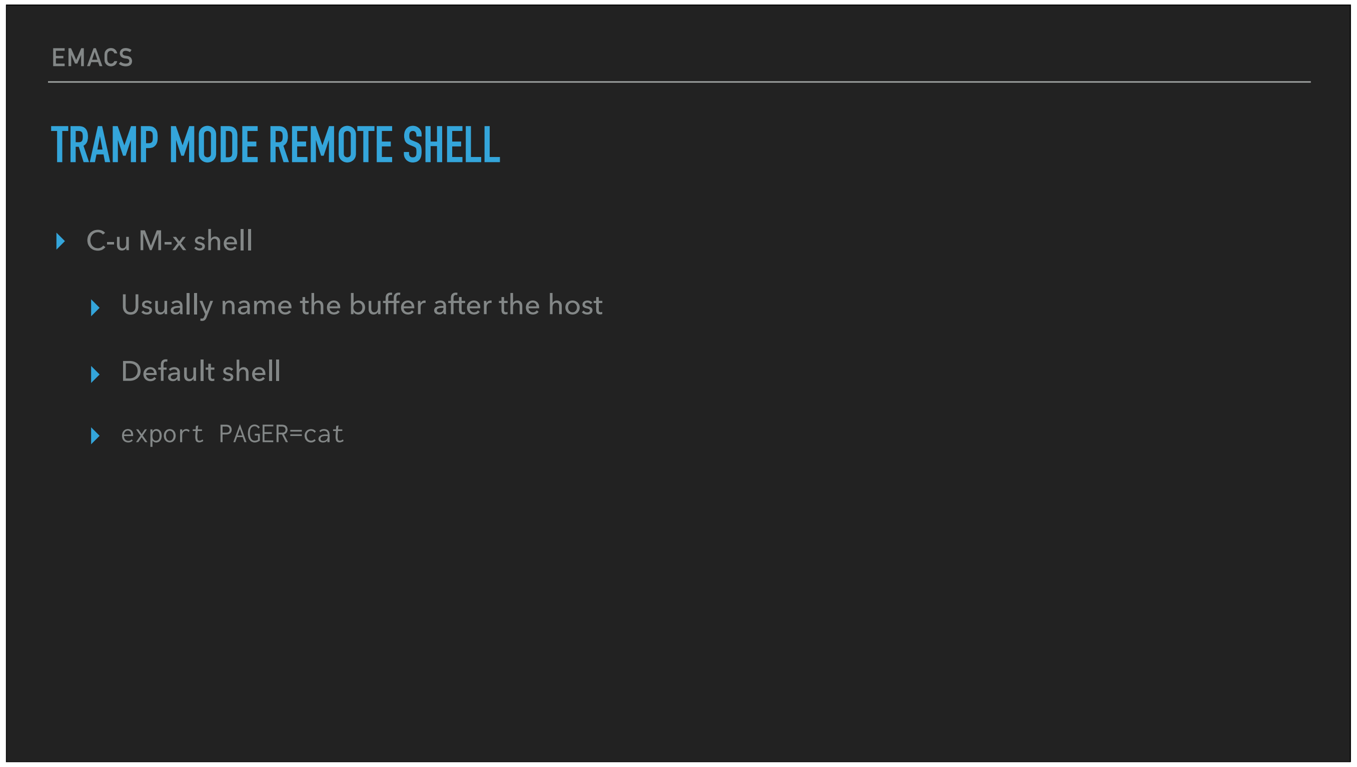
- Transparent Remote Access, Multiple Protocols
- Remote file editing over multiple protocols
- C-x C-f /[method/user@remotehost]/filename
  - /ssh:cubert:/etc/hosts
- Sudo support
  - /ssh:cubert|sudo@cubert:/etc/hosts
  - /ssh:cubert|sudo:omnibus@cubert:~/


DEMO

EMACS

- C-x C-f /ssh:hyperchicken:/etc/hosts
- /ssh:hyperchicken|sudo:hyperchicken:/etc/
- Show off Dir-ed +1
- Edit /etc/motd
- Open iterm, ssh to the box to show off the MOTD
- Open up the hello.sh, write it remotely

