DevOps

SwiNOG-29
November 5th, 2015
Gurtenpark, Berne

André Keller, VSHN AG

about me

- System-Engineer at VSHN AG since Nov 2014

- AtrilA GmbH (2010 – 2014)
- Network Design GmbH (2005 – 2012)

- @andrekeller_ch
- https://github.com/andrekeller
- andre.keller@vshn.ch

Agenda

- What is this DevOps thing?
  - Culture & Tools
- A few words about GIT
- A few words about Vagrant
- Puppet

DevOps

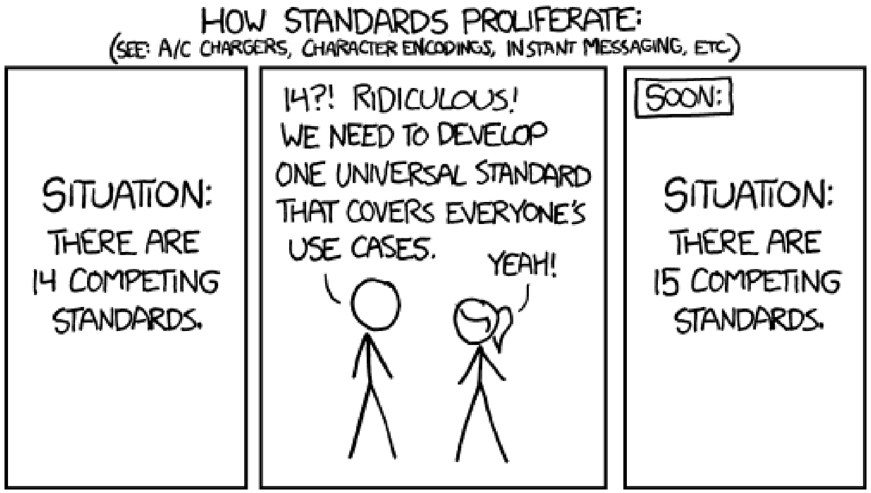
- Collaboration: Development (Dev) and Operations (Ops)
- Bring agile software engineering methods to operations
  - Automation: infrastructure as a code, versioning/rollback
  - Testing: continuous integration/testing/deployment
- Bring operations engineering experience to developers
  - Scalability: independent microservices
  - Production insight: monitoring/logging/metrics
- Together: make the application's owner happier

Infrastructure as code

- Change from hand-groomed servers to Operations Engineering (from pets to cattle)
- Speed & reliability
- Versioning & rollback
- Prerequisite for self-service
  - Give each developer a full stack
  - No manual changes in production
  - As many testing instances as needed

Tools

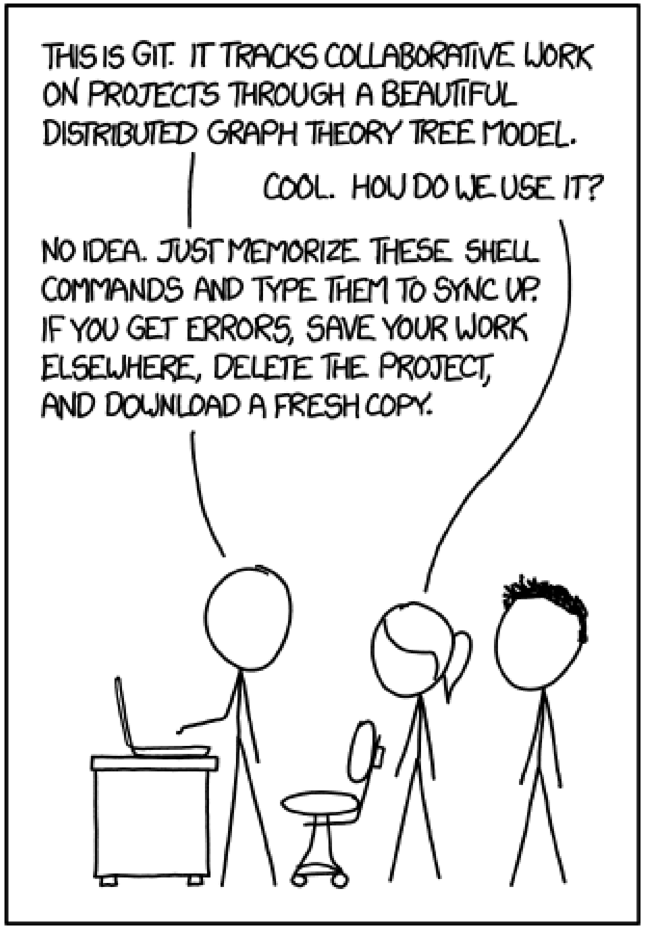
- SCM, Version Control
  - **git**, subversion, mercurial, cvs
- Packaging (code & dependencies)
  - deb, rpm, docker
- Infrastructure state management (configuration mgmt)
  - **Puppet**, saltstack, chef, ansible, cfengine, Fabric
- Continuous Integration/Testing/Deployment
  - Jenkins, TravisCI, GitlabCI
- Self-Service
  - **Vagrant**, Otto, Docker


