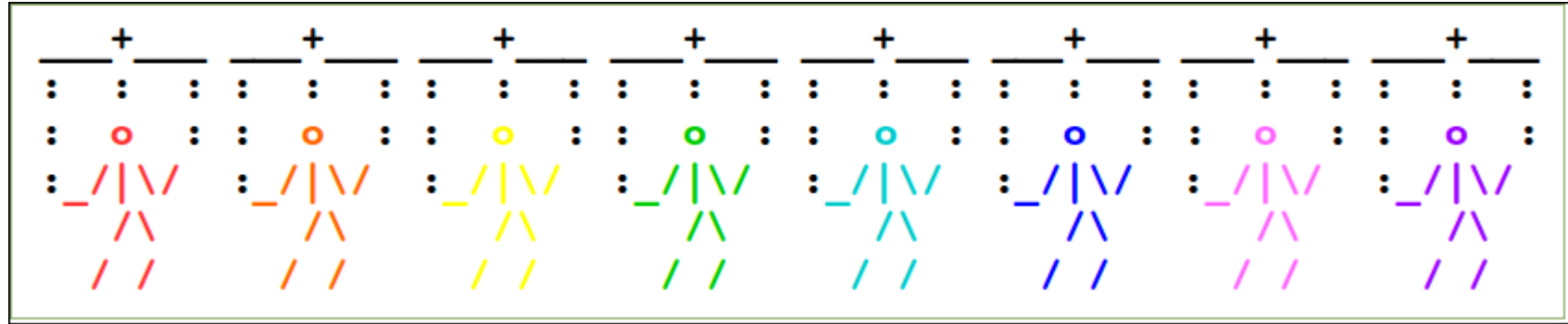


Systems & Infrastructure as Code



"Ever tried. Ever failed.
No matter. Try Again.
Fail again. Fail better."
- Samuel Beckett

Processing this presentation

- If you were not able to attend Penguicon 2017, this presentation has been exported with the notes.
- You will see duplicate slides after the license slide, but the duplicate slides contain my notes.

About Me

- Married, two cats
- Lifelong Type 1 Diabetic
- Michigan Native
- Systems Administrator && Programmer
- Ability to reach things on top of the refrigerator
- Very amateur woodworker
- Maintainer of vim-hashicorp-tools



philporada@gmail.com && <https://keybase.io/pgporada>

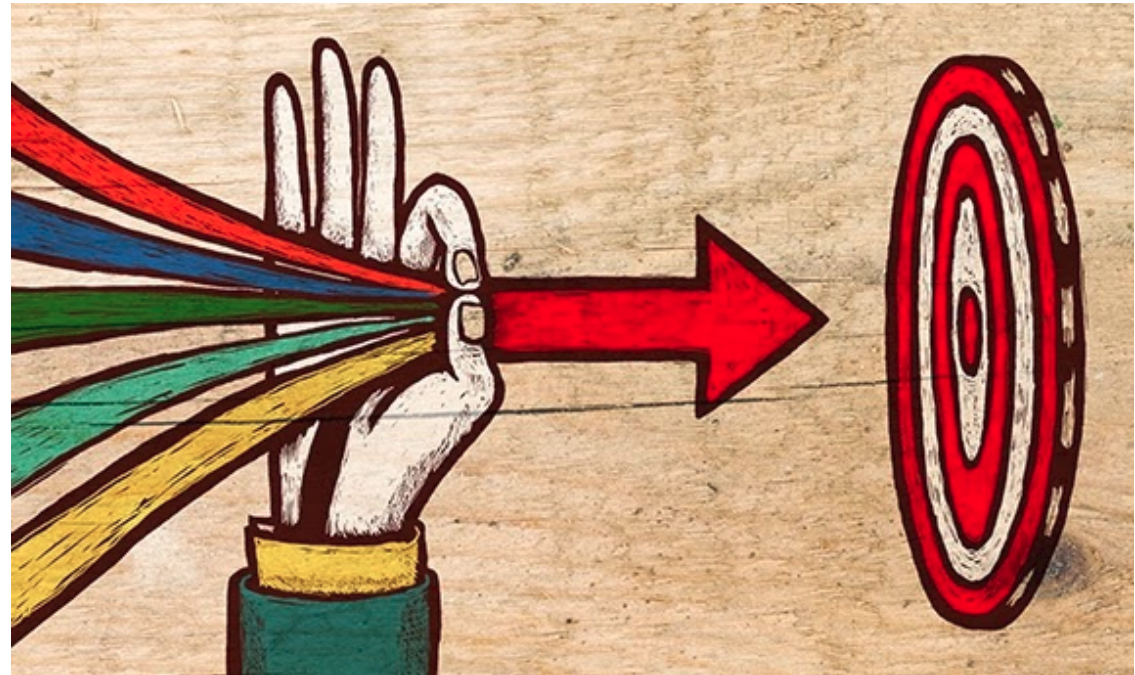
Why?

- Be the admin you needed when you were younger
- Seeing config management helps to troubleshoot problems because you aren't starting from scratch



