provis Documentation

Release 0.1.0

Wes Turner

July 25, 2014

Contents

CHAPTER 1

README

Docs | Github | Issues

CHAPTER 2

Provis

Infrastructure Provisioning Scripts and Configuration Sets

A Python package with a few tests, system bootstrap scripts, and a Makefile for building and configuring cloud servers with a number of helpful tools:

- Ubuntu Linux
- Packer (Go)
- Vagrant (Ruby)
- Salt (Python)

Contents

CHAPTER 3

Installation

- **'Install build requirements'_**
- **'Install the Provis Package'_**

Clone and install the package from source:

```
pip install -e ssh://git@github.com/westurner/provis#egg=provis
```

Or, clone the repository and manually install:

```
## clone
git clone ssh://git@github.com/westurner/provis
git clone https://github.com/westurner/provis

cd ./provis

## install
python setup.py develop  # creates a provis.egg-link in site-packages
python setup.py install  # copies the binary dist to site-packages
```

Install Python requirements:

```
cd ./provis
pip install -r requirements.txt
```

Install make, build requirements:

apt-get install make python pip

- Install **'make'_**:

  ```
  apt-get install make
  ```

- Install **'pip'_**:

  ```
  apt-get install pip
  pip install --upgrade pip
  ```

- Install virtualenv and virtualenvwrapper (optional):

  ```
  pip install virtualenv virtualenvwrapper
  ```

  Create a virtualenv with virtualenvwrapper (optional):

  ```
  mkvirtualenv provis
  workon provis
  ```

CHAPTER 4

Usage

Run the Provis Python package tests with the current environment:

```
## Check localhost
python runtests.py  # python setup.py test

## Check reference set
python runtests.py tests/provis_tests.py
```

Run the Provis Python package tests with tox and many environments:

```
tox
```

Install `make` (if it is not already installed):

```
sudo apt-get install make
```

List `Makefile` command descriptions:

```
make help
make

#cd ./provis
ls -al

$EDITOR Makefile
```

Run `make` with the **'Makefile'_**:

```
cd ./provis
make
make help
make setup
```

CHAPTER 5

License

- Free software: BSD license

Contents:

5.1 README

Docs | Github | Issues

5.1.1 Provis

Infrastructure Provisioning Scripts and Configuration Sets

A Python package with a few tests, system bootstrap scripts, and a Makefile for building and configuring cloud servers with a number of helpful tools:

- Ubuntu Linux
- Packer (Go)
- Vagrant (Ruby)
- Salt (Python)

Contents

5.1.2 Installation

- Install build requirements
- Install the Provis Package

Install the Provis Package

Clone and install the package from source:

```
pip install -e ssh://git@github.com/westurner/provis#egg=provis
```

Or, clone the repository and manually install:

```
## clone
git clone ssh://git@github.com/westurner/provis
git clone https://github.com/westurner/provis

cd ./provis

## install
python setup.py develop  # creates a provis.egg-link in site-packages
python setup.py install  # copies the binary dist to site-packages
```

Install Python requirements:

```
cd ./provis
pip install -r requirements.txt
```

Install Build Requirements

Ubuntu 12.04 LTS

Install make, build requirements:

apt-get install make python pip

- Install **'make'_**:

  ```
  apt-get install make
  ```

- Install **'pip'_**:

  ```
  apt-get install pip
  pip install --upgrade pip
  ```

- Install virtualenv and virtualenvwrapper (optional):

  ```
  pip install virtualenv virtualenvwrapper
  ```

  Create a virtualenv with virtualenvwrapper (optional):

  ```
  mkvirtualenv provis
  workon provis
  ```

5.1.3 Usage

Tests

Run the Provis Python package tests with the current environment:

```
## Check localhost
python runtests.py  # python setup.py test

## Check reference set
python runtests.py tests/provis_tests.py
```

Run the Provis Python package tests with tox and many environments:

```
tox
```

Makefile

Install `make` (if it is not already installed):

```
sudo apt-get install make
```

List `Makefile` command descriptions:

```
make help
make

#cd ./provis
ls -al

$EDITOR Makefile
```

Run `make` with the Makefile:

```
cd ./provis
make
make help
make setup
```

5.1.4 License

- Free software: BSD license

5.2 Goals

- Generate new servers as easily as possible
- Generate new servers from infrastructure as code
- Maintain flexibility across multiple virtualization and cloud service providers

5.3 Products

5.3.1 Provis Package

Source: https://github.com/westurner/provis
Documentation: https://provis.readthedocs.org/

`README.rst` – ref:*Provis README Documentation <readme>*
`setup.py` – Python packaging
`Makefile` – Provis Makefile
`docs/` – Provis Documentation
`tox.ini` – tox build configuration
`scripts/` – host bootstrap scripts
`packer/TODOTOTODO` – Packer Configuration JSON
`packer/scripts/` – packer image bootstrap scripts
`vagrant/Vagrantfile` – vagrant configuration
`salt/` – **'salt configuration set'_**
`pillar/` – salt pillar configset
`tests/` – testing utilities
`runtests.py` – test runner

5.3.2 Provis Documentation

https://github.com/westurner/provis/tree/master/docs

System documentation.

- https://provis.readthedocs.org/en/latest/ – ReadTheDocs hosted version
  - Manually update ReadTheDocs for the time being.

5.3.3 Provis Makefile

https://github.com/westurner/provis/blob/master/Makefile

Make *Makefile* with make tasks to bootstrap a host BLD:

- Vagrant
- Packer
- Docker
- VirtualBox
- Downloading ISOs

5.3.4 Provis Tests

https://github.com/westurner/provis/blob/master/tox.ini https://github.com/westurner/provis/tree/master/tests

make test make debugfile

5.3.5 Packer Configuration

Packer build configuration for scripting an Debian/Ubuntu OS install from an ISO in order to create a virtual image.

5.3.6 Vagrant Basebox Image

Virtual image created with Packer, Vagrant, and VirtualBox from which Vagrant VirtualBox instances can be provisioned.

5.3.7 Vagrant Configuration

Vagrantfile to launch image instances and provision salt

5.3.8 Provis Stack

Host-specific catalog of files, packages, and services.