HOW STANDARDS PROLIFERATE:
(SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC)
SITUATION: THERE ARE 14 COMPETING STANDARDS.
14?! RIDICULOUS! WE NEED TO DEVELOP ONE UNIVERSAL STANDARD THAT COVERS EVERYONE'S USE CASES.
YEAH!
SOON:
SITUATION: THERE ARE 15 COMPETING STANDARDS.

https://xkcd.com/927/

GIT

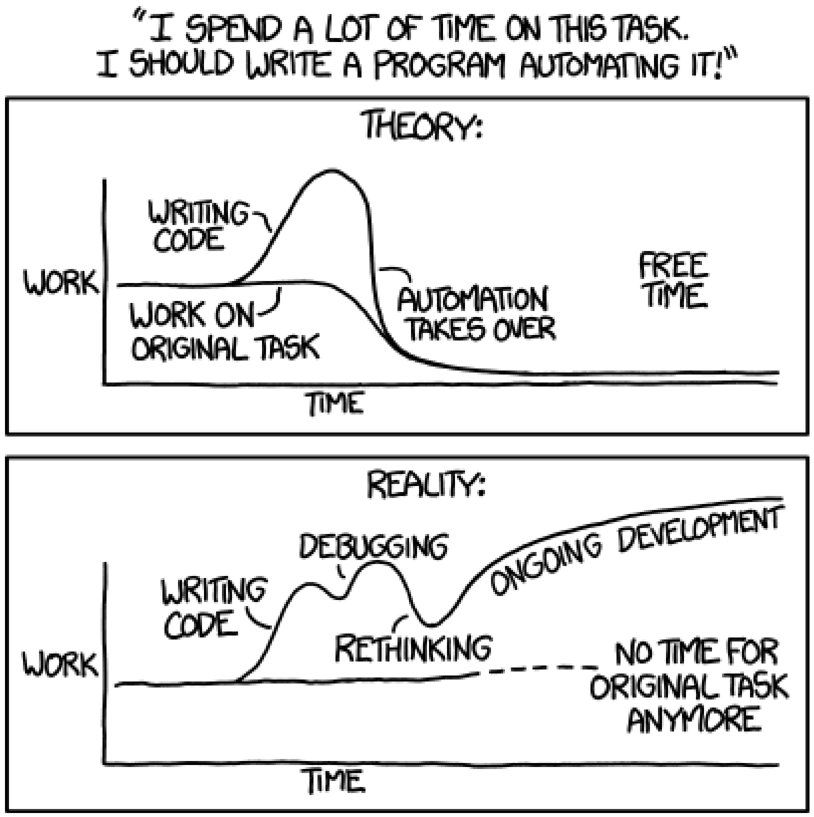
- Popular DVCS
- Hosting git repositories:
  - github.com
  - bitbucket.com
  - gitlab.com
- Selfhosting:
  - All of the above (VSHN uses GitLab)
- New to git?
  - https://try.github.io/


THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.
COOL. HOW DO WE USE IT?
NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.

https://xkcd.com/1597/

Vagrant

- Create and configure lightweight, reproducible, and portable development environments.
- Leverages existing tools such as VirtualBox, VMware, LXC or Libvirt/KVM
- Get it at: https://vagrantup.com/



https://xkcd.com/1319/

Vagrant

- Initialize a new vagrant box:
  - `vagrant init puppetlabs/ubuntu-14.04-64-puppet`

    https://atlas.hashicorp.com/boxes/search
- Provision the new box:
  - `vagrant up`
- SSH into the new box:
  - `vagrant ssh`
- Get rid of the box:
  - vagrant destroy

Puppet

- Configuration management system: You define the state of your infrastructure, puppet enforces it.
- Puppet uses its own declarative language to describe a nodes resources and their states.
- Puppet ships with types and providers to manage basic system resources such as:
  - Files
  - Mounts
  - Packages
  - Services
  - Users
  - Groups
  - SSH Keys
  - Cronjobs

Facter

- Gathers basic facts about nodes such as:
  - hardware details,
  - network settings,
  - OS type and version
  - and much more
- Facts are made available as variables within manifests
- Running `/opt/puppetlabs/bin/facter` will list available facts

Fact example

```
node default {

  case $::osfamily: {
    'debian': {
      $apache_package = 'apache2'
    }
    'redhat': {
      $apache_package = 'httpd'
    }
  }

  package {$apache_package:
    ensure => 'present',
  }

}
```

- Now this simple example will install apache on Debian and RedHat based nodes.