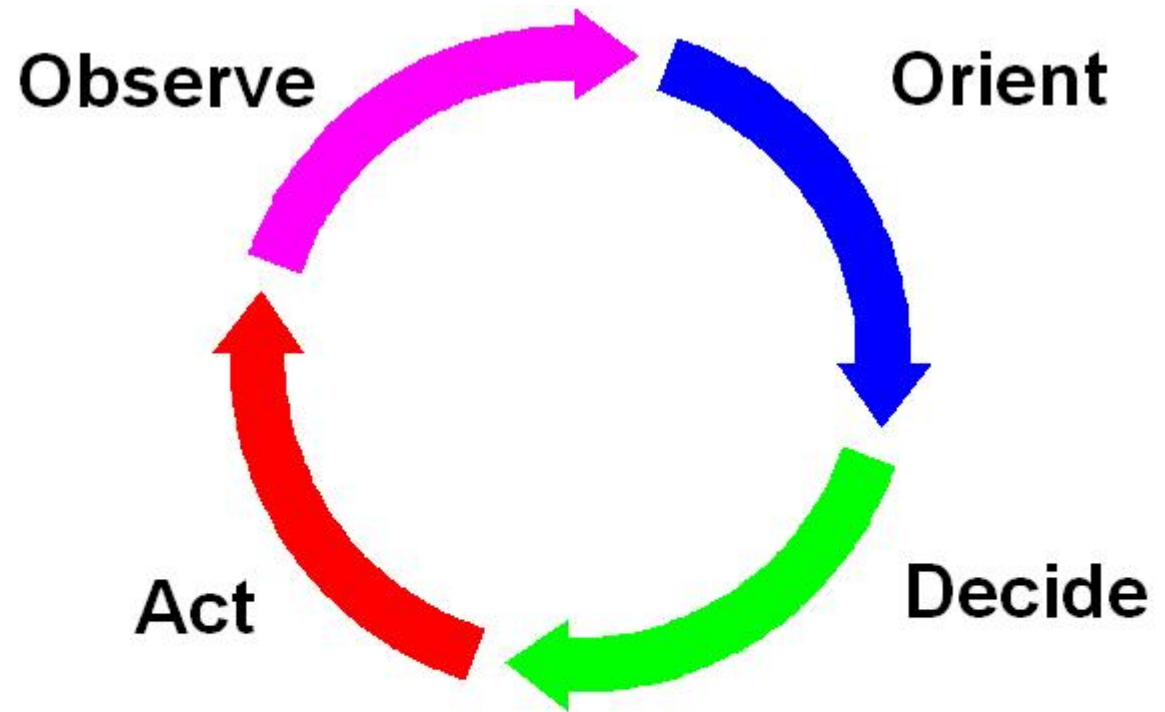
The Gist

- OODA Loop
- Planning
- Developing
- Deploying
- Promoting
- Documenting
- Backups and Restores
- Testing
- Demo
- Questions