Salt Formulas

Reusable *Salt* state packages.

Salt Modules

Python functions of the Salt API which are remotely-callable

Salt States

Salt policies

Salt Environment

An

- /etc/salt/minion.conf
- /etc/salt/master.conf
- /srv/salt/*top.sls*

Salt states and modules.

Basebox Configset

```
# currently accomplished by a sequence of shell scripts
# launched by the packer virtualbox-iso provisioner
# in
#$ ls ./packer/scripts/
#$ ls ./scripts/bootstrap-salt.sh

users:
 - root/vagrant
 - vagrant/vagrant
   + insecure SSH key
   +

 - ubuntu/ubuntu

sudo:
  passwordless sudo for vagrant user
  no 'requiretty'
etckeeper:
```

Salt Minion Configset

```
configmaster

policies

data
```

Salt Master Configset

```
configmaster

policies

data
```

Gateway/Router Configset

```
networking:
 ip_forward: True
firewall:
 specific ports
dns:
 local dns
 passthrough dns
vpn:
 remote access
```

MySQL Configset

```
mysql
```

Postgres Configset

```
postgres
```

Appserver Configset

```
nginx
build-essentials?
gunicorn
supervisord
upstart
```

Devserver Configset

```
#TODO
```

Workstation Configset

```
TODO: list installed packages (transitive reduction)
i3wm
docker
dotfiles
apt-cacher-ng
nginx
```

5.3.9 Deployment Workflow

- Create and configure an image locally
- Push to cloud
- Paste together

5.3.10 Instrumentation Plan

Instrumentation and scaling are primary concerns that should be kept in mind while developing these templated configurations and application infrastructure topologies.

5.3.11 Collaboration Plan

See *Contributing*

5.4 Activities

Somewhere between usage, process checklists, and *Provis Makefile* documentation.

5.4.1 Project

Create a new repository

See *Contributing*.

- [x] Create a new git repository
  - [x] git init
- [x] Fork an existing repository
  - [x] git clone https://path/to/gitrepo
  - [x] git checkout -b feature_branch_name

Create a new package

- [x] Create templated *Python* package
  - [x] pip install cookiecutter
  - [x] `cookiecutter` https://github.com/audreyr/cookiecutter-pypackage.git
  - [x] README.rst
  - [x] AUTHORS.rst – Authors / Contributors
  - [x] CONTRIBUTING.rst –

Create project documentation

- [x] Create *Sphinx* documentation set
  - [x] docs/conf.py – *Sphinx* build configuration
  - [x] docs/Makefile – *Sphinx Make* file
  - [x] docs/index.rst – *Sphinx ReStructuredText* Index
  - [x] docs/readme.rst – *Sphinx* README
  - [x] docs/authors.rst – Credits
  - [x] docs/tools.rst – Tools Catalog Notes (homepage, documentation, source)
  - [x] docs/installation.rst – Installation Procedure
  - [x] docs/usage.rst – Usage (Makefile)
- [x] Create Seven Layer Model pages
  - [x] docs/goals.rst
  - [x] docs/products.rst
  - [x] docs/activities.rst
  - [x] docs/patterns.rst
  - [x] docs/techniques.rst
  - [x] docs/tools.rst
  - [x] docs/scripts.rst

  - [x] Add {...} to docs/index.rst

5.4.2 Usage

Setup host machine

- [x] Install Ubuntu 12.04
- [x] Update and upgrade Ubuntu 12.04
- [x] Download OS netboot ISOs
- [x] Configure OS package mirrors
  - [x] Find a drive with storage
  - [x] Install apt-cacher-ng
    - [x] Cherrypick apt-cacher-ng binary from trusty (HTTPS support for docker apt repos)
    - [ ] dpkg-divert /usr/sbin/apt-cacher-ng
- [x] Install tools (`make setup_tools`)
  - [x] Review Tool Homepage, Source, Docs (docs/tools.rst)
  - [x] Script installation (`scripts/install_tools.sh`)
    - [x] Install Python, Ruby, Go, Git, Wget:

      ```
      apt-get install python ruby golang-go git wget
      ```

    - [x] Install *Vagrant*
    - [x] Install *Packer*
    - [x] Install *Docker*
    - [x] Install *VirtualBox*

Configure networking and DNS

- [ ] /etc/network/interfaces (salt)
  - [ ] Ethernet interfaces
  - [ ] DHCP IP addresses
  - [ ] Static IP addresses
  - [ ] IP routes
  - [ ] IP tunneling
- [ ] /etc/network/interfaces.d (TODO)
- [ ] Configure DNS
  - [ ] /etc/host.conf (salt)
  - [ ] /etc/hosts Hosts file (salt)
  - [ ] /etc/hostname (salt)
  - [ ] /etc/resolv.conf (resolvconf, salt)

  - [ ] /etc/resolvconf/interface-order
  - [ ] /etc/resolfconf/{base, head, tail}

Create virtual image

- [ ] Create new [VirtualBox]/[Vagrant] basebox with Packer
  - [ ] Script image build with Packer and `packer/scripts`.
    * [x] setup shell scripts
    * [x] vagrant (SSH errors)
    * [x] vagrant FS errors
    * [x] VirtualBox guest tools image (~NTP)
    * [x] etckeeper
    * [x] ufw
    * [x] Apt.conf (apt proxy copied from preseed: apt.create.wrd.nu)

Provision vagrant image instance

- [ ] Create Vagrantfile for launching VirtualBox Vagrant basebox
  - [x] Create a new Vagrantfile: `vagrant init`
  - [x] Configure virtualbox networking support (Vagrant)
    * [x] Rod: eth0 NAT, eth1 Bridged to host eth0, eth2 Host-Only
  - [x] Configure vagrant salt provisioning bootstrap
  - [ ] Configure DNS support
- [x] Launch virtual instance: `vagrant up [<hostname>]`
- [x] Provision with salt: `vagrant provision [<hostname>]`
- [x] Shudown with salt: `vagrant halt [<hostname>]`

Bootstrap salt minion

- [x] Bootstrap salt installation
- [x] Configure salt minion ID
  - [ ] /etc/hosts "salt"
  - [ ] Set minion ID in /etc/salt/minion
  - [ ] Set minion ID in /etc/salt/minion_id:

    ```
    hostname --fqdn | sudo tee /etc/salt/minion_id
    ```

- [ ] Configure for standalone minion setup
  - [ ] Check *file_roots* and *pillar_roots* in /etc/salt/minion
  - [ ] Verify that salt files are in `/srv/salt` and `/srv/pillar`

- [ ] Configure for master/minion setup
  - [ ] DNS resolve 'salt'
  - [ ] Set `master:` in /etc/salt/minion.conf
  - [ ] Pair salt minion/master keys:

    ```
    salt-key --help
    ```

- [x] Run salt
  - [x] Run salt locally as a standalone minion:

    ```
    salt-call --local grains.items
    ```

  - [ ] Run salt from master:

    ```
    salt 'minion_id' grains.items
    ```

  - [ ] Run salt over SSH:

    ```
    salt-ssh 'minion_id' grains.items
    ```

Bootstrap salt master

- [x] Bootstrap salt installation
  - [ ] TODO

Create salt environment

- salt/top.sls
- pillar/top.sls

Create salt formula

- [ ] Create configsets

Test bootstrapped setup

- [ ] Create basic functional network tests
  - [ ] **Python standard library** sockets
  - [*] ICMP
  - [*] TCP Ports
  - [*] TCP Banners
  - [*] HTTP GET 200 OK

5.5 Patterns

5.5.1 Patterns of Collaboration

Generate : space.

Reduce : space.

Clarify : space.

Organize : space.

Evaluate : space.

Build Commitment : space.

5.5.2 Networking Patterns

Needs:

- Gateway-routed topology ('bridged' public / private internal network)
  - Single Point of Ingress/Egress
  - Single Point of Failure
  - Vagrant: multi-machine config with
    - https://docs.vagrantup.com/v2/networking/public_network.html
    - https://docs.vagrantup.com/v2/networking/private_network.html
  - GCE Route Collections
  - EC2 Network ACLs
  - RackSpace Cloud Networks
  - OpenStack Neutron
- Attach an additional NIC
  - Packer: virtualbox-iso provider (VBoxManager)
  - Vagrant: Vagrantfile (VBoxManager)
- DNS dependence

Wants:

- VLANs (OpenStack Neutron)

5.5.3 Storage Patterns

Image Storage

- Packer has builders for various clouds and virtualization solutions (GCE)
- VirtualBox: local filesystem: VDI, VMDK
- Vagrant 'boxes'
- EC2 AMI

- GCE Images
- Docker Images / Registries
- OpenStack Glance Images

Block Storage

- VirtualBox supports (elastic) VDI and VMDK files
- Latest Docker can use BTRFS
- GCE Compute Engine Disks
- EC2 EBS Elastic Block Store
- RackSpace Block Storage
- OpenStack Cinder (RBD, Gluster, Nexenta, NFS)

Object Storage

- GCE Cloud Storage
- AWS S3
- OpenStack Swift
- RackSpace Cloud Files

5.5.4 Remote Filesystems

While remote filesystem access is mostly the wrong pattern for production, for development, it's nice to be able to work in local GVim with synchronous reads and writes on a networked filesystem; though, arguably, the correct deployment pattern is a commit/push CI hook.

- SSHFS is less than consistent; even when synchronized.
- NFS is fairly standard and supports labeling
  - Vagrant has NFS access tools
  - NFS requires at least three ports

5.5.5 Operating Systems

Ubuntu LTS 12.04 is widely implemented.

Ubuntu LTS 14.04 is just out with strong support for OpenStack.

Debian is Debian.

CentOS is more closely tracking RHEL than ever before.

CoreOS and etcd are designed to scale.

5.5.6 Configuration

TODO: fstat misses or explicit conditionals?:

```
1. source:
   1. salt://ntp/local_server/ntp.{{ grains['os'] }}.{{ grains['osrelease'] }}.conf
   2. salt://ntp/local_server/ntp.{{ grains['os'] }}.{{ grains['osrelease'][0] }}.conf
   3. salt://ntp/local_server/ntp.{{ grains['os'] }}
   4. salt://ntp/local_server/ntp.{{ grains['os_family'] }}.conf
   5. salt://ntp/local_server/ntp.{{ grains['kernel'] }}.conf
   6. salt://ntp/local_server/ntp.conf
```

5.6 Techniques

- DRY: Don't Repeat Yourself
- Save that link and put in into context
- In support of reproducible science.
  - "Ten Simple Rules for Reproducible Computational Research"

5.7 Tools

Tools for bootstrapping, installing, and managing change in systems.

Objective: Maximize Tool Value (Output / Input)

5.7.1 Distro Packages

Operating Systems Packaging

Source and/or binary packages to install from a standard archive with a *signed* manifest containing file signatures of package files.

RPM Package

https://en.wikipedia.org/wiki/RPM_Package_Manager

- Installable with yum, {...}
- Build with TODO: rpmbuild
- Python: build with bdist_rpm, {...}
- List contents:

  ```
  # with lesspipe
  less ~/path/to/local.rpm
  ```

- Package Repositories (yum):
  - Local: directories of packages and metadata
  - Network: HTTP, HTTPS, RSYNC, FTP

DEB Package

https://en.wikipedia.org/wiki/Deb_(file_format)

- Installable with apt-get, aptitutde,
- Build with dpkg
- List contents:

```
# with lesspipe
less ~/path/to/local.deb
```

- Package Repositories (apt):
  - Local: directories of packages and metadata
  - Network: HTTP, HTTPS, RSYNC, FTP (apt transports)
- Linux/Mac/Windows: Yes / Fink / No

Homebrew

https://en.wikipedia.org/wiki/Homebrew_(package_management_software)

- Linux/Mac/Windows: No / Yes / No
- Package Recipe Repositories (brew):
  - Local:
  - Network: HTTP, HTTPS

NuGet

https://en.wikipedia.org/wiki/NuGet

- Package Repositories (chocolatey):
  - https://chocolatey.org/
- Linux/Mac/Windows: No / No / Yes

Portage

https://en.wikipedia.org/wiki/Portage_(software)

- Build recipes with flag sets
- Package Repositories (portage)

Port Tree

Sources and Makefiles designed to compile software packages for particular distributions' kernel and standard libraries on a particular platform.