BeagleBone black ARM box
I can open files on the remote system
I can sudo in and edit them as root
I can use DirEd to wander through the filesystem, poking around
I can also open a file locally, then write it to the remote system
I'll do that with my 'hello.sh'

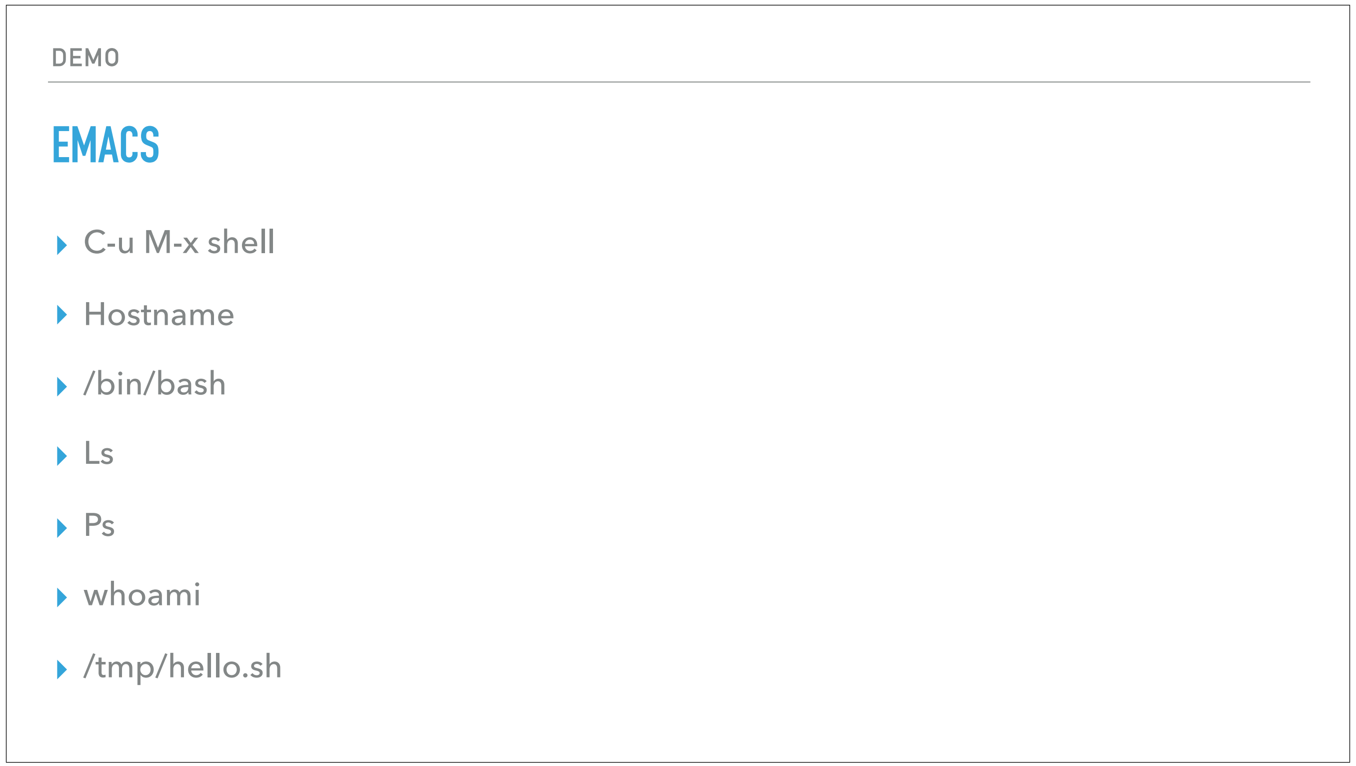


EMACS

TRAMP MODE REMOTE SHELL

- C-u M-x shell
  - Usually name the buffer after the host
  - Default shell
  - `export PAGER=cat`

Now editing files on a remote system is great and all, but I just showed off how I use Shell Mode.
I can use TRAMP to open a remote shell and run from there, just like on my local workstation.
I like to open a remote shell and name the buffer after the host, to make it easier to identify in my Buffer list