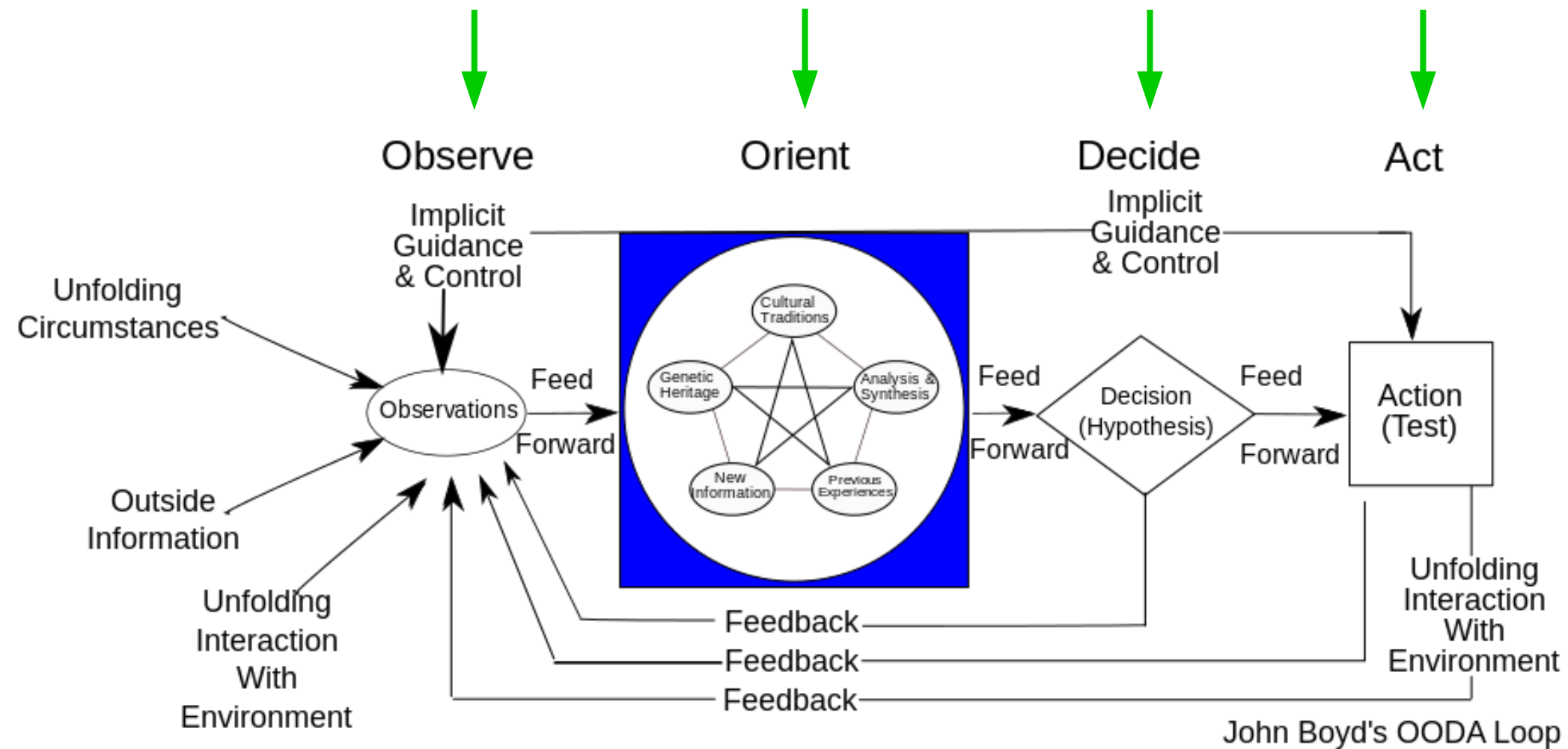
OODA Loop pt1

- Observe
- Orient
- Decide
- Act



Slow OODA loop = death
Fast OODA loop = success

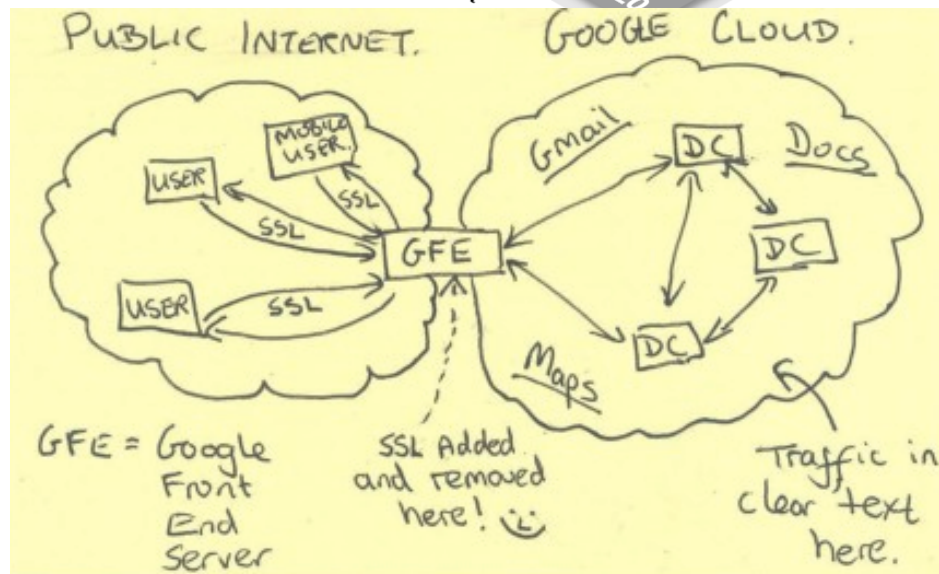
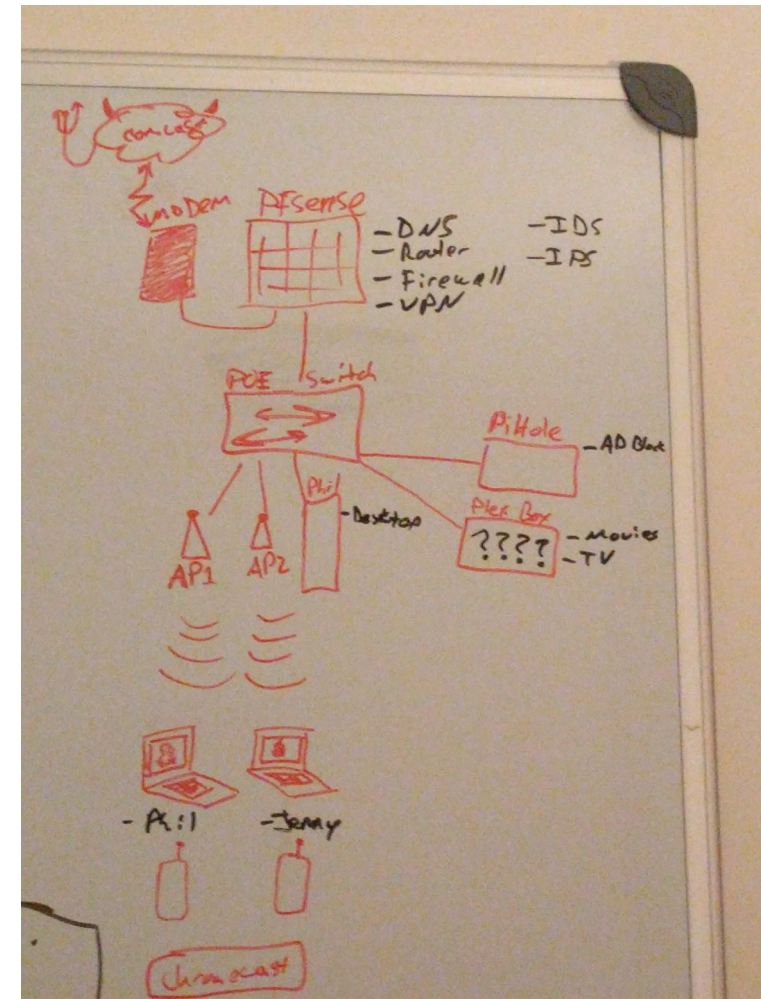
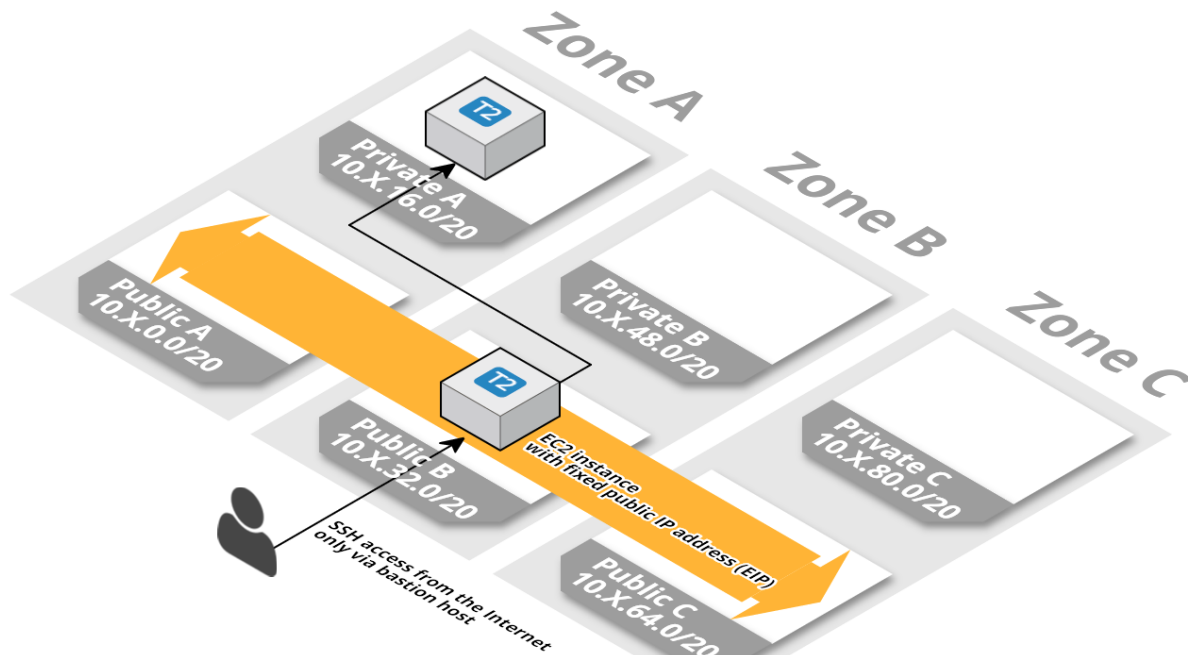
OODA Loop pt2