CoreOS Docker Images

CoreOS schedules redundant docker images and configuration over etcd, a key-value store with a D-Bus interface.

- Create high availability zone clusters with fleet
- Systemd init files

5.7.2 Apt

Wikipedia: https://en.wikipedia.org/wiki/Advanced_Packaging_Tool
Homepage: http://alioth.debian.org/projects/apt
Docs: https://wiki.debian.org/Apt
Docs: https://www.debian.org/doc/manuals/debian-reference/ch02.en.html
Docs: https://www.debian.org/doc/manuals/apt-howto/
Source: git git://anonscm.debian.org/git/apt/apt.git
IRC: irc://irc.debian.org/debian-apt

APT is the Debian package management system.

APT retrieves packages over FTP, HTTP, HTTPS, and RSYNC.

```
man apt-get
man sources.list
echo 'deb repo_URL distribution component1' >> /etc/apt/sources.list
apt-get update
apt-cache show bash
apt-get install bash
apt-get upgrade
apt-get dist-upgrade
```

5.7.3 Bash

Wikipedia: https://en.wikipedia.org/wiki/Bash_(Unix_shell)
Homepage: http://www.gnu.org/software/bash/
Docs: https://www.gnu.org/software/bash/manual/
Source: git git://git.savannah.gnu.org/bash.git

Bash, the Bourne-again shell.

```
type bash
bash --help
help help
help type
apropos bash
info bash
man bash
```

- Designed to work with unix command outputs and return codes

- Functions
- Portability: sh (sh, bash, dash, zsh) shell scripts are mostly compatible
- Logging:

```
set -x  # print commands and arguments
set -v  # print source
```

Bash Configuration:

```
/etc/profile
/etc/bash.bashrc
/etc/profile.d/*.sh
${HOME}/.profile        /etc/skel/.profile   # PATH=+$HOME/bin  # umask
${HOME}/.bash_profile   # empty. preempts .profile
```

Linux/Mac/Windows: Almost Always / Bash 3.2 / Cygwin/Mingwin

5.7.4 Dpkg

Wikipedia: https://en.wikipedia.org/wiki/Dpkg
Homepage: http://wiki.debian.org/Teams/Dpkg
Docs: https://en.wikipedia.org/wiki/Debian_build_toolchain
Docs: https://en.wikipedia.org/wiki/Deb_(file_format)

Lower-level package management scripts for creating and working with .DEB Debian packages.

5.7.5 Docker

Wikipedia: https://en.wikipedia.org/wiki/Docker_(software)
Homepage: https://docker.io/
Docs: http://docs.docker.io/
Source: https://github.com/dotcloud/docker

Docker is an OS virtualization project which utilizes Linux LXC Containers to partition process workloads all running under one kernel.

Limitations

- Writing to */etc/hosts*: https://github.com/dotcloud/docker/issues/2267
- Apt-get upgrade: https://github.com/dotcloud/docker/issues/3934

5.7.6 Docutils

Homepage: http://docutils.sourceforge.net
Docs: http://docutils.sourceforge.net/docs/
Docs: http://docutils.sourceforge.net/rst.html
Docs: http://docutils.sourceforge.net/docs/ref/doctree.html

Source: svn http://svn.code.sf.net/p/docutils/code/trunk

Docutils is a text processing system which 'parses" *ReStructuredText* lightweight markup language into a doctree which it serializes into HTML, LaTeX, man-pages, Open Document files, XML, and a number of other formats.

5.7.7 Filesystem Hierarchy Standard

Wikipedia: https://en.wikipedia.org/wiki/Filesystem_Hierarchy_Standard
Website: http://www.linuxfoundation.org/collaborate/workgroups/lsb/fhs

The Filesystem Hierarchy Standard is a well-worn industry-supported system file naming structure.

Ubuntu and *Virtualenv* implement a Filesystem Hierarchy.

Docker layers filesystem hierarchies with aufs and now also btrfs subvolumes.

5.7.8 Git

Wikipedia: https://en.wikipedia.org/wiki/Git_(software)
Homepage: http://git-scm.com/
Docs: http://git-scm.com/documentation
Docs: http://documentup.com/skwp/git-workflows-book
Source: git https://github.com/git/git

Git is a distributed version control system for tracking a branching and merging repository of file revisions.

5.7.9 Go

Wikipedia: https://en.wikipedia.org/wiki/Go_(programming_language)
Homepage: http://golang.org/
Docs: http://golang.org/doc/
Source: hg https://code.google.com/p/go/

Go is a relatively new statically-typed C-based language.

5.7.10 Json

Wikipedia: https://en.wikipedia.org/wiki/JSON
Homepage: http://json.org/

Parse and indent JSON with *Python* and *Bash*:

```
cat example.json | python -m json.tool
```

5.7.11 Libcloud

Homepage: https://libcloud.apache.org/
Docs: https://libcloud.readthedocs.org/
Docs: https://libcloud.readthedocs.org/en/latest/supported_providers.html
Source: git git://git.apache.org/libcloud.git
Source: git https://github.com/apache/libcloud

Apache Libcloud is a *Python* library which abstracts and unifies a large number of Cloud APIs for Compute Resources, Object Storage, Load Balancing, and DNS.

5.7.12 Libvirt

Wikipedia: http://libvirt.org/
Homepage: http://libvirt.org/
Docs: http://libvirt.org/docs.html
Docs: http://docs.saltstack.com/en/latest/ref/modules/all/salt.modules.virt.html
Source: git git://libvirt.org/libvirt-appdev-guide.git

Libvirt is a system for platform virtualization with various *Linux* hypervisors.

- KVM/QEMU
- Xen
- LXC
- OpenVZ
- VirtualBox

5.7.13 Linux

Wikipedia: https://en.wikipedia.org/wiki/Linux
Homepage: https://www.kernel.org
Docs: https://www.kernel.org/doc/
Source: git https://github.com/torvalds/linux

A free and open source operating system kernel written in C.

```
uname -a
```

5.7.14 Make

Wikipedia: https://en.wikipedia.org/wiki/Make_(software)
Homepage: https://www.gnu.org/software/make/
Project: https://savannah.gnu.org/projects/make/
Docs: https://www.gnu.org/software/make/manual/make.html
Source: git git://git.savannah.gnu.org/make.git

GNU Make is a classic, ubiquitous software build tool designed for file-based source code compilation.

