Building Our First Salt Project

DevOps 103

What Are We Building?

- The Gateway servers regularly require the upload of new firmware revisions. We will be attempting to build a project that automates the lifecycle of this process.

Requirements

- Transcribe commands from Davos' notes and a change control document
- Copy firmware to a "lead" GW server.
- Perform checksum to verify firmware has not been modified from original.
- Make commands modular, such that they can support differing versions, packages, products, etc.
- Build-in capabilities not only to provision, but to deprecate firmware.

Out-of-scope

- Installing any software (cmssupport is a part of STOP RPM packages)
- Configuring cmssupport

Logic

If / Then / Else

```
INSTALL_PACKAGE=True
PACKAGE=BluTag

if $INSTALL_PACKAGE == True
then
    yum install -y $PACKAGE
    echo "$PACKAGE installed!"
else
    echo "$PACKAGE will not be installed.
done
```

For each

```
$PACKAGES="$REDACTED,$REDACTED,
$REDACTED"

for $PACKAGE in $PACKAGES
do
    yum install –y $PACKAGE
    echo "$PACKAGE installed"
done
```

Test-Driven Development

- Tests are ALWAYS about what data you are feeding into a system, and what results you expect to see.
- In Salt, tests can be as simple as creating pillars before you start working with code.
- Pillars are ultimately what will be used to perform integration tests, when using a tool like KitchenCI.

Create a Pillar File

- To begin, we will create a pillar file that contains a data structure we would like to use.
- We will save this under:
  - ./pillar/gateway.sls

```
gateway:
  firmware:
    packages:
      $REDACTED:
        V7_9_1_27:
          target: ALL
          version_simple: 9.1.27
          present: True
          checksum: 15BC
          type: T
          bit_length: 128
```

Add Pillar to Top File

- Now, let's assign our pillar to an environment and a target in the top file:
  - ./pillar/top.sls

```
development:
  '*gw*':
    - gateway
```

Create a Salt State

- With pillars out of the way, let's create our first state file:
  - ./salt/gateway/init.sls

Include:

- gateway.configure

Create a Salt State

- Now, let's create a second:
  - ./salt/gateway/configure.sls
- Let's also update our top file:

```
development:
  '*gw*':
    - gateway
```

Working with “configure.sls”

- Looking at our code, it is clear that most of what we need to do will be performed over the command line. Thus, we should use the cmd.run function.
- Let’s create the most simple of states – just to confirm that cmd.run is functioning correctly.

```
gateway-configure-{{saltenv}}:
    cmd.run:
        - name: echo “Testing Salt function!”
```

Testing “configure.sls”

- With that completed, let’s commit and push our code to the “development” branch.
- Now, run the following command to test your changes:
  - salt ‘d1gw01’ state.highstate saltenv=development pillarenv=development

```
[root@appsalt01 salt]# salt 'd2gw21' state.highstate saltenv=development pillarenv=development
d2gw21:
17:28:27.845828 [902.429 ms]        pkg.installed  Clean     Name: logrotate-pkg
17:28:28.751041 [155.762 ms]       file.managed    Clean     Name: /etc/logrotate.conf
17:28:28.907078 [  1.033 ms]       file.directory  Clean     Name: /etc/logrotate.d
17:28:28.910010 [ 44.766 ms]    service.running    Clean     Name: crond
17:28:28.955477 [ 49.114 ms]       file.managed    Clean     Name: /etc/logrotate.d/tdcc
17:28:29.005068 [ 48.497 ms]       file.managed    Clean     Name: /etc/logrotate.d/pm2
17:28:29.054088 [ 46.795 ms]       file.managed    Clean     Name: /etc/logrotate.d/psacct-rpm
17:28:29.101352 [ 49.825 ms]       file.managed    Clean     Name: /etc/logrotate.d/stop
17:28:29.151702 [ 49.855 ms]       file.managed    Clean     Name: /etc/logrotate.d/rabbitmq
17:28:29.202065 [ 49.465 ms]       file.managed    Clean     Name: /etc/logrotate.d/kafka
17:28:29.252020 [ 47.735 ms]       file.managed    Clean     Name: /etc/logrotate.d/kannel
17:28:29.300612 [ 49.859 ms]       file.managed    Clean     Name: /etc/logrotate.d/mysql
17:28:29.351135 [ 57.246 ms]       file.managed    Clean     Name: /etc/logrotate.d/apache
17:28:29.409211 [ 48.328 ms]       file.managed    Clean     Name: /etc/logrotate.d/enrolleepay
17:28:29.458039 [ 48.026 ms]       file.managed    Clean     Name: /etc/logrotate.d/salt

Summary for d2gw21
-------------
Succeeded: 15
Failed:     0
-------------
Total states run:     15
Total run time:    1.649 s
```

Working with “configure.sls”

- Now that our states are working properly, let’s feed some data into them. To do this, we must import pillar data.

```
{% for package, versions in salt['pillar.get']('gateway:firmware:packages').items() %}
gateway-configure-{{ saltenv }}-{{ package }}-{{ version }}:
    cmd.run:
        - name: echo “Installing {{ package }} version {{ version }}!”
        - runas: tdcc
        - cwd: /usr/local/tdcc/cmssupport
{% endfor %}
```

Working with “configure.sls”

Expected Result

- Now, if we run another highstate, we expect to get this result:
  - Installing $REDACTED version V2_11_00!