Planning

- Listen to your team
- Note the skills of your team
- Ask questions
- Research potential technology
- Check mailing lists, reddit, IRC, etc
- Notepad/napkin sketches
- Visio, Cloudcraft.co, Lucid Charts, Gliffy, etc
- Get feedback

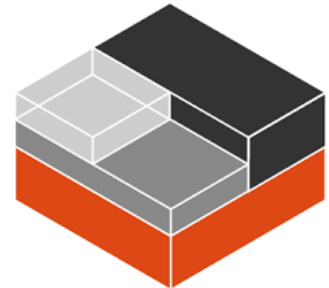
Planning – Visualize your system



Developing Infrastructure - Servers

- Phase 1

- Vagrant via virtualbox or libvirt
- Containers
- Free and fast



- Phase 2

- If you can do it in the cloud – AWS/GCE/Azure
- If you have spare physical gear – great
- Costly, but more of a “true” environment

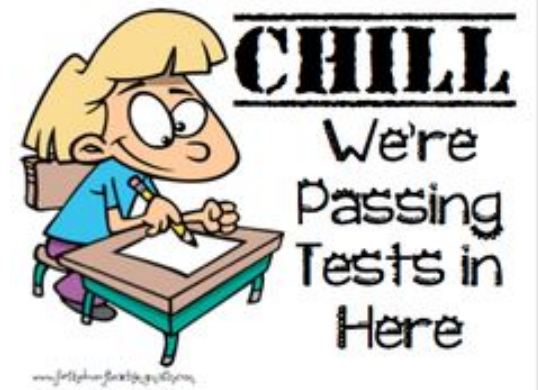
Developing Infrastructure – Networks

- Phase 1
 - GNS3, a little wonky but can load networking images from multiple vendors _(ツ)_/
 - Packet Tracer for Cisco
- Phase 2
 - Spare physical gear



Testing – Trust but verify

- How do you know your code is doing what you think it's doing?
- How do others know that your code is doing what you think it's doing?
- How do you know that others code is doing what they say it's doing?
- Allows managing confidence levels via code



Testing pt2 – Testing is hard



More Testing

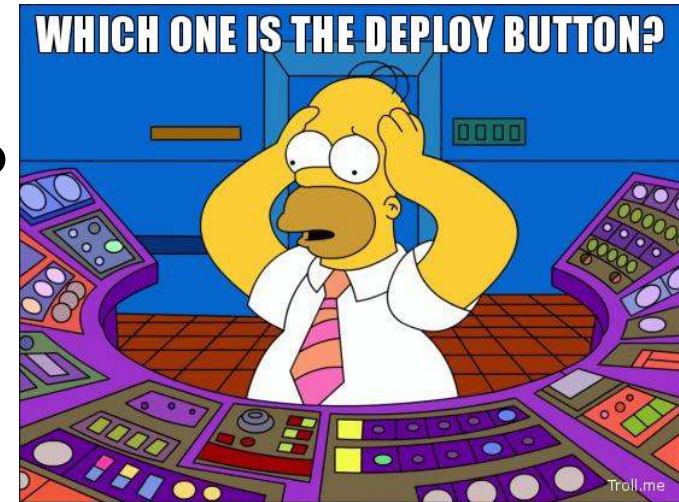


Testing – How I do it

- **Packer** runs some **ServerSpec** tests after building the base image to ensure certain aspects have been configured and are as the configuration management defined
- <https://github.com/pgporada/packer-centos7-devenvironment>
- **Test-kitchen** and **BATS** allow you to run tests locally before pushing your changes back upstream to your team.
- <https://github.com/pgporada/ansible-role-terraform>
- <https://github.com/pgporada/ansible-role-cve>
- **Terraform** to spin up infrastructure in AWS/GCE/OpenStack/vSphere/etc
- <https://github.com/pgporada/terraform-bastion> Please note that this project won't work verbatim for you. I used terraform as a way to teach myself how Amazon is put together and how all their little pieces interact. This project uses user_data to run private **ansible** code to post-provision a server for me upon boot.

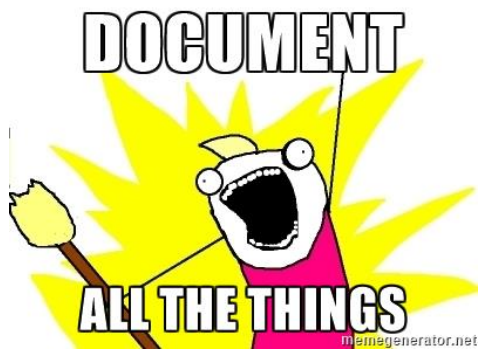
Deploying

- Are the instructions documented?
 - If not, why?
- Is automation in place to do the deploy?
 - If not, why?
- Do deploys happen from employee machines or from a centralized deploy server?



Promoting code

- What are your procedure/processes/instructions to promote code from dev → staging → prod?
- You have a staging/test environment... right?
- You have a rollback strategy...right?