Bash, *Python*, and the GNU/*Linux* kernel are all built with Make.

Make build task chains are represented in a *Makefile*.

Pros

- Simple, easy to read syntax
- Designed to build files on disk
- Nesting: `make -C <path> <taskname>`
- Variable Syntax: `$(VARIABLE_NAME)`
- Bash completion: `make <tab>`
- Python: Parseable with disutils.text_file Text File
- Logging: command names and values to stdout

Cons

- Platform Portability: make is not installed everywhere
- Global Variables: Parametrization with shell scripts
- Linux/Mac/Windows: Usually / brew / executable

5.7.15 MessagePack

Wikipedia: https://en.wikipedia.org/wiki/MessagePack
Homepage: http://msgpack.org/

MessagePack is a data interchange format with implementations in many languages.

Salt

5.7.16 Packer

Homepage: http://www.packer.io/
Docs: http://www.packer.io/docs
Docs: http://www.packer.io/docs/basics/terminology.html
Source: git https://github.com/mitchellh/packer

Packer generates machine images for multiple platforms, clouds, and hypervisors from a parameterizable template.

Packer Artifact Build products: machine image and manifest

Packer Template JSON build definitions with optional variables and templating

Packer Build A task defined by a JSON file containing build steps which produce a machine image

Packer Builder Packer components which produce machine images for one of many platforms:

- VirtualBox
- Docker
- OpenStack
- GCE
- EC2
- VMware
- QEMU (KVM, Xen)
- http://www.packer.io/docs/templates/builders.html

Packer Provisioner Packer components for provisioning machine images at build time

- Shell scripts
- File uploads
- ansible
- chef
- solo
- puppet
- salt

Packer Post-Processor Packer components for compressing and uploading built machine images

5.7.17 Perl

Wikipedia: https://en.wikipedia.org/wiki/Perl
Homepage: http://www.perl.org/
Project: http://dev.perl.org/perl5/
Docs: http://www.perl.org/docs.html
Source: git git://perl5.git.perl.org/perl.git

Perl is a dynamically typed, C-based scripting language.

Many of the Debian system management tools are or were originally written in Perl.

5.7.18 Python

Wikipedia: https://en.wikipedia.org/wiki/Python_(programming_language)
Homepage: https://www.python.org/

Docs: https://docs.python.org/2/
Source: hg http://hg.python.org/cpython

Python is a dynamically-typed, C-based scripting language.

Many of the RedHat system management tools are or were originally written in Python.

Pip, *Sphinx*, *Salt*, *Tox*, *Virtualenv*, and *Virtualenvwrapper* are all written in Python.

5.7.19 Python Package

Archive of source and/or binary files containing a setup.py.

A setup.py calls a `distutils.setup` or `setuptools.setup` function with package metadata fields like name, version, maintainer name, maintainer email, and home page; as well as package requirements: lists of package names and version specifiers in `install_requires` and `tests_require`, and a dict for any `extras_require` such that '`easy_install setup.py`, `python setup.py install`, and `pip install --upgrade pip` can all retrieve versions of packages which it depends on.

- Distutils is in the Python standard library
- Setuptools is widely implemented: `easy_install`
- Setuptools can be installed with `python ez_setup.py`
- Setuptools can be installed with a system package manager (apt, yum)
- Python packages are tested and repackaged by package maintainers
- Python packages are served from a package index
- PyPi is the Python Community package home
- Packages are released to PyPi
- Package Repositories (setup.py -> pypi)
- Package Repositories (conda)
- Package Repositories (enpkg)
- Package Repositories (deb/apt, rpm/yum)
- Build RPM and DEB packages from Python packages with setuptools
  - `python setup.py bdist_rpm --help`
  - `python setup.py --command-packages=stdeb.command bdist_deb --help`

5.7.20 Pip

Wikipedia: https://en.wikipedia.org/wiki/Pip_(package_manager)
Homepage: http://www.pip-installer.org/
Docs: http://www.pip-installer.org/en/latest/user_guide.html
Docs: https://pip.readthedocs.org/en/latest/
Docs: http://packaging.python.org/en/latest/
Source: git https://github.com/pypa/pip
Pypi: https://pypi.python.org/pypi/pip

IRC: #pypa
IRC: #pypa-dev

Pip is a tool for working with *Python* packages.

```
pip help
pip help install
pip --version

sudo apt-get install python-pip
pip install --upgrade pip

pip install libcloud
pip install -r requirements.txt
pip uninstall libcloud
```

- Pip retrieves and installs packages from package indexes
- Pip can do uninstall and upgrade
- Pip builds upon distutils and setuptools
- Pip can install from version control repository URLs
- Pip configuration is in `${HOME}/.pip/pip.conf`.
- Pip can maintain a local cache of downloaded packages

Note: With *Python* 2, pip is preferable to `easy_install` because Pip installs `backports.ssl_match_hostname`.

Pip Requirements File Plaintext list of packages and package URIs to install.

Requirements files may contain version specifiers (`pip >= 1.5`)

Pip installs Pip Requirement Files:

```
pip install -r requirements.txt
pip install --upgrade -r requirements.txt
pip install --upgrade --user --force-reinstall -r requirements.txt
```

An example `requirements.txt` file:

```
# install pip from the default index (PyPi)
pip
--index=https://pypi.python.org/simple --upgrade pip

# Install pip 1.5 or greater from PyPi
pip >= 1.5

# Git clone and install pip as an editable develop egg
-e git+https://github.com/pypa/pip@1.5.X#egg=pip

# Install a source distribution release from PyPi
# and check the MD5 checksum in the URL
https://pypi.python.org/packages/source/p/pip/pip-1.5.5.tar.gz#md5=7520581ba0687dec1ce85bd154965

# Install a source distribution release from Warehouse
https://warehouse.python.org/packages/source/p/pip/pip-1.5.5.tar.gz
```

```
# Install an additional requirements.txt file
-r requirements/more-requirements.txt
```

5.7.21 ReStructuredText

Wikipedia: https://en.wikipedia.org/wiki/ReStructuredText
Homepage: http://docutils.sourceforge.net/rst.html
Docs: http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html
Docs: http://docutils.sourceforge.net/docs/ref/rst/directives.html
Docs: http://docutils.sourceforge.net/docs/ref/rst/roles.html
Docs: http://sphinx-doc.org/rest.html

ReStructuredText (RST, ReST) is a plaintext lightweight markup language commonly used for narrative documentation and Python docstrings.