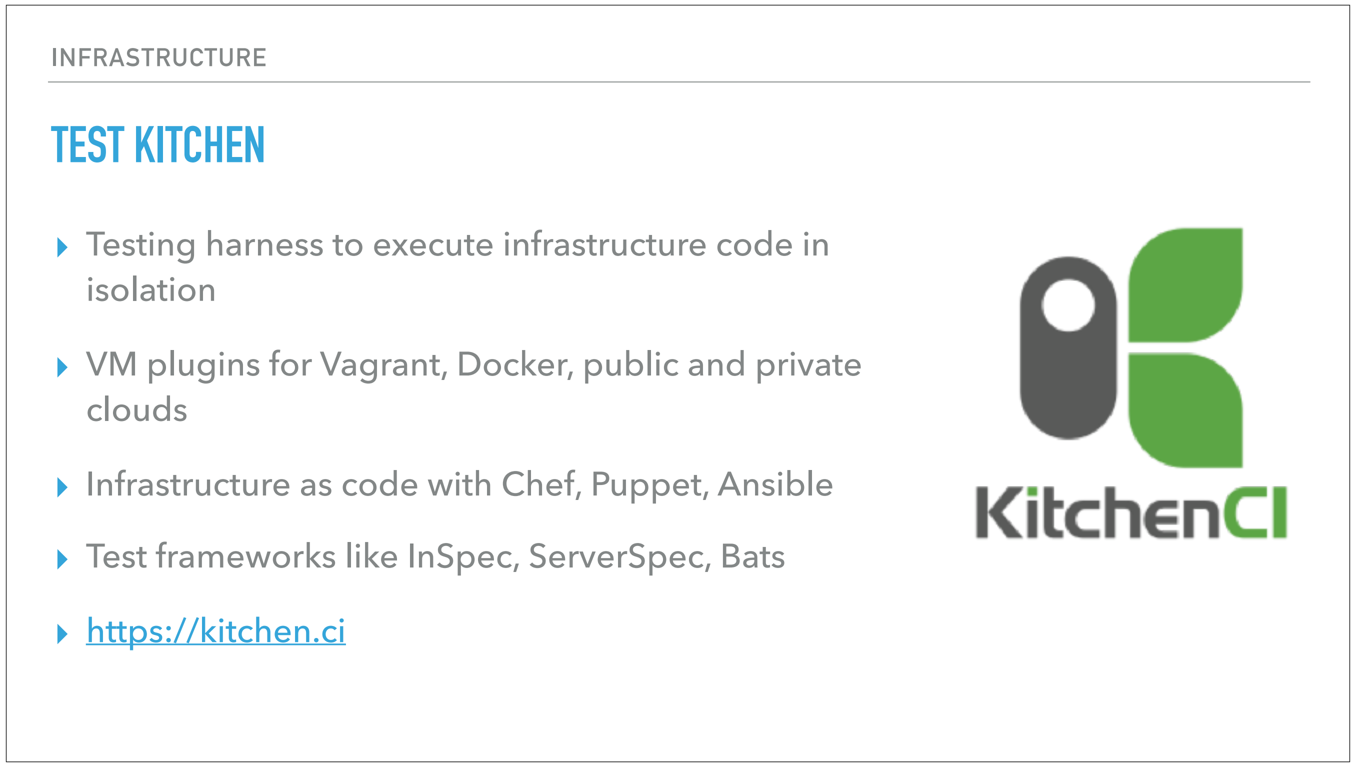
DEMO

EMACS

- C-u M-x shell
- Hostname
- /bin/bash
- Ls
- Ps
- whoami
- /tmp/hello.sh

Default shell
export PAGER=cat

I'm now running my shell over SSH on a remote server, editing files and copying content between buffers, pretty cool.



