title: Configuration Management Solutions For FreeBSD Systems Administrators author: Roller Angel roller@bsd.pw date: 2023-05-17 extensions: [] theme: light styles: style: one-dark

BSDCan 2023

Hello,

My name is Roller Angel.

My first trip to BSDCan was in 2017, Thanks for having me back.

My slideshow is powered by Markdown and the Python project LookAtMe

You can download the slides Markdown file from https://raw.githubusercontent.com/possnfiffer/bsd-pw/gh-pages/bsdcan2023-slides.md (https://raw.githubusercontent.com/possnfiffer/bsd-pw/gh-pages/bsdcan2023-slides.md)

Configuration Management Software

Ansible

What I reach for when I need to create something from scratch * agentless * client only needs SSH * Group Vars * Vault * Playbooks

SaltStack

What I rely on when I have an infrastructure I need to maintain * client typically needs an Agent installed and running * also supports SSH if you're interested * Salt Master * Salt Minion * Pillar * Grains * Highstate

Ansible

What does Ansible configuration look like from a high level?

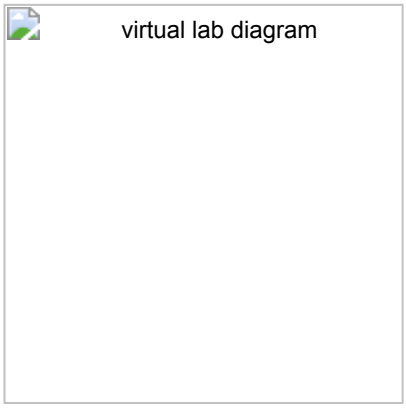
```
├── ansible.cfg
├── group_vars
│   └── all
│       └── config
├── hosts
├── playbook.yml
└── roles
    └── common
        ├── handlers
        │   ├── main.yml
        │   └── ssh_setup.yml
        ├── tasks
        │   ├── bootstrap.yml
        │   ├── main.yml
        │   ├── python_config.yml
        │   └── ssh_config.yml
        └── templates
            └── rc.conf
```

SaltStack

What does SaltStack configuration look like from a high level?

```
├── base
│   ├── etc
│   │   └── freebsd-update.conf
│   ├── init.sls
│   ├── users.sls
│   └── usr
│       └── local
│           └── etc
│               ├── salt
│               │   └── minion.j2
│               └── sudoers
├── top.sls
└── update
    ├── init.sls
    └── usr
        └── local
            └── etc
                └── apache24
                    ├── Includes
                    │   └── update-cache.conf
                    └── httpd.conf
```

Virtual Lab



Link to diagram https://github.com/possnfiffer/bsd-pw/blob/fadf00acca1fde55d83d719855c7dda6d6066585/img/lab_network_diagram.png (https://github.com/possnfiffer/bsd-pw/blob/fadf00acca1fde55d83d719855c7dda6d6066585/img/lab_network_diagram.png)

The diagram was published in the FreeBSD Journal article I wrote called Virtual Lab - BSD Programming workshop https://freebsdfoundation.org/wp-content/uploads/2023/02/angel_virtuallab.pdf (https://freebsdfoundation.org/wp-content/uploads/2023/02/angel_virtuallab.pdf)

I turned the instructions from this article into an Ansible Playbook then improved the playbook to include setting up Salt and a FreeBSD Update cache. Find the link to the tar.gz Ansible Playbook at http://BSD.pw (http://BSD.pw)

Create the Lab

PROBLEM: The virtual lab doesn't exist yet

- How do we automate it's creation?
- Doesn't Ansible require SSH and Salt needs an Agent installed?
- Where do we begin?

SOLUTION: Ansible

When we don't already have infrastructure and want to create it out of thin air, Ansible is perfect.

Using the connection type of `local` we can skip using SSH for Ansible and directly use it to control our localhost.

How do these tools work?

From a high level, these tools use python modules and arguments passed to these modules to execute commands on local and remote systems

Why we should use configuration management software?

Sanity

A great alternative to handwritten notes in various notebooks and scribbles on random pieces of paper scattered throughout your backpack and across your desk

Plus, they automate the typing

We can all do with less typing, right!

Setting up our Virtual Lab with Ansible controlling our lab host

We'll need to install the Ansible package on our local lab host FreeBSD machine.

`py39-ansible`

Then we can run the playbook in virtual-lab

```
ansible-playbook -vv playbook.yml
```

Monitoring salt events

Salt uses events for all sorts of functionality.

To see what salt is doing, you can issue the following command in a separate terminal window and every event will appear as it happens. You can look into salt reactors and salt can be configured to react to events and do something like apply a state or trigger a script.

```
salt-run state.event pretty=True
```

Manually adding a minion to salt

To get a lab machine to use salt, we need to do a few things on the minion

- install `py39-salt`
- enable the `salt_minion` service
- privide an IP address for the salt master to the minion via the minion configuration file
- start the `salt_minion` service

Then we need to accept the key on the master

```
salt-key -A
```

Automating adding a lab machine to the network and connecting it to salt

I'll use Ansible to get my new lab machine onto the network. In this case, we want an update caching server. Our salt master already has some states in place once this server connect it will be able to check in with the master and configure itself to begin playing its role as a freebsd-update cache server.

There's another folder with an Ansible Playbook for adding a virtual host to the lab, after changing to that directory we can use a feature of Ansible to replace values our Playbook will use from the command line.

The unique bits about a lab machine are it's hostname and IP address so we'll provide those details to Ansible

```
ansible-playbook -vv playbook.yml -e "lab_client_hostname=update ip_address=3"
```

Inspecting traffic to/from our lab network

We can see what traffic is going to and coming from the internet with TCPDump on the gateway's external interface

`tcpdump -vi e0b_gateway proto TCP`

We can now run a `freebsd-update fetch` to see all the traffic we're caching

When we run `freebsd-update fetch` again on another lab host, now we can see pretty much all of the files are cached and distributed internally via our LAN interface to our lab clients

We can run freebsd-update from salt with

```
salt '*' freebsd_update.fetch
salt '*' freebsd_update.install
```

Questions

Feel free to email me, roller@bsd.pw

I'm starting a BSD User group and would love to share cool things. It's called PyBUG for Python BSD User Group - cool things we can do with Python on FreeBSD.