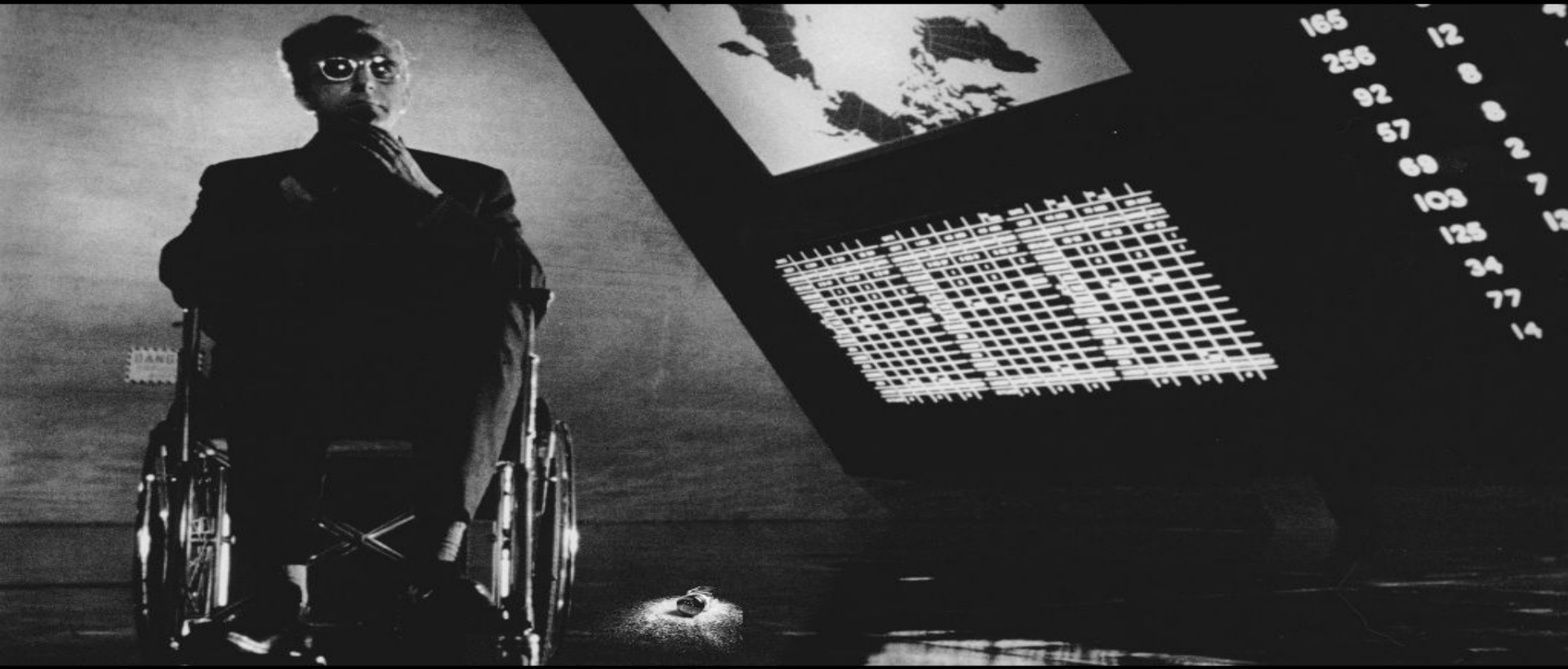