INFRASTRUCTURE

TEST KITCHEN

- Testing harness to execute infrastructure code in isolation
- VM plugins for Vagrant, Docker, public and private clouds
- Infrastructure as code with Chef, Puppet, Ansible
- Test frameworks like InSpec, ServerSpec, Bats
- https://kitchen.ci

KitchenCI

Apache v2 licensed Free Software

VAGRANT

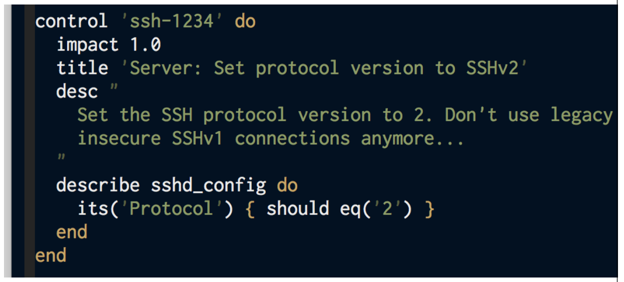
- Developer testing tool for desktop virtual machines
- Virtualbox, Docker and other providers
- Simple configuration with SSH access
- Standardized 'bento' images from Chef for basic machines
- https://vagrantup.com
- C-x C-f /ssh:vagrant@127.0.0.1#2222:/

EMACS

- Kitchen status
- Kitchen create
- Open kitchen.yml
- C-x C-f /ssh:vagrant@127.0.0.1#2222:/
- C-u M-x shell
- Whom

INSPEC

- Compliance as Code
- Ruby DSL
- 100s of Resources for auditing machines, databases, APIs, cloud platforms, etc.
- Local or remote scanning
- https://inspec.io



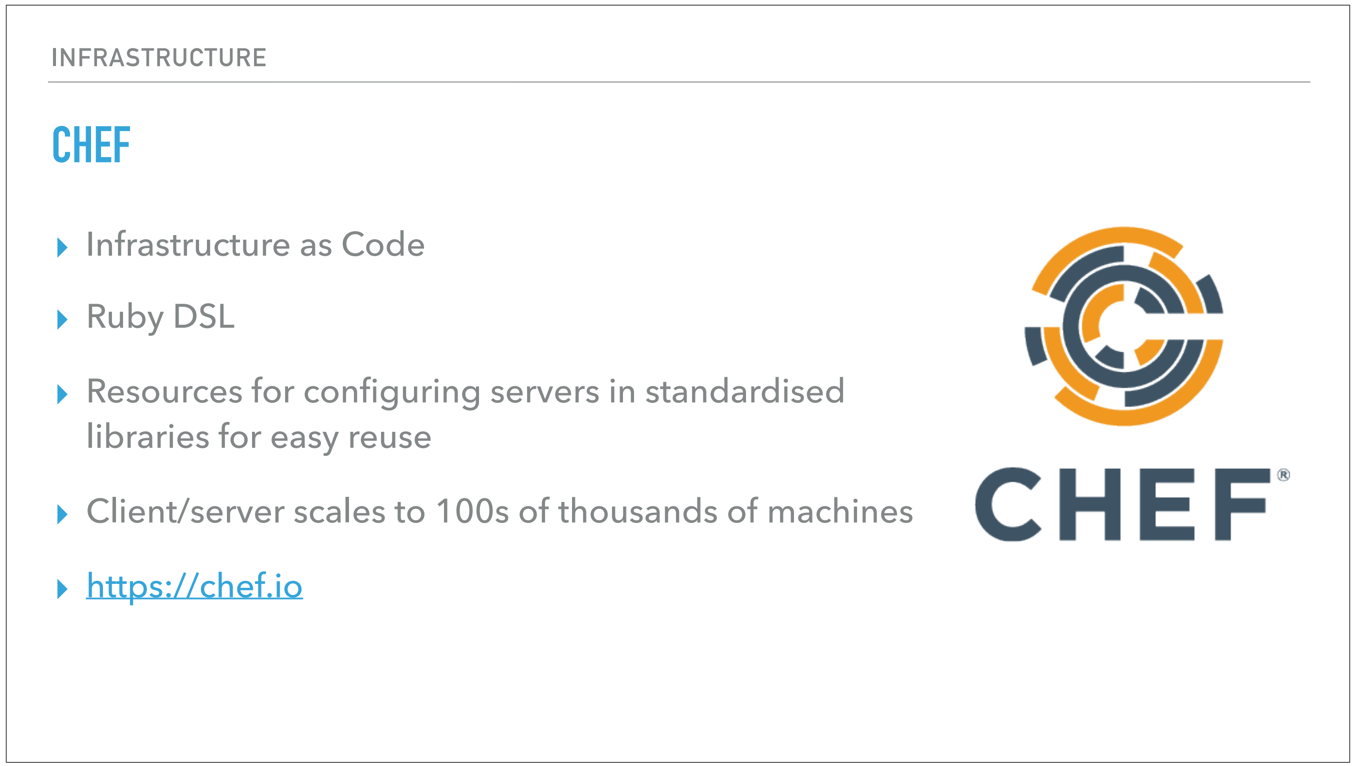
```
control 'ssh-1234' do
  impact 1.0
  title 'Server: Set protocol version to SSHv2'
  desc "
    Set the SSH protocol version to 2. Don't use legacy
    insecure SSHv1 connections anymore...
  "
  describe sshd_config do
    its('Protocol') { should eq('2') }
  end
end
```

