SaltStack from Scratch

Mark Bools

December 10, 2020

Document ID: A000000002
Last Modified: 2022-01-25

Contents

Chapter 1

How to...

This chapter offers some guidance on getting the most from this book.

1.1 ...read this book

If I was being flippant I might say, "with your eyes" but I'm bigger than that so instead I will suggest some ways you might use this material.

This book is organised as a single narrative course centred around Salt. If you start on page one, read through each page and follow along with all the material, you should end up being proficient with Salt by the end.

This book contains mistakes. Deliberate mistakes (and no doubt some mistakes that I did not intend, that's life). Sometimes we will do things more than once. WTF? In most educational material we are presented with 'perfect' solutions, this is not realistic. The real world sucks. It changes constantly and today's ideal solution is okay but tomorrow the boss (or customer) decides new technology is desirable how do we handle this? This is were some soft skills may be required to persuade them to change their mind. Assuming we cannot persuade them to change we need to plan and execute a migration from the older solution to the new solution. Rather than avoid this sort of complexity this course takes it head on.

Don't have time to follow along from the start? Perhaps you already know enough about Salt's state system and feel confident skipping that material. No problem. At the start of each section you will find details of how to create an appropriate environment for that section (see §1.3). These 'checkpoints' also mean that if you mess up you can simply throw your environment away, recreate it using the closest checkpoint and continue with the course. In fact I encourage you to mess up your environment. You will learn much more by 'playing'. So set up and environment, mess around with your own ideas and then, when you are ready to do the next part of the course, either restore your own saved snapshots or tear down the environment and build a new one using the provided checkpoint code.

Having difficulty? No problem. Ask for help. Someone in the community may help and since I am in the community I can also help clarify things. If enough people are confused by the material then obviously I messed up and need to rewrite that material to be more clear; it shall be done.

1.2 ...get the most from this book

Do the examples! You'll get much more from the material by following along and investigating for yourself.

1.3 ...manage your workspace

We are going to be doing a lot of practical work throughout this book so it is worthwhile considering how we will manage our workspace.

A lot of this section will only make sense once you have read about the tools Git, VirtualBox, and Vagrant but I'm assembling basic advice here to make it easier to refer to later.

1.3.1 Initial setup of your workspace

If you follow this book from page one to the end you should find your workspace is always in sync with whatever the book is dealing with but practically many of you will jump to sections of particular interest, skipping many sections. Anticipating this I've put in plenty of checkpoints. All of the material (including checkpoints) is held in a single Git repository, so I recommend getting that first.

Assuming you have installed Git (see §2.4) you can create your project workspace.

bash

```
1 mkdir sfs
2 cd sfs
3 git clone
  ↳ https://gitlab.com/saltyvagrant.classes/sfs-material.git
4 mkdir classroom
5 mkdir archive
```

Your `sfs` workspace now contains three directories; `sfs-material` holds all this books accompanying material, `classroom` is where you will follow along with the course, and `archive` is where we will store various backup files.

Throughout this book I use the `sfs` directory as the root of your workspace. Any path that does not explicitly start from the `sfs` root assumes you are following instructions from the last checkpoint and are relative to whichever directory you should be in at the time.

bash

```
1 cd sfs
2 cd sfs/classroom
3 cd ../sfs-material
4 cd sfs
5 cd classroom
```

Lines 1, 2, and 4 each start with `sfs` and are therefore not really relative to your current working directory. You should take these to be absolute directories rooted at your workspace root `sfs`.

Line 3 is relative you your current working directory (in this example `sfs/classroom`) and is refering to `sfs/sfs-material` (since the parent of `sfs/classroom` it `sfs` and `sfs-material` is to be found directly under this directory).

Line 5 is again relative to your current working directory. As you just moved to `sfs` (line 4) this refers to your `classroom` directory under that root.

1.3.2 Regular activities

There are a number of actions you may want to repeat throughout this course. Rather than repeat them in full each time I present them here and simply refer to these entries as necessary.

1.3.2.1 Snapshot Classroom

If you are following the advice given above and 'playing' with your classrooms then I suggest your take a snapshot of your environment just before you start to play. This way you can quickly reset your classroom back you a point where it is ready for you to continue following along with this course.