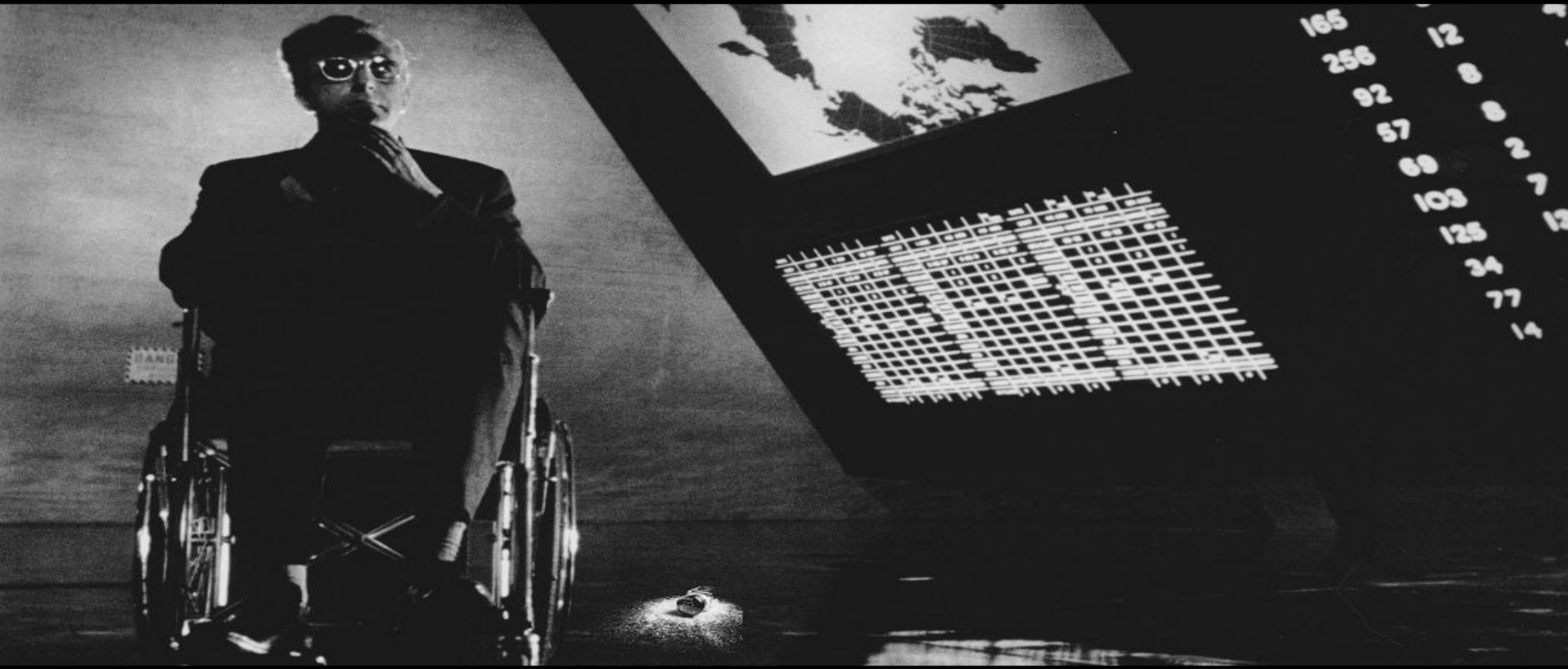
Dr. SaltStack