Documenting



- Helps train future engineers
- Always able to refer back to it
- Can link to it via alerts/monitoring panes
- Searchable if you take time to add labels/keys
- Allows you to pass tools to teammates who may not have experience with that tool
- There's only one of you. Let it be a force multiplier.
- Leave docs better than when you found them

Documenting – Don't be an ass

Dashboard > MiChart > ... > MiChart Web Services > User Request Parameters

Search

- ☐ Contact Information
- ☒ MiChart Web Services
 - ☒ Web Service Reports
 - ☐ Web Service Environments
 - ☐ Context Diagram
 - ☐ Example Using JDK 1.6 Only
 - ☐ Example Using Spring
 - ☐ Interconnect Administration
 - ☐ MiChart Web Services WSDL

M MICHIGAN MEDICINE UNIVERSITY OF MICHIGAN

User Request Parameters

Added by Stephen Scott Fayz , last edited by Stephen Scott Fayz on Sep 24, 2013 (view change)

User Request Parameters

Each web service requires a UserID and UserIDType which must be set to: **UserID**={Users Unique Name} and **UserIDType**=SYSTEMLOGIN. However, due to a bug in Epic, GetActiveProblemList should use "External" for the **UserIDType**. The UserID should be the user logging into your application and NOT a group account.

last edited by Stephen Scott Fayz on Sep 24, 2013

Edit - Working Environment - T x

50.56.102.168:9090/pages/editpage.action?pageId=2720118

Dashboard > Thats Enterprising Intranet > ... > Working Environment

Browse Summer El-Zorkany Search this space

Working Environment

Working Environment



Documentation – Help each other!

https://confluence.desy.de/display/PUP/Puppet+Hands-On

Confluence

Puppet

Pages

Blog

PAGE TREE

- Git Hands-On
- Hiera classes
- Puppet Classes
- Puppet Hands-On
 - puphandson_solution1
 - puphandson_solution2
 - puphandson_solution3
 - puphandson_solution4
 - puphandson_solution5
 - solution_a
 - solution_b

Pages / Puppet Home

Puppet Hands-On

Sven Sternberger created on 20. Apr. 2017 14:13h - last edited by Sven Sternberger on 23. Apr. 2017 21:49h

Environment

1. Login via ssh to the workgroup server naf-school06.desy.de with your assigned account
2. Check if you can login via ssh to your assigned VM as **root** - hint: An ssh key-pair was generated and put into your home directory on your VM
3. Check if you can edit a text file on the wgs and the VM node. We provide vim, emacs, nano

First steps

1. Use the puppet RAL to examine and configure the **VM** node [solution](#)
 - a. Show all managed resources
 - b. Show description of the resource service
 - c. Show all configured services
 - d. Create a testuser
 - e. Show resource of the created user
 - f. Delete the testuser
 - g. HINT:
 - puppet resource --help
 - puppet describe
 - <https://docs.puppet.com/puppet/latest/reference/type.html#user>

Manifests

1. Write the first Manifest [solution](#)
 - a. Create a puppet manifest which creates two testuser. Filename: *handson1create.pp*
 - b. Create a puppet manifest which removes both testuser. Filename: *handson1remove.pp*
 - c. Concatenate both puppet manifest. Filename: *handson1combine.pp*
 - d. Apply the manifests to your node.
 - e. Try to understand why the *handson1combine.pp* manifest fails?

Documentation – We're still here?

- Document to automate because the documentation will eventually be disposed
- You can't automate something you've never done manually
- Don't be this Homer



Backups and Restores

- Backups are important
- Restores are infinitely more important
- Is your backup and restore process automated?
 - If not, why?
- What's your mean time to recovery for problems?
- Is your backup/restore code versioned so that it can be iterated upon?

Live Demo




Questions?

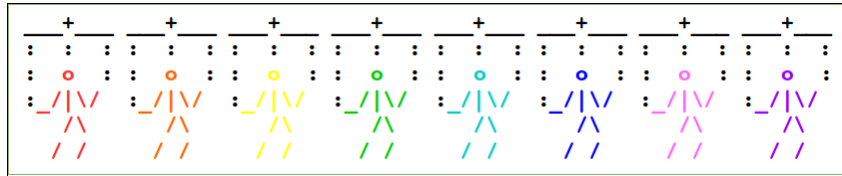
- philporada@gmail.com
- <https://github.com/pgporada>
- <https://keybase.io/pgporada>



License

- The logo for the Creative Commons Attribution (CC BY) license. It consists of a rounded rectangle divided into two horizontal sections. The top section is light gray and contains two circular icons: the first with the letters 'cc' and the second with a stylized person icon. The bottom section is black and contains the text 'BY' in white.
- This work is licensed under a Creative Commons Attribution 4.0 International License.
- <https://creativecommons.org/licenses/by/4.0/>

Systems & Infrastructure as Code



"Ever tried. Ever failed.
No matter. Try Again.
Fail again. Fail better."
- Samuel Beckett

Samuel Beckett is a French badass

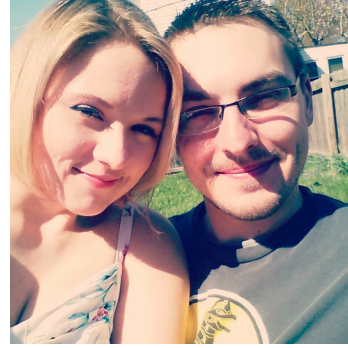
ASCII marionette by RetroJunkie Ascii ART page

Processing this presentation

- If you were not able to attend Penguicon 2017, this presentation has been exported with the notes.
- You will see duplicate slides after the license slide, but the duplicate slides contain my notes.

About Me

- Married, two cats
- Lifelong Type 1 Diabetic
- Michigan Native
- Systems Administrator & Programmer
- Ability to reach things on top of the refrigerator
- Very amateur woodworker
- Maintainer of vim-hashicorp-tools



philporada@gmail.com && <https://keybase.io/pgporada>

It's me!

Why?