Sphinx is built on *Docutils*, which is the primary implementation of ReStructuredText.

Pandoc also supports a form of ReStructuredText.

ReStructuredText Directive Actionable blocks of ReStructuredText

```
.. include:: goals.rst

.. contents:: Table of Contents
   :depth: 3

.. include:: LICENSE
```

ReStructuredText Role RestructuredText role extensions

```
.. _anchor-name:

:ref:`Anchor <anchor-name>`
```

5.7.22 Salt

Wikipedia: https://en.wikipedia.org/wiki/Salt_(software)
Homepage: http://www.saltstack.com
Docs: http://docs.saltstack.com/en/latest/
Docs: http://salt.readthedocs.org/en/latest/ref/clients/index.html#python-api
Docs: http://docs.saltstack.com/en/latest/topics/development/hacking.html
Glossary: http://docs.saltstack.com/en/latest/glossary.html
Source: git https://github.com/saltstack/salt
Pypi: https://pypi.python.org/pypi/salt
IRC: #salt

Salt is an open source configuration management system for managing one or more physical and virtual machines running various operating systems.

Salt Top File Root of a Salt Environment (`top.sls`)

Salt Environment Folder of Salt States with a top.sls top file.

Salt Bootstrap Installer for salt master and/or salt minion

Salt Minion Daemon process which executes Salt States on the local machine.

Can run as a background daemon. Can retrieve and execute states from a salt master

Can execute local states in a standalone minion setup:

```
salt-call --local grains.items
```

Salt Minion ID Machine ID value uniquely identifying a minion instance to a Salt Master.

By default the minion ID is set to the FQDN

```
python -c 'import socket; print(socket.getfqdn())'
```

The minion ID can be set explicitly in two ways:

- /etc/salt/minion.conf:

  ```
  id: devserver-123.example.org
  ```

- /etc/salt/minion_id:

  ```
  $ hostname -f > /etc/salt/minion_id
  $ cat /etc/salt/minion_id
  devserver-123.example.org
  ```

Salt Master Server daemon which compiles pillar data for and executes commands on Salt Minions:

```
salt '*' grains.items
```

Salt SSH Execute salt commands and states over SSH without a minion process:

```
salt-ssh '*' grains.items
```

Salt Grains Static system information keys and values

- hostname
- operating system
- ip address
- interfaces

Show grains on the local system:

```
salt-call --local grains.items
```

Salt Modules Remote execution functions for files, packages, services, commands.

Can be called with salt-call

Salt States Graphs of nodes and attributes which are templated and compiled into ordered sequences of system configuration steps.

Naturally stored in `.sls` *YAML* files parsed by `salt.states.<state>.py`.

Salt States files are processed as Jinja templates (by default) they can access system-specific grains and pillar data at compile time.

Salt Renderers Templating engines (by default: Jinja) for processing templated states and configuration files.

Salt Pillar Key Value data interface for storing and making available global and host-specific values for minions: values like hostnames, usernames, and keys.

Pillar configuration must be kept separate from states (e.g. users, keys) but works the same way.

In a master/minion configuration, minions do not have access to the whole pillar.

Salt Cloud Salt Cloud can provision cloud image, instance, and networking services with various cloud providers (libcloud):

- Google Compute Engine (GCE) [KVM]
- Amazon EC2 [Xen]
- Rackspace Cloud [KVM]
- OpenStack [https://wiki.openstack.org/wiki/HypervisorSupportMatrix]
- Linux LXC (Cgroups)
- KVM

5.7.23 Sphinx

Wikipedia: https://en.wikipedia.org/wiki/Sphinx_(documentation_generator)
Homepage: https://pypi.python.org/pypi/Sphinx
Docs: http://sphinx-doc.org/contents.html
Docs: http://sphinx-doc.org/markup/code.html
Docs: http://pygments.org/docs/lexers/
Docs: http://thomas-cokelaer.info/tutorials/sphinx/rest_syntax.html
Source: hg https://bitbucket.org/birkenfeld/sphinx/
Pypi: https://pypi.python.org/pypi/Sphinx

Sphinx is a tool for working with *ReStructuredText* documentation trees and rendering them into HTML, PDF, LaTeX, ePub, and a number of other formats.

Sphinx extends *Docutils* with a number of useful markup behaviors which are not supported by other ReStructuredText parsers.

Most other ReStructuredText parsers do not support Sphinx directives; so, for example,

- GitHub and BitBucket do not support Sphinx but do support ReStructuredText so README.rst containing Sphinx tags renders in plaintext or raises errors.

  For example, the index page of this *Sphinx* documentation set is generated from a file named `index.rst` and referenced by `docs/conf.py`.

  - Input: https://raw.githubusercontent.com/westurner/provis/master/docs/index.rst
  - Output: https://github.com/westurner/provis/blob/master/docs/index.rst
  - Output: *ReadTheDocs* http://provis.readthedocs.org/en/latest/

Sphinx Builder Render Sphinx *ReStructuredText* into various forms:

- HTML
- LaTeX
- PDF

- ePub

See: Sphinx Builders

Sphinx ReStructuredText Sphinx extends *ReStructuredText* with roles and directives which only work with Sphinx.

Sphinx Directive Sphinx extensions of *Docutils ReStructuredText* directives.