Translate compliance into Code
Clearly express statements of policy
Move risk to build/test from runtime
Find issues early
Write code quickly
Run code anywhere
Inspect machines, data and APIs
100+ built-in resources

DEMO

EMACS

- Look at hello/controls/example.rb
- Kitchen verify
- Chmod +x hello.sh
- Kitchen verify
- Let's check on the command output, paste it in to our matcher
- It fails, why?
- Fix, kitchen verify
- Kitchen destroy
- Kitchen verify, now it's gone and broken



CHEF

- Infrastructure as Code
- Ruby DSL
- Resources for configuring servers in standardised libraries for easy reuse
- Client/server scales to 100s of thousands of machines
- https://chef.io

Configuration Management,
▪ Managesdeployment and on-going automation
▪ Definereusableresources
and infrastructure state as code
▪ Scale elegantly from one to tens of thousands of managed nodes across multiple complex environments
▪ Community, Certified Partner, and Chef supported content available for all common automation tasks

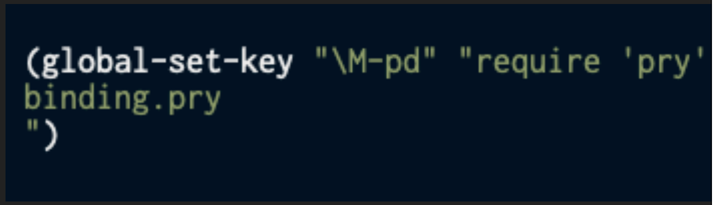
DEMO

EMACS

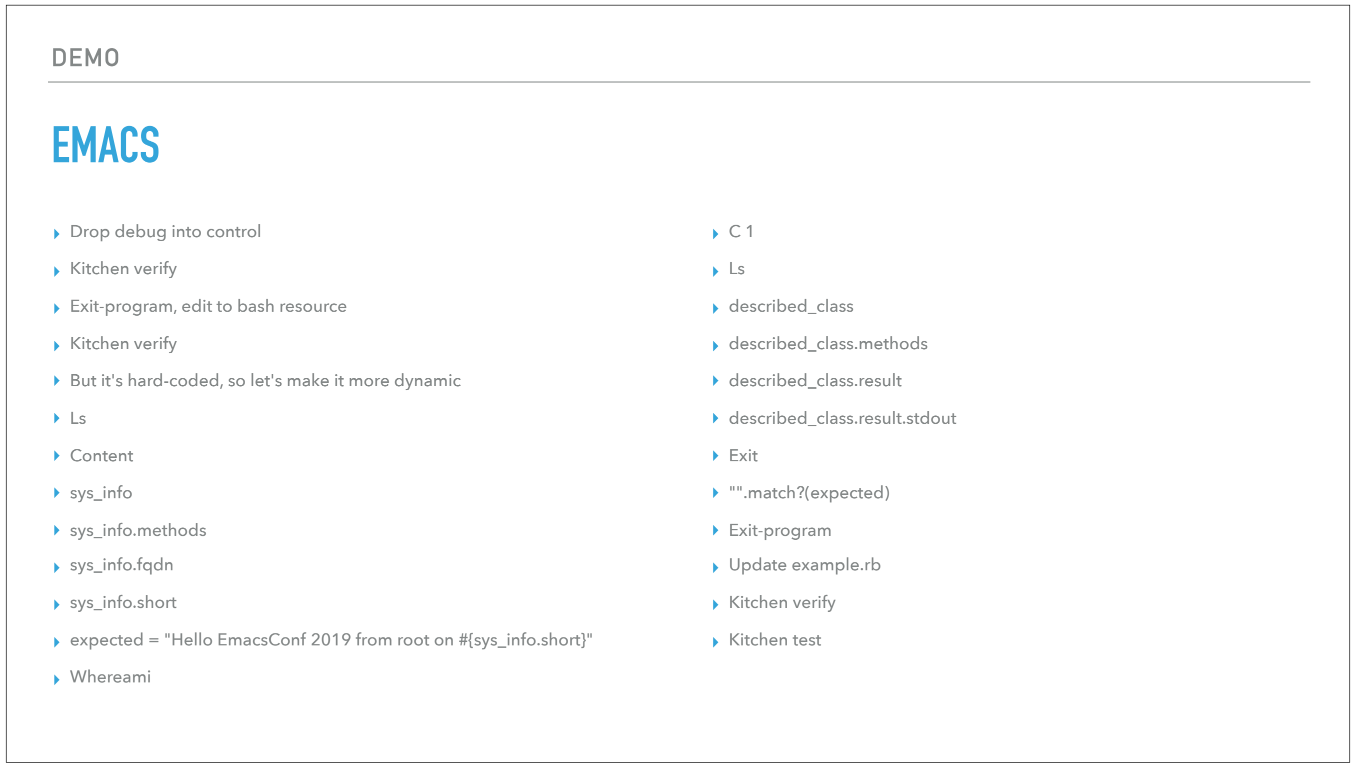
- Let's ensure that our file is always there
- Add the file resource
- Kitchen converge
- Kitchen verify
- Permissions again
- Mode '0755'
- Kitchen converge
- Kitchen verify
- Kitchen test

PRY

- Interactive Ruby debugger
- Built into Chef and InSpec
- Drop breakpoints into code
- ~/.emacs.d/ruby.el



```
(global-set-key "\M-pd" "require 'pry'
binding.pry
")
```


DEMO

EMACS

- Drop debug into control
- Kitchen verify
- Exit-program, edit to bash resource
- Kitchen verify
- But it's hard-coded, so let's make it more dynamic
- Ls
- Content