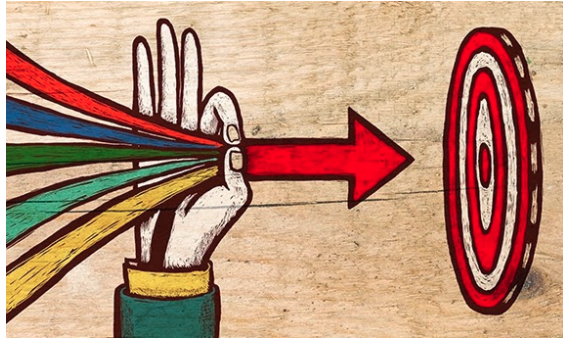
- Be the admin you needed when you were younger
- Seeing config management helps to troubleshoot problems because you aren't starting from scratch



- When we were junior engineers we would always look up to the senior engineers and wonder how they did the things they did.
- We can help the next generation of engineers by providing them with better docs, better code, better processes/procedures/instructions
- New skills and technology will keep YOU learning.
- Don't ever stop learning

The Gist

- OODA Loop
- Planning
- Developing
- Deploying
- Promoting
- Documenting
- Backups and Restores
- Testing
- Demo
- Questions

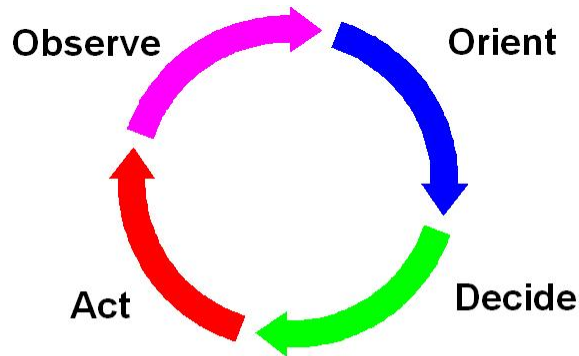


Don't take this list verbatim

Acquired this image from Google Images

OODA Loop pt1

- Observe
- Orient
- Decide
- Act



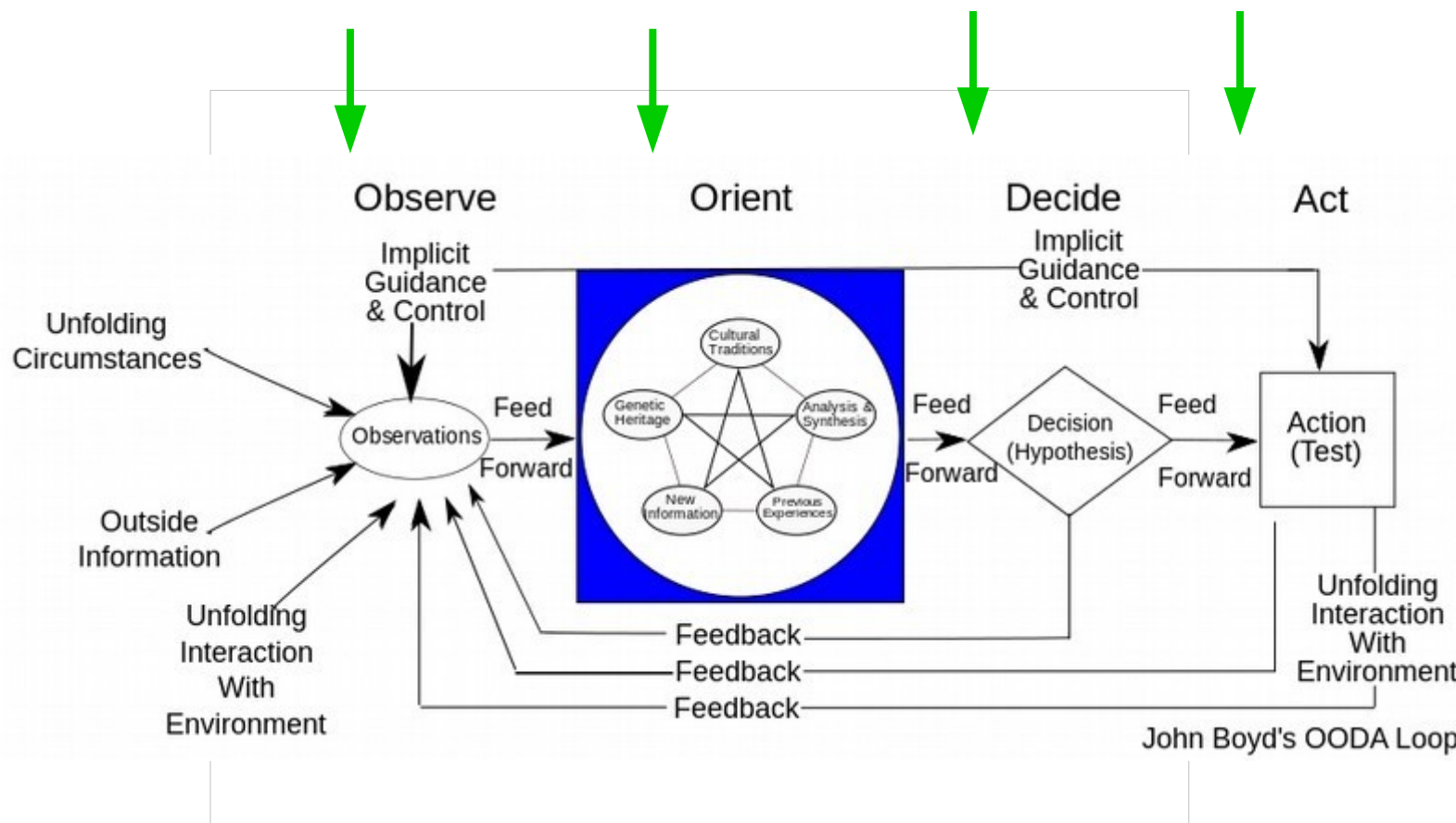
Slow OODA loop = death
Fast OODA loop = success

John Boyd – military strategist

We do these steps continuously just by existing.

Monitoring and alerting help us to observe and orient so that we can decide on the path to take.

It is a completely valid thought to decide to wait for an alert to fire multiple times before acting upon that alert.