1. Follow along with this course.
2. Decide to 'play' for a while so take a snapshot.

   ```bash
   cd sfs/classroom
   vagrant snapshot save class_<date>
   ```

   Replace `<date>` with the date of the snapshot (I recommend using a `YYMMDD` format as this sorts properly). For example, if today where December 3rd 2020 and I wanted to backup my classroom I would use the following.

   ```bash
   cd sfs/classroom
   vagrant snapshot save class_201203
   ```

3. Play with your classroom environment.
4. Decide to resume the course as described in this book.
5. Restore your classroom to the saved snapshot.

   ```bash
   cd sfs/classroom
   vagrant snapshot restore class_<date>
   ```

   Where `<date>` is the date of the snapshot to be restored. For example, to restore the snapshot from December 3rd 2020 we created earlier I would use the following.

```bash
cd sfs/classroom
vagrant snapshot restore class_201203
```

6. Resume course.

One other useful snapshot command is `list`, this can be used to show your previously saved snapshots (useful if, like me, your forget this sort of thing).

```bash
cd sfs/classroom
vagrant snapshot list
```

1.3.2.2 Checkpoint Classroom

If you get lost in the material you can reset your classroom to one of the checkpoints in the book. This will clean up you `classroom` directory ensuring you are ready to proceed with the book's follow-along lessons.

1. Shutdown any running classroom Virtual Machine (VM)[1]

   ```bash
   cd sfs/classroom
   vagrant halt
   ```

2. Backup your current classroom

   ```bash
   cd sfs
   mv classroom archive/classroom_<date>
   ```

   Replace `<date>` with the date of the backup (I recommend using a `YYYMMDD` format as this sorts properly). For example, if today where December 3rd 2020 and I wanted to backup my classroom I would us the following[2].

   ```bash
   cd sfs
   mv classroom archive/classroom_20201203
   ```

3. Copy the relevant material from `sfs-material`

4. Start up the classroom

   ```bash
   vagrant up
   ```

   This will start up any VM required for the classes.

[1] A segmented presentation or emulation of a physical computer allowing multiple 'guest' machines to share the physical resources of the 'host' computer.

[2] Windows users should use `move` rather than `mv`

1.3.2.3 Update material

This book, and consequently the accompanying material, is continually being updated[3]. You will want to periodically update your workspace to get these updates, so here's what I recommend you do just before you start any session.

```bash
cd sfs/sfs-material
git pull
```

This will update the course material.

What if this updates material I've already used? This should not cause problems in most cases. However, occasionally I will make significant revisions to the book and consequently you may see larger changes than usual. Assuming you are using the recommended workspace layout (see §1.3.1), the `checkpoint` script can be used to test your current environment against the current checkpoint.

```bash
cd sfs/classroom
vagrant ssh -c 'checkpoint verify'
```

Occasionally the `checkpoint` script will identify changes that requires one of more VMs to be recreated. Since the `checkpoint` script runs inside one of your classroom VMs it cannot preform operations on your host computer. In these circumstances the script will provide instructions to be run on your host to update your classroom.

If this verify fails there are two potential solutions.

1. Scrub your current `classroom` and return to the nearest checkpoint.
2. If the `checkpoint verify` can update your `classroom` it will offer a '`fix [y/N]?`' prompt. If you enter a '`Y`' then the checkpoint script will update your current `classroom` (and any VMs in that `classroom`).

[3] If you have downloaded the PDF version of this book then you should download the latest version at the same time you update the course material, otherwise they will get out of sync.

Chapter 2

Setting Up Your Environment

In order that we are all seeing the same environment as we progress through the following material you will need to install three applications onto your computer (the host computer):

- `VirtualBox`
- `vagrant`
- `git`

Let's take a look at each and discuss why they are required.

2.1 VirtualBox

VirtualBox is Oracle's virtual machine application. This allows us to run a virtual (guest) machine on our host computer. This in turn means that even if you are running, for example, a Windows PC you will be able to run the Linux servers required to follow along with this material.

Virtualisation also isolates our host computer from the machines that we use. This has the advantage that no matter how badly we mess up the virtual environment it will have no effect on our host computer and any change to our host computer will have no effect on the virtual machines[1].