or: How I Learned to Stop Worrying and Replace the Cron

Dr. SaltStack

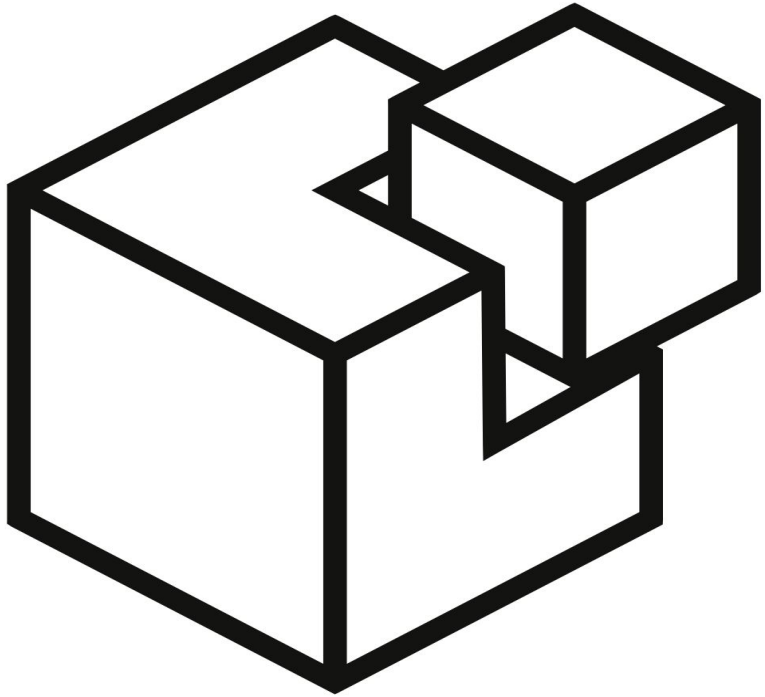


or: How I Learned to Stop Worrying and Replace the Cron

Hello
my name is

Gareth


@garethgreenaway

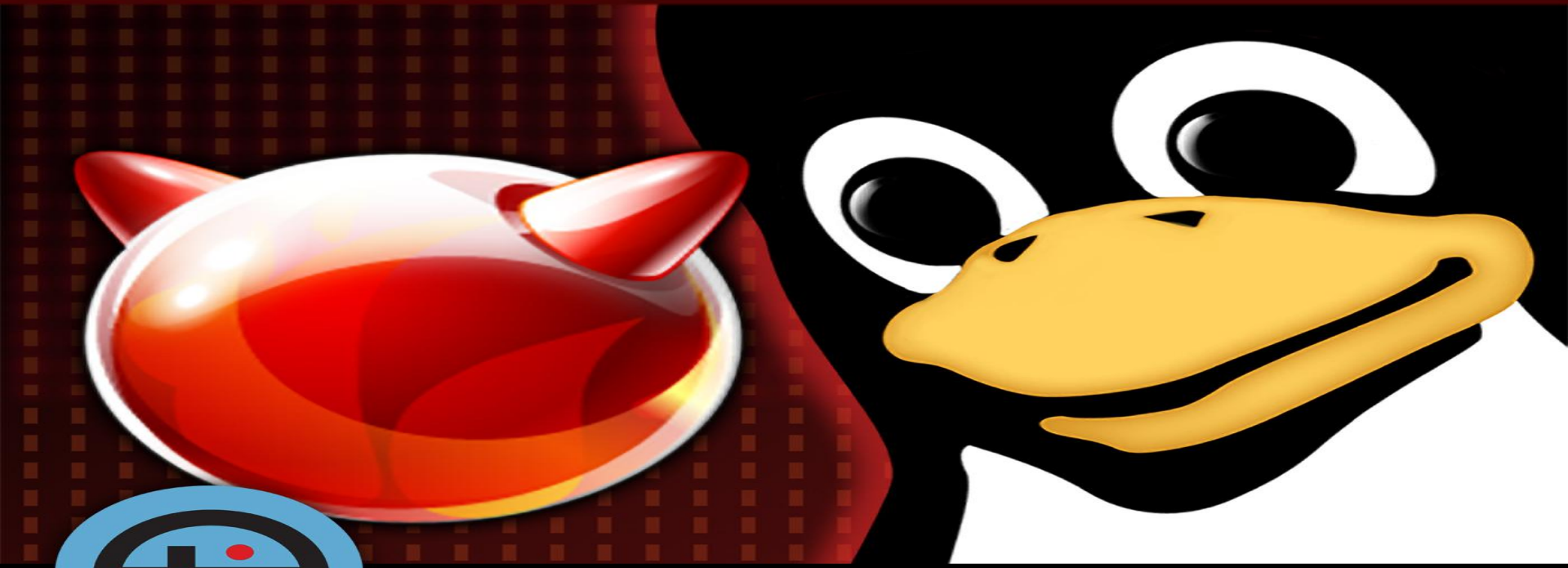


SALTSTACK