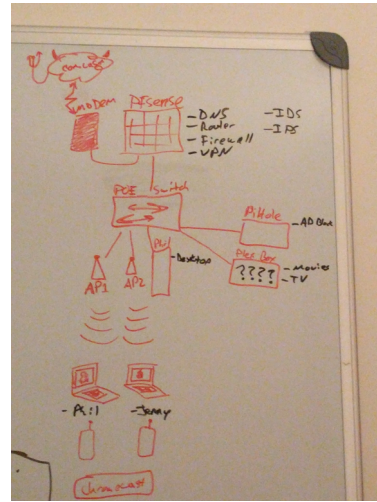
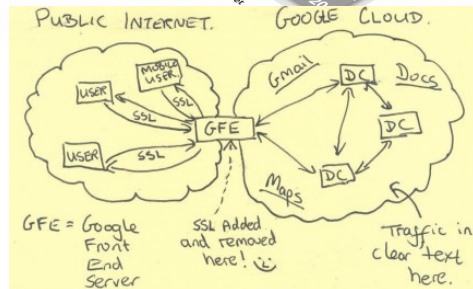
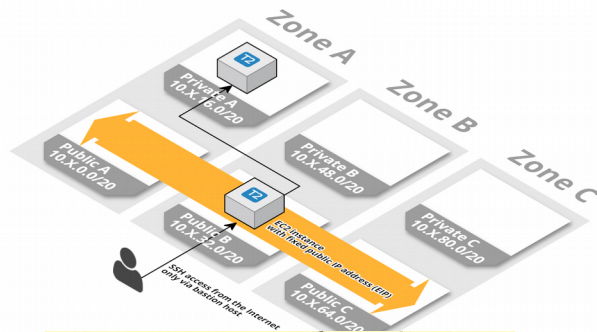
This is the real OODA loop with all the subtleties

Planning

- Listen to your team
- Note the skills of your team
- Ask questions
- Research potential technology
- Check mailing lists, reddit, IRC, etc
- Notepad/napkin sketches
- Visio, Cloudcraft.co, Lucid Charts, Gliffy, etc
- Get feedback

Kung Fu tv show - “I seek not to know the answers, but to understand the questions”

Planning – Visualize your system



Cloudcraft.co network diagram

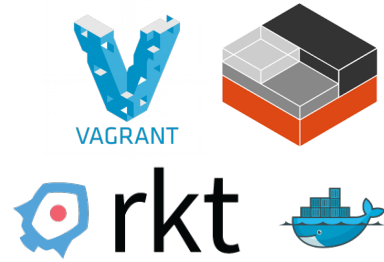
Very old and since restructured home network

NSA PRISM napkin sketch

Developing Infrastructure - Servers

- Phase 1

- Vagrant via virtualbox or libvirt
- Containers
- Free and fast



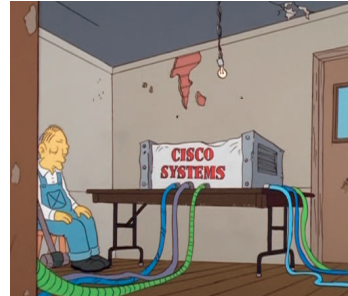
- Phase 2

- If you can do it in the cloud – AWS/GCE/Azure
- If you have spare physical gear – great
- Costly, but more of a “true” environment

Developing Infrastructure – Networks

- Phase 1
 - GNS3, a little wonky but can load networking images from multiple vendors _(͡°)_
 - Packet Tracer for Cisco

- Phase 2
 - Spare physical gear



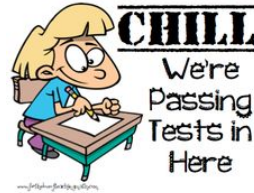
Back up your network configs

- Rancid
- Git
- Svn (ew)

Just do it, ok

Testing – Trust but verify

- How do you know your code is doing what you think it's doing?
- How do others know that your code is doing what you think it's doing?
- How do you know that others code is doing what they say it's doing?
- Allows managing confidence levels via code



Kitchen comes from the Chef world, but is useful to test all configuration management codebases.

Pytest is a framework for writing tests in Python. Alternatives to that are writing tests in Ruby via Serverspec/InSpec or writing tests in Bash via the Bash Automated Test Suite aka BATS

The test language doesn't matter, having tests is what matters. I am a programmer that uses ruby tools to test python tools with bash. Does it work? Yes. Will you do it the same way? Maybe. Does it matter that you're doing it? Absolutely.

Testing pt2 – Testing is hard



Stop prayer based development

Failing tests are ok on a developer (sysadmins) machine because failure is how you get better and learn

The faster you fail, the faster you fix problems

More Testing



Don't be Dwight Schrute or this little girl.

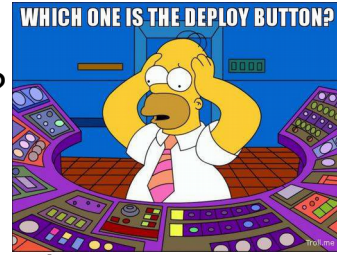
Be like Darth Vader

Testing – How I do it

- **Packer** runs some **ServerSpec** tests after building the base image to ensure certain aspects have been configured and are as the configuration management defined
- <https://github.com/pgporada/packer-centos7-devenvironment>
- **Test-kitchen** and **BATS** allow you to run tests locally before pushing your changes back upstream to your team.
- <https://github.com/pgporada/ansible-role-terraform>
- <https://github.com/pgporada/ansible-role-cve>
- **Terraform** to spin up infrastructure in AWS/GCE/OpenStack/vSphere/etc
- <https://github.com/pgporada/terraform-bastion> Please note that this project won't work verbatim for you. I used terraform as a way to teach myself how Amazon is put together and how all their little pieces interact. This project uses user_data to run private **ansible** code to post-provision a server for me upon boot.

Deploying

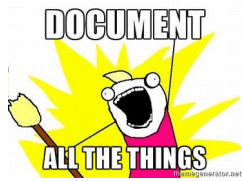
- Are the instructions documented?
 - If not, why?
- Is automation in place to do the deploy?
 - If not, why?
- Do deploys happen from employee machines or from a centralized deploy server?



Please don't sue me Fox.

Promoting code

- What are your procedure/processes/instructions to promote code from dev → staging → prod?
- You have a staging/test environment... right?
- You have a rollback strategy...right?