Puppet manifests

- The manifests is where we define our state.
- Example:

```
node default {

  package {'vim':
    ensure => 'present',
  }

  user {'foobar':
    ensure => 'present',
    home => '/home/foobar',
  }

}
```

- This will install the vim package and the foobar user.

Puppet classes and defined types

- Classes or defined types are units of configuration that group several resources.
- Classes can only be used once per node, defined types several times.
- Class example:

```
class editor {
  package {'vim': ensure => present, }
  file {'/etc/vim/vimrc': … }
}
```

- In our manifest we can now use `include editor` instead of defining the resources induvidually.

Defined Type

- Classes and defined types can also take parameters:

```
define systemuser ($username) {

  user {$username:
    ensure => present,
    home => "/home/${username}",
  }->

  file {"/home/${username}":
    ensure => directory,
    owner => $username,
    mode => '0700',
  }

}
```

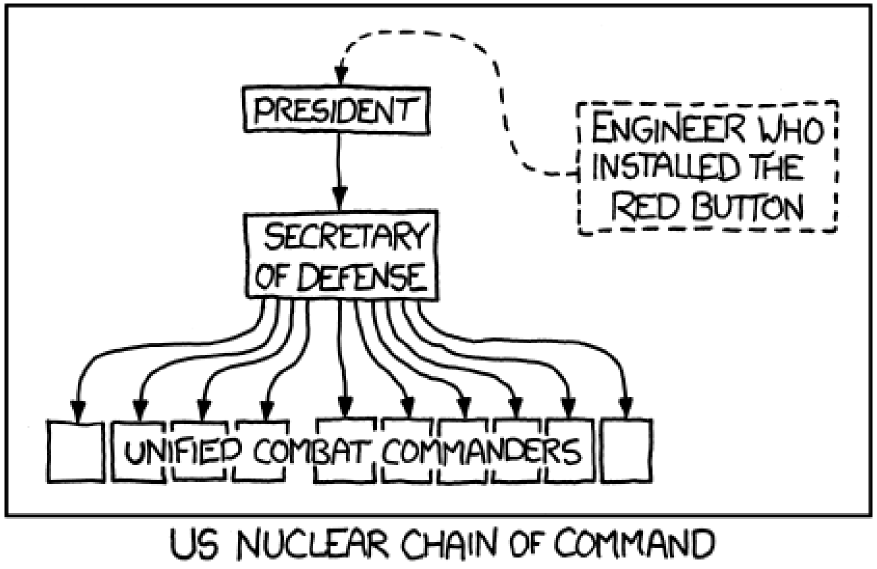
- We can now use `systemuser {'foobar': }` in our manifest

Puppet modules

- Puppet modules group classes for a specific task.
- Examples for available modules are:
  - **puppetlabs/apache**: Installing Apache webservers and configure various virtual host setups.
  - **puppetlabs/postgresql**: Installing Postgresql server, managing postgres users, databases and permissions
  - **vshn/gitlab**: Installation and configuration of GitLab.
- List of available modules https://forge.puppetlabs.com/

Hiera

- Key/value lookup tool for configuration data
- Helps to keep site-specific data out of your manifests
- Lets you build a hierarchy for configuration data, f.e.:
  - common
  - location specific
  - role specific
  - node specific


PRESIDENT
ENGINEER WHO INSTALLED THE RED BUTTON
SECRETARY OF DEFENSE
UNIFIED COMBAT COMMANDERS
US NUCLEAR CHAIN OF COMMAND

https://xkcd.com/898/

Hiera example

- manifest.pp:

```
hiera_include('classes')
```

- hiera.yaml:

```
---
classes:
  - identity

users:
  foobar:
    home: '/home/foobar'
```

Install puppet

- Checkout: http://docs.puppetlabs.com/puppet/latest/reference/install_pre.html
- Instructions for Ubuntu 14.04:
  - `wget https://apt.puppetlabs.com/puppetlabs-release-pc1-trusty.deb`
  - `dpkg -i puppetlabs-release-pc1-trusty.deb`
  - `apt-get update`
  - `apt-get install puppet-agent`

Installation overview

- /etc/puppetlabs/code/environments/production/
  - /manifests/
    - Your manifests. Single manifests loading hiera is recommended, call it `site.pp`
  - /modules/
    - Holds the puppet modules (i.e. from forge.puppetlabs.com)
  - /hieradata/common.yaml
    - Holds the hiera key/value configuration

Install puppet modules

- Install modules from forge.puppetlabs.com:
- `/opt/puppetlabs/bin/puppet module install vshn-identity`