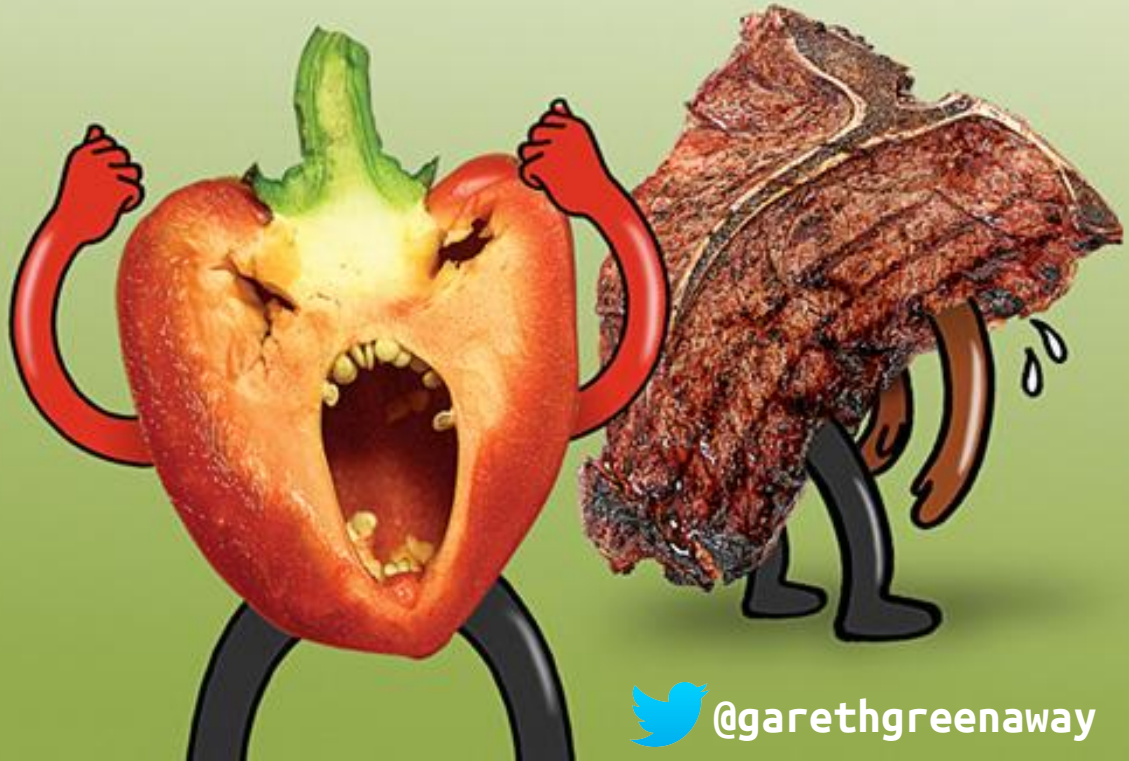
@garethgreenaway

@garethgreenaway

FLOSS WEEKLY



@garethgreenaway

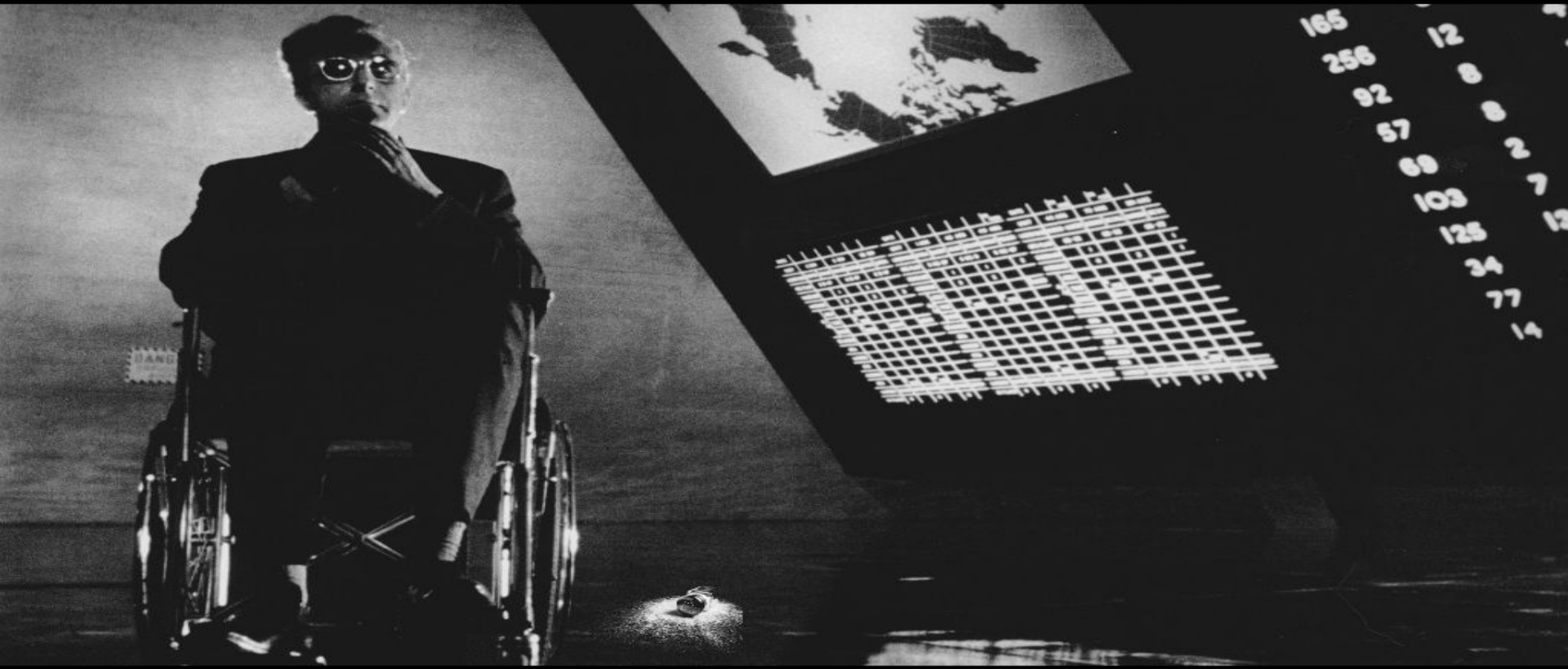


@garethgreenaway


GLOBAL COMMUNITIES
COMING TOGETHER
Leadership and Change
Market Challenges
SPC Group
@garethgreenaway