- sys_info
- sys_info.methods
- sys_info.fqdn
- sys_info.short
- expected = "Hello EmacsConf 2019 from root on #{sys_info.short}"
- Whereami
- C 1
- Ls
- described_class
- described_class.methods
- described_class.result
- described_class.result.stdout
- Exit
- "".match?(expected)
- Exit-program
- Update example.rb
- Kitchen verify
- Kitchen test

Combining TRAMP with Shell Mode and local desktop testing with Kitchen make for a very fast feedback loop.
I can test through the shell and through Ruby without leaving Emacs, going back and forth between my Chef infrastructure code and my InSpec compliance code.
You might not be using these tools, but this pattern of rapid development feedback should be available to you no matter what you're working with.
If you're a system administrator, you don't need to have screen or tmux anymore.
If you're a developer you can deploy your applications into VM and test their behaviour in a clean room environment, using the same configurations as production environments.
This was just a quick dive into Shell Mode, TRAMP, Chef, InSpec and Pry but hopefully you can take some of these ideas into your own Emacs setup and day to day workflow.

THANKS!

https://github.com/mattray/emacsconf2019

emacs@mattray.dev

This was just a quick dive into Shell Mode, TRAMP, Chef, InSpec and Pry but hopefully you can take some of these ideas into your own Emacs setup and day to day workflow.
I've put this demo up on GitHub, there's a link to my emacs and bash configuration there as well.
Thanks again for allowing me to present, I'm available in IRC for any questions you may have or you can email me at emacs@mattray.dev