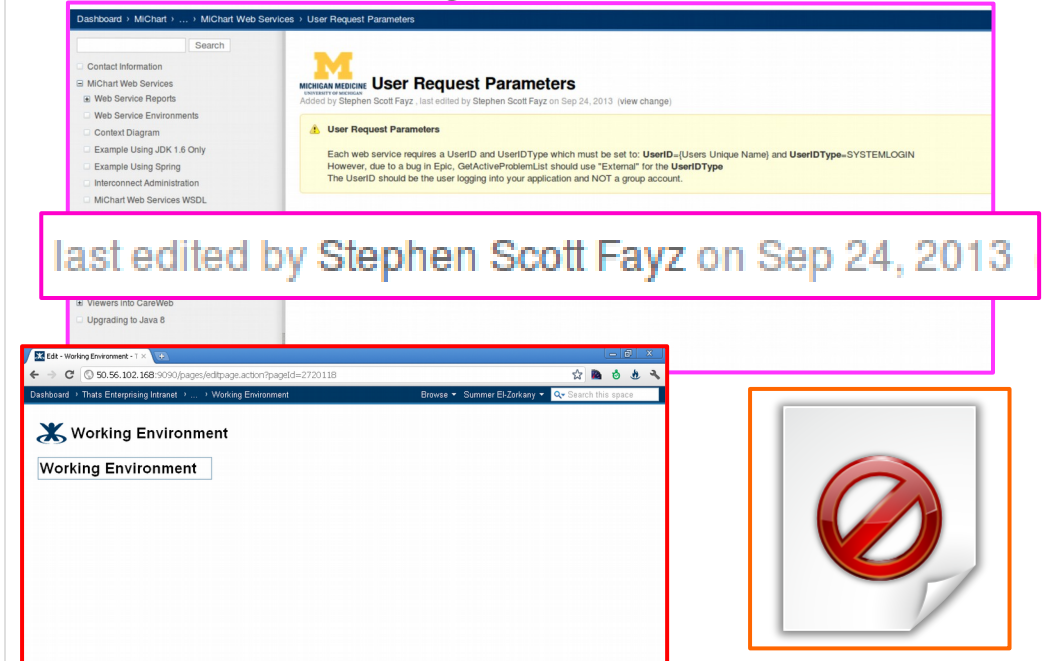
Documenting



- Helps train future engineers
- Always able to refer back to it
- Can link to it via alerts/monitoring panes
- Searchable if you take time to add labels/keys
- Allows you to pass tools to teammates who may not have experience with that tool
- There's only one of you. Let it be a force multiplier.
- Leave docs better than when you found them

Pic is the Adepticus Mechanicus from WH40k and a meme from the Hyperbole and a Half comic

Documenting – Don't be an ass



Last edited fucking forever ago

A page with no content just like a company with no employees

What's a wiki? ...

Documentation – Help each other!

The screenshot shows a Confluence page titled "Puppet Hands-On" under the "Puppet" space. The page is structured with a sidebar on the left and a main content area on the right. The sidebar contains a "PAGE TREE" with a list of pages, including "Puppet Hands-On" and several sub-pages. The main content area has a header with the page title and author information, followed by sections for "Environment", "First steps", and "Manifests". Green arrows point to specific elements: the "Puppet" space name, the "Puppet Hands-On" page title, the "Puppet Hands-On" page in the sidebar, the "Environment" section, the "First steps" section, and a link to the Puppet documentation.

https://confluence.desy.de/display/PUP/Puppet+Hands-On

Confluence

Puppet

Pages / Puppet Home

Puppet Hands-On

Sven Sternberger on 20. Apr. 2017 14:13h - last edited by Sven Sternberger on 23. Apr. 2017 21:49h

Environment

1. Login via ssh to the workgroup server naf-school06.desy.de with your assigned account
2. Check if you can login via ssh to your assigned VM as **root** - hint: An ssh key-pair was generated and put into your hon your VM
3. Check if you can edit a text file on the wgs and the VM node. We provide vim, emacs, nano

First steps

1. Use the puppet RAL to examine and configure the VM node [solution](#)
 - a. Show all managed resources
 - b. Show description of the resource service
 - c. Show all configured services
 - d. Create a testuser
 - e. Show resource of the created user
 - f. Delete the testuser
 - g. HINT:
 - puppet resource --help
 - puppet describe
 - <https://docs.puppet.com/puppet/latest/reference/type.html#user>

Manifests

1. Write the first Manifest [solution](#)
 - a. Create a puppet manifest which creates two testuser. Filename: *handson1create.pp*
 - b. Create a puppet manifest which removes both testuser. Filename: *handson1remove.pp*
 - c. Concatenate both puppet manifest. Filename: *handson1combine.pp*
 - d. Apply the manifests to your node.
 - e. Try to understand why the *handson1combine.pp* manifest fails?

Space name is relevant to what a person would find inside

The page is relevant to this year or has an edit trail.
Don't leave docs with 1 entry to just fester. Clean that stuff up.

Use of headers

Actual commands to run and links to other documentation

Documentation – We're still here?

- Document to automate because the documentation will eventually be disposed
- You can't automate something you've never done manually
- Don't be this Homer



Backups and Restores

- Backups are important
- Restores are infinitely more important
- Is your backup and restore process automated?
 - If not, why?
- What's your mean time to recovery for problems?
- Is your backup/restore code versioned so that it can be iterated upon?

Cron + tar + gzip + rsync

Tarsnap

rdiffbackup

Amanda

Bacula because Bareos had the main developer leave

Live Demo



Packer to build an image that we can deploy in Vagrant and also an AMI that we can deploy in AWS

Git + Test-kitchen + Vagrant + Ansible

- Run converge
- Run tests

Git + Terraform + Ansible

- Run a plan
- Run apply
- Run destroy then apply

Jenkins push button deploy + Slack message for alerts of when deploys start/finish


Questions?

- philporada@gmail.com
- <https://github.com/pgporada>
- <https://keybase.io/pgporada>



My wife and I carved this pumpkin some Halloweens ago

License

- The logo consists of two circular icons: the first contains the letters 'cc' and the second contains a stylized person icon. Below these icons, the letters 'BY' are displayed.
- This work is licensed under a Creative Commons Attribution 4.0 International License.
- <https://creativecommons.org/licenses/by/4.0/>