@garethgreenaway

Dr. SaltStack



or: How I Learned to Stop Worrying and Replace the Cron

Salt Basics

@garethgreenaway

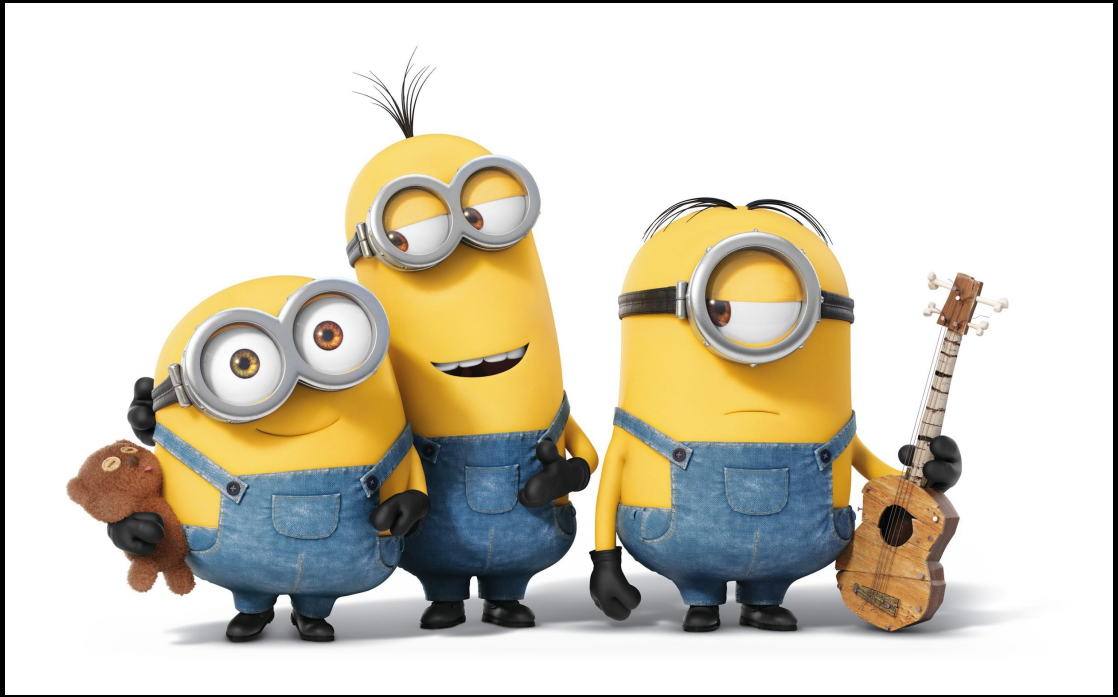
Salt Stack

The SaltStack platform or Salt open-source configuration management software and remote execution engine.