Actual Result

- Instead, we get this result:
  - Installing $REDACTED version V2_11_00!
  - Installing $REDACTED version V1_11_00!

Working with “configure.sls”

- This happens because we have not excluded packages from installation.
- To do this, we must use if/then/else logic.

<See project code>

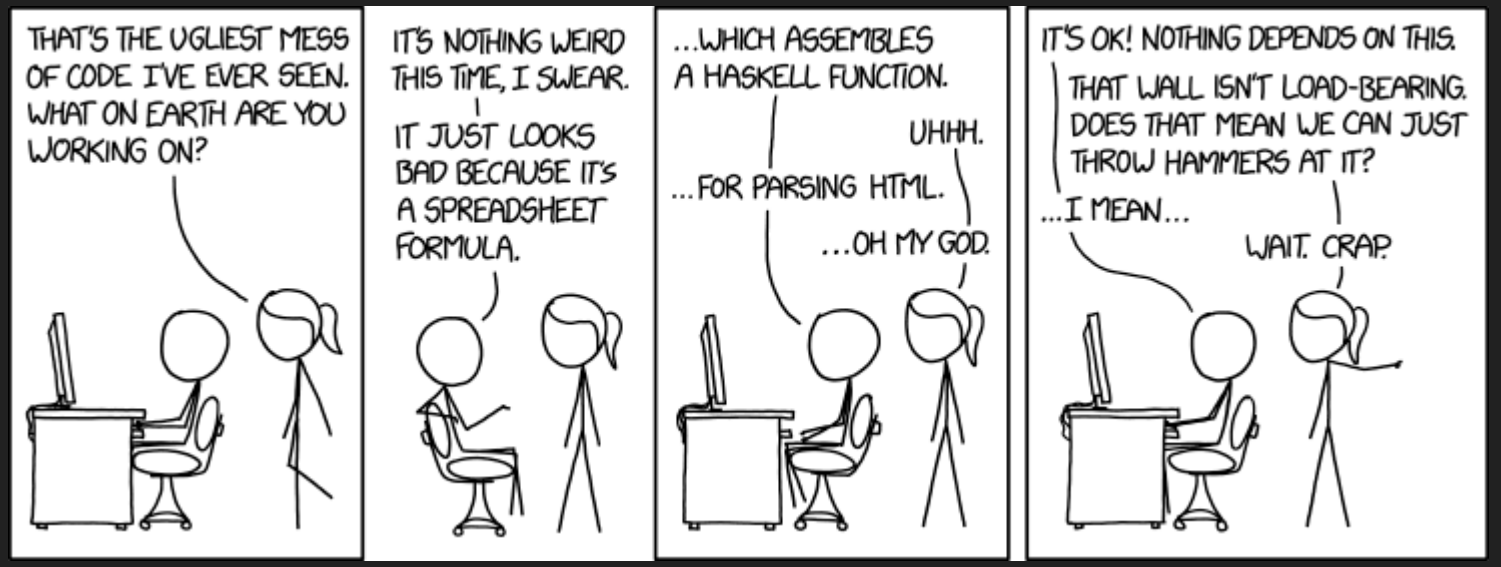
Testing “configure.sls”

- From here, we should be able to test installing/removing packages by toggling data in our pillar files.

Moving Forward

- Our project is far from complete. There are many more commands we must configure, and there are inevitably going to be flaws in our design. Revision and testing will be necessary.
- Remember to make incremental and small changes. Changing too much at once adds unnecessary complexity to the troubleshooting process.

Questions?


THAT'S THE UGLIEST MESS OF CODE I'VE EVER SEEN. WHAT ON EARTH ARE YOU WORKING ON?
IT'S NOTHING WEIRD THIS TIME, I SWEAR.
IT JUST LOOKS BAD BECAUSE IT'S A SPREADSHEET FORMULA.
...WHICH ASSEMBLES A HASKELL FUNCTION.
...FOR PARSING HTML.
UHHH.
...OH MY GOD.
IT'S OK! NOTHING DEPENDS ON THIS.
THAT WALL ISN'T LOAD-BEARING. DOES THAT MEAN WE CAN JUST THROW HAMMERS AT IT?
...I MEAN...
WAIT. CRAP.