2.2 Vagrant

Vagrant is HashiCorp's command line tool for managing virtual machines. Vagrant provided a simple consistent method for defining virtual machines as code. This means we can all easily set up the same virtual machine environment without the need to rely on following complex set up instructions.

As with many topics covered in this course, there is a more detailed book covering Vagrant *Vagrant from Scratch*[Boo20b].

[1] This is not 100% true, but close enough for our purposes here.

2.3 git

Git has become the *de facto* standard in version control tools. Git is a powerful tool, unfortunately its history means it has a bloated command line interface that is often daunting and confusing to newcomers. Fear not! We will initially use `git` commands to obtain some files and nothing more (so you can just type the commands with no need to understand them) but as we progress we will explain the `git` command line and if you are interested in learning Git in detail there is a complete book on the topic *Git in Depth*[Boo20a].

2.4 Installing the host tools

I have prepared some brief installation videos but to get the most up-to-date instructions for installing these host tools follow the instructions on their web sites.

Part I

Roadmap

Author Note

Overview of SaltStack.
Quick demo.
Plan of attack.

Part II

Configuring a Single Machine

Author Note

Introduce all the basic configuration mechanics.

To introduce the basic mechanics of Salt configuration management we use it to build out a fairly simple server.

Chapter 3

Installing Masterless Salt

One of Salt's modes is running locally. In this chapter we install Salt onto a server with the intention of using it locally.

3.1 Using bootstrap

Chapter 4

Introducing `salt-call`

`salt-call` invokes Salt locally.

4.1 Starting at the End

Setup to Follow Along

If you have not done so already, setup according to §1.3.1 and, if you need to archive any current class files §1.3.2.2.

```bash
cd sfs
```

On Mac/Linux:

```bash
cp -r sfs-material/sfsP2020cp010 classroom
```

On Windows:

```bash
xcopy sfs-material\sfsP2020cp010 classroom /E
```

All Platforms:

```bash
cd classroom
vagrant up
```

Once your classroom VM is available connect using SSH.

```bash
vagrant ssh
```

This VM is preloaded with a Salt configuration to build this server into a simple web server consisting of `nginx` serving a simple static website form `/var/www/html`.

Run Salt locally on this server using:

```bash
sudo salt-call state.highstate
```

This command will take a while to run, during which nothing will appear to happen. On completion you will see results dumped to the screen.

On your host computer, open a web browser and visit `http://localhost:35555`. You will see a simple web site being served from the VM.

Part III

Configuring Remote Machines

Author Note

`salt-ssh`. Running arbitrary commands remotely. Running configurations remotely.

Configuring a local machine allowed us to focus on all Salt's core configuration management functionality. This can be useful but most organisations use more than one server and those servers are often configured to work together. Configuring multiple servers as individual servers is possible but problematic. Salt provides `salt-ssh` for configuring remote machines using your enterprise SSH infrastructure.

Part IV

Controlling Many Machines

Author Note

Setting up Master/Slave. We switch to 'control' because now we're stepping into a 'live' environment.
Doing away with SSH infrastructure.
Proxy minions.
Access control.
Monitoring; beacons, reactor.
'self-healing' infrastructure. Increased security.
Multi-master setup.

`salt-ssh` is useful for directly distributing your multi-server configuration over your SSH infrastructure. Now now we move on to Salt's Master/Minion facility, this allows distribution of a configuration across multiple machines without an SSH infrastructure. It also allows much more; monitoring of your infrastructure combined with reacting to event in your infrastructure allows you to create self-healing systems.

Part V

Custom Salt

Author Note

Custom plugins, modules, etc.

Sometimes you find Salt is missing some facility you need. No problem. Add the functionality. Salt's plug-in architecture provides a core bus via which events and data are passed between a Master and one or more Minions, beyond this all functionality is provided by plug-ins.

Bibliography

[Boo20a] Mark Bools. *Git in Depth*. From Scratch. 2020. URL: https://saltyvagrant.com/books/git/git.html.

[Boo20b] Mark Bools. *Vagrant from Scratch*. From Scratch. 2020. URL: https://saltyvagrant.com/books/vagrant/vagrant.html.

Index