Most other ReStructuredText parsers do not support Sphinx directives.

```
.. toctree::

   readme
   installation
   usage
```

See: Sphinx Directives

Sphinx Role Sphinx extensions of *Docutils ReStructuredText* roles

Most other ReStructured

```
.. _anchor-name:

:ref:`Anchor <anchor-name>`
```

5.7.24 Ruby

Wikipedia: https://en.wikipedia.org/wiki/Ruby_(programming_language)
Homepage: https://www.ruby-lang.org/
Docs: https://www.ruby-lang.org/en/documentation/
Source: svn http://svn.ruby-lang.org/repos/ruby/trunk

Ruby is a dynamically-typed programming language.

Vagrant is written in Ruby.

5.7.25 Tox

Homepage: https://testrun.org/tox/
Docs: https://tox.readthedocs.org
Source: hg https://bitbucket.org/hpk42/tox
Pypi: https://pypi.python.org/pypi/tox

Tox is a build automation tool designed to build and test Python projects with multiple language versions and environments in separate *virtualenvs*.

Run the py27 environment:

```
tox -v -e py27
tox --help
```

5.7.26 Ubuntu

Wikipedia: https://en.wikipedia.org/wiki/Ubuntu_(operating_system)
Homepage: http://www.ubuntu.com/
Docs: https://help.ubuntu.com/
Source: https://launchpad.net/ubuntu
Source: http://archive.ubuntu.com/
Source: http://releases.ubuntu.com/

5.7.27 Vagrant

Wikipedia: https://en.wikipedia.org/wiki/Vagrant_(software)
Homepage: http://www.vagrantup.com/
Docs: http://docs.vagrantup.com/v2/
Source: git https://github.com/mitchellh/vagrant

Vagrant is a tool for creating and managing virtual machine instances with CPU, RAM, Storage, and Networking.

- Vagrant:
  - provides helpful commandline porcelain on top of *VirtualBox* `VboxManage`
  -

```
vagrant help
vagrant status
vagrant init ubuntu/trusty64
vagrant up
vagrant ssh
$EDITOR Vagrantfile
vagrant provision
vagrant halt
vagrant destroy
```

Vagrantfile Vagrant script defining a team of one or more virtual machines and networks.

Create a Vagrantfile:

```
vagrant init [basebox]
cat Vagrantfile
```

Start virtual machines and networks defined in the Vagrantfile:

```
vagrant status
vagrant up
```

Vagrant Box Vagrant base machine virtual machine image.

There are many baseboxes for various operating systems.

Essentially a virtual disk plus CPU, RAM, Storage, and Networking metadata.

Locally-stored and cached vagrant boxes can be listed with:

```
vagrant help box
vagrant box list
```

A running vagrant environment can be packaged into a new box with:

```
vagrant package
```

Packer generates *VirtualBox* Vagrant Boxes with a Post-Processor.

Vagrant Cloud Vagrant-hosted public Vagrant Box storage.

Install a box from Vagrant cloud:

```
vagrant init ubuntu/trusty64
vagrant up
vagrant ssh
```

Vagrant Provider A driver for running Vagrant Boxes with a hypervisor or in a cloud.

The Vagrant *VirtualBox* Provider is well-supported.

With Plugins: https://github.com/mitchellh/vagrant/wiki/Available-Vagrant-Plugins

See also: *Libcloud*.

Vagrant Provisioner Set of hooks to install and run shell scripts and configuration managment tools over `vagrant ssh`.

Vagrant up runs `vagrant provision` on first invocation of `vagrant up`.

```
vagrant provision
```

Note: Vagrant configures a default NFS share mounted at `/vagrant`.

Note: Vagrant adds a default NAT Adapter as eth0; presumably for DNS, the default route, and to ensure `vagrant ssh` connectivity.

5.7.28 VirtualBox

Wikipedia: https://en.wikipedia.org/wiki/VirtualBox
Homepage: https://www.virtualbox.org/
Docs: https://www.virtualbox.org/wiki/Documentation
Source: svn svn://www.virtualbox.org/svn/vbox/trunk

Oracle VirtualBox is a platform virtualization package for running one or more guest VMs (virtual machines) within a host system.

VirtualBox:

- runs on many platforms: *Linux*, OSX, Windows
- has support for full platform NX/AMD-v virtualization
- requires matching kernel modules

Vagrant scripts VirtualBox.

5.7.29 Virtualenv

Homepage: http://www.virtualenv.org
Docs: http://www.virtualenv.org/en/latest/
Source: git https://github.com/pypa/virtualenv
PyPI: https://pypi.python.org/pypi/virtualenv
IRC: #pip

Virtualenv is a tool for creating reproducible *Python* environments.

Virtualenv sets the shell environment variable $VIRTUAL_ENV when active.

Paths within a virtualenv are more-or-less *FSH* standard paths, making virtualenv structure very useful for building chroot and container overlays.

A standard virtual environment:

```
bin/           # pip, easy_install, console_scripts
bin/activate   # source bin/activate to work on a virtualenv
include/       # (symlinks to) dev headers (python-dev/python-devel)
lib/           # libraries
lib/python2.7/site-packages/  # pip and easy_installed packages
local/         # symlinks to bin, include, and lib

src/           # pip installs editable requirements here

# also useful
etc/           # configuration
var/log        # logs
var/run        # sockets, PID files
tmp/           # mkstemp temporary files with permission bits
srv/           # local data
```

Virtualenvwrapper wraps virtualenv. In the following code shell example, comments with ## are virtualenvwrapper

```
# Print Python site settings
python -m site

# Create a virtualenv
cd $WORKON_HOME
virtualenv example
source ./example/bin/activate
## mkvirtualenv example
## workon example

# Review virtualenv Python site settings
python -m site

# List files in site-packages
ls -altr $VIRTUAL_ENV/lib/python*/site-packages/**
## (cdsitepackages && ls -altr **)
## lssitepackages -altr **
```

5.7.30 Virtualenvwrapper

Docs: http://virtualenvwrapper.readthedocs.org/en/latest/
Source: hg https://bitbucket.org/dhellmann/virtualenvwrapper
PyPI: https://pypi.python.org/pypi/virtualenvwrapper

Virtualenvwrapper is a tool which extends virtualenvwrapper.

Virtualenvwrapper provides a number of useful shell commands and python functions for working with and within *virtualenvs*, as well as project event scripts (e.g. `postactivate`, `postmkvirtualenv`) and two filesystem configuration variables useful for structuring development projects of any language within *virtualenvs*: `$PROJECT_HOME` and `$WORKON_HOME`.