Vagrant Environment for SwiNOG 29

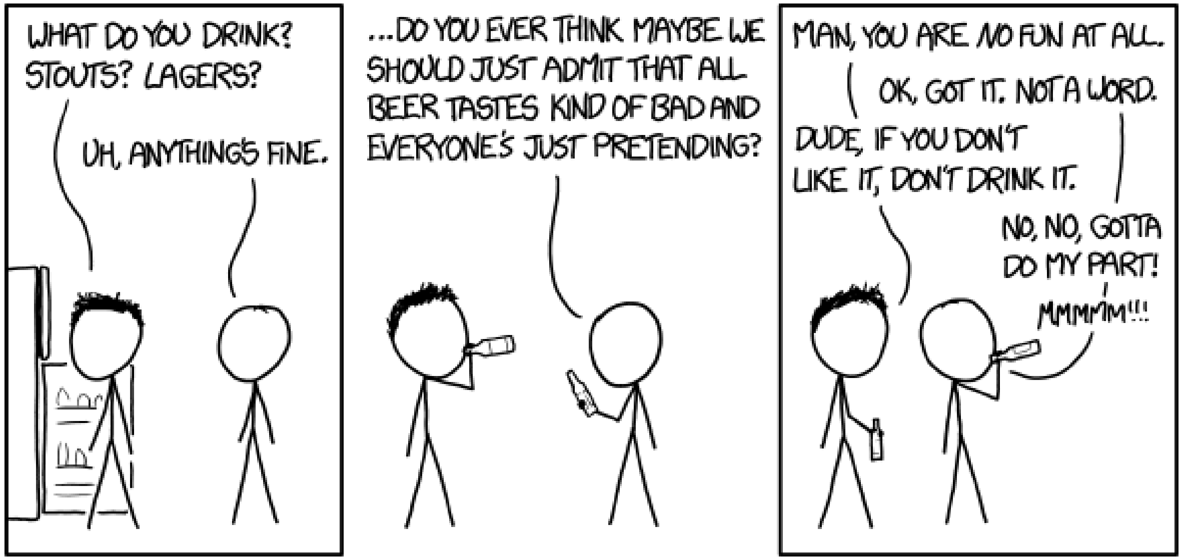
- https://github.com/andrekeller/swinog-29
- Install VirtualBox (https://www.virtualbox.org/)
- Install Vagrant (https://vagrantup.com/)
- Clone the repository (no account needed):
  - `git clone https://github.com/andrekeller/swinog-29.git`
- Start playing:
  - `cd swinog-29/basic-example`
  - `vagrant up`
  - `vagrant ssh`
- I'm happy to answer questions either by direct email or on the SwiNOG mailinglist.

About VSHN

- Swiss DevOps & Ops Company, 12 people in Zürich
- Building the tools and workflows for self-service
- Managing web applications in the cloud
  - We are cloud-agnostic: we run on AWS, MSA, GCE, DO, Hetzner, OVH, SafeSwissCloud, Cloudscale, Exascale and in different enterprise-internal private clouds
- We work for Amazee Labs, Liip, Mercedes Benz Switzerland,Migros, SaltCinema, SIX Group, Sherpany, Sobrado, Starticket, Suisa, Taskfleet, zurichopenair.ch, etc
- Maybe we can help you ?

Thank you

- Questions ?
- Beer!


WHAT DO YOU DRINK? STOUTS? LAGERS?
UH, ANYTHING'S FINE.
...DO YOU EVER THINK MAYBE WE SHOULD JUST ADMIT THAT ALL BEER TASTES KIND OF BAD AND EVERYONE'S JUST PRETENDING?
MAN, YOU ARE NO FUN AT ALL.
OK, GOT IT. NOT A WORD.
DUDE, IF YOU DON'T LIKE IT, DON'T DRINK IT.
NO, NO, GOTTA DO MY PART!
MMMMM!!!

https://xkcd.com/1534/

- We're hiring System and Software Engineers @vshn_ch !
- Get in touch with @andrekeller_ch, @aarnoaukia or @tobruzh

Appendix: GIT

- Create a new repository in your githosting, f.e. github.com
- Create a working directory and add some content:
  - `mkdir ~/myproject`
  - `touch ~/myproject/README`
- Initialize git repository and add content to it:
  - `git init`
  - `git add README && git commit -m 'initial'`
- Push to githosting:
  - `git remote add origin git@github.com:user/repo.git`
  - `git push -u origin master`

Appendix: GIT

- Clone existing repository into local directory:
  - `git clone git@github.com:user/repository.git`
- Commit changes:
  - `git add myfile && git commit -m 'my change' myfile`
- Push changes to remote repository:
  - `git push origin master`
- Update from remote repository
  - `git pull (or git fetch && git rebase -p)`