Virtualenvwrapper is sourced into the shell:

```
# pip install --user --upgrade virtualenvwrapper
source ~/.local/bin/virtualenvwrapper.sh

# sudo apt-get install virtualenvwrapper
source /etc/bash_completion.d/virtualenvwrapper

echo $PROJECT_HOME; echo ~/wrk        # default: ~/workspace
echo $WORKON_HOME;  echo ~/wrk/.ve    # default: ~/.virtualenvs

mkvirtualenv example
workon example
cdvirtualenv ; ls
mkdir src ; cd src/

cdsitepackages
lssitepackages


deactivate
rmvirtualenv example
```

5.7.31 YAML

Wikipedia: https://en.wikipedia.org/wiki/YAML
Homepage: http://yaml.org

YAML ("YAML Ain't Markup Language") is a concise data serialization format.

Most *Salt* states and pillar data are written in YAML. Here's an example `top.sls` file:

```
base:
 '*':
   - openssh
 '*-webserver':
   - webserver
 '*-workstation':
```

```
- gnome
- i3
```

5.8 Scripts

See: *Makefile*

5.9 Contributing

5.9.1 Goals

- Generate new servers as easily as possible
- Generate new servers from infrastructure as code
- Maintain flexibility across multiple virtualization and cloud service providers

5.9.2 Ways to Contribute

This is a small project.

Contributions are welcome, and they are greatly appreciated! Every little bit helps.

You can contribute in many ways:

- *Report bugs*
- *Add project tags*
- *Fix bugs*
- *Suggest features*
- *Implement features*
- *Write documentation*
- Submit feedback
- *Send a patch or pull request*

5.9.3 Project Tags

Use these uppercase tags in issues and commit messages to help organize commits:

```
FEAT: Feature
BUG: Bug
DOC: Documentation
TST: Test
BLD: Build
PERF: Performance
CLN: Cleanup
SEC: Security
```

Commit Messages:

```
FEAT: Add new feature (closes #3)
FIX: Fixes #3
```

Separate multiple tags with a comma:

```
DOC,BLD,TST: Improve build docs and tests
TST,BUG: Add a test for reproducing a bug
```

5.9.4 FEAT: Feature

Implement Features

Look through the GitHub issues for features. Anything tagged with "FEAT" is open to whoever wants to implement it.

If you are proposing a feature (FEAT):

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

5.9.5 DOC: Documentation

Write Documentation

provis could always use more documentation (DOC), whether as part of the official provis docs, in docstrings, or even on the web in blog posts, articles, and such.

5.9.6 BUG: Bug

Report Bugs

Report bugs at https://github.com/westurner/provis/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "BUG" is open to whoever wants to implement it.

5.9.7 TST: Test

5.9.8 BLD: Build

5.9.9 PERF: Performance

5.9.10 CLN: Cleanup

5.9.11 SEC: Security

Contributing References

- feat, fix, docs, style, refactor, perf, test, chore – AngularJS CONTRIBUTING.md
- ENH, BUG, DOC, TST, BLD, PERF, CLN – Pandas CONTRIBUTING.md
- `{{ cookiecutter.repo_name }}` – Cookiecutter-pypackage CONTRIBUTING.rst

5.9.12 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/westurner/provis/issues.

5.9.13 Get Started!

Ready to contribute? Here's how to set up *provis* for local development.

1. Fork the *provis* repo on GitHub.
2. Clone your fork locally:

   ```
   $ git clone ssh://git@github.com/your_name_here/provis.git
   ```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

   ```
   $ mkvirtualenv provis
   $ cd provis/
   $ python setup.py develop
   ```

4. Create a branch for local development:

   ```
   $ git checkout -b name-of-your-bugfix-or-feature
   ```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

   ```
   $ flake8 provis tests
   $ python setup.py test
   $ tox
   ```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

5.9.14 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests (TST).
2. The pull request could have Project Tags.
3. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
4. The pull request should work for Python 2.6, 2.7, and 3.3, and for PyPy. Check https://travis-ci.org/westurner/provis/pull_requests and make sure that the tests pass for all supported Python versions.

5.9.15 Tips

To run a subset of tests:

```
$ python -m unittest tests.test_provis
```

5.10 API

5.10.1 Provis Module

Functions for testing provisioned infrastructure

5.10.2 Submodules

5.11 Credits

5.11.1 Development Lead

- Wes Turner <wes@wrd.nu>

5.11.2 Contributors

-

5.11.3 Pandas

- https://github.com/pydata/pandas/blob/master/CONTRIBUTING.md

5.11.4 Seven Layer Model of Collaboration

```
Briggs, Robert O., Gwendolyn Kolfschoten, Gert-Jan de Vreede,
Conan Albrecht, Douglas R. Dean, and Stephan Lukosch.
"A seven-layer model of collaboration: Separation of concerns
for designers of collaboration systems." (2009).
```

5.12 History

5.12.1 0.1.0 (Unreleased)

- Created roadmap.rst
- Created repository with cookiecutter and cookiecutter-pypackage
- Created docs/tools.rst
- Created scripts/install_tools.sh
- Created scripts/download_ubuntu_isos.sh
- Created scripts/parse_ubuntu_miniiso_checksums.py
- Created and tested packer JSON and ubuntu 12.04 mini.iso preseed
- Imported packer/scripts from cargomedia/vagrant-boxes
- Adapted, modified, and tested Packer JSON, preseed, and scripts
- Created Vagrantfile
- Created initial salt policies (webserver package, service, and ufw cmd)
- Created provis.net utilities: ICMP, Ports, Banners; socket, sarge, structlog)
- Created vm tests with net utilities
- Switched to py.test w/ pytest-capturelog (setup.py, runtests)
- Updated tox.ini (py.test, style (flake8), docs)
- Updated Makefile
- Imported scripts/salt-bootstrap.sh from salt-bootstrap
- Updated packer JSON: salt-bootstrap.sh from git@develop,
- Updated packer JSON: update tty config in /root/.profile for vagrant provisioner

CHAPTER 6

Indices and tables

- *genindex*
- *modindex*
- *search*

p