- Written in Python
- Execution Modules
- Salt Event Bus
- State Modules
- Salt API
- Pillar
- Reactors

@garethgreenaway

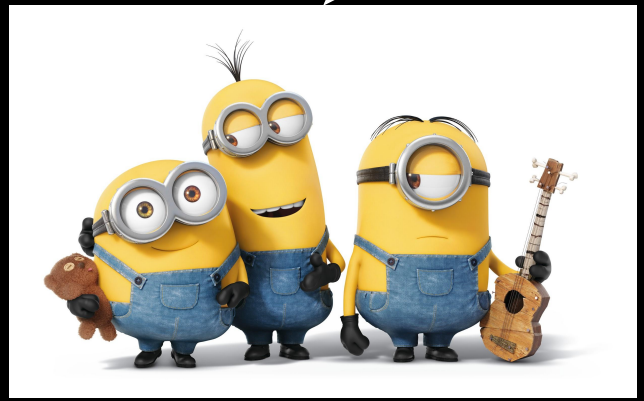
Minions

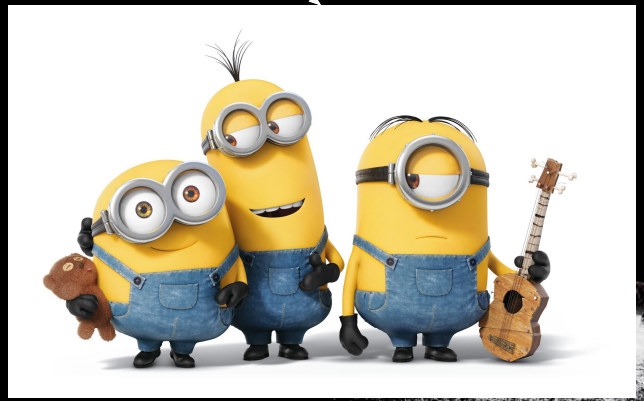


@garethgreenaway

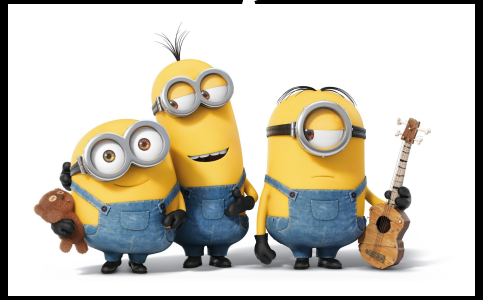
Master

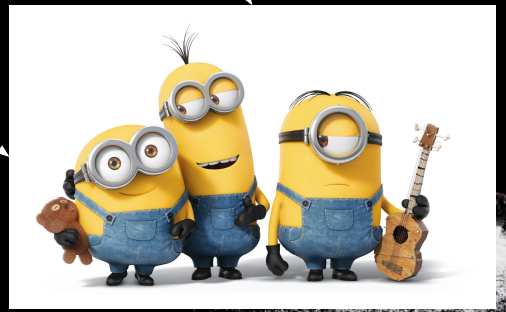
@garethgreenaway



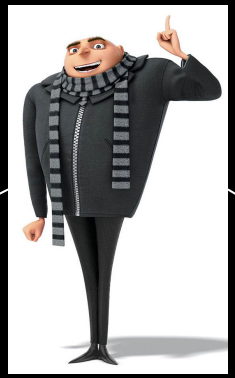


@garethgreenaway





@garethgreenaway



Syndic Masters

@garethgreenaway

@garethgreenaway

Scheduling Jobs

*under Un*x like operating systems.

@garethgreenaway

What we want

What We Want

1. Easily schedule a job.
2. Easy notification of job completion.
3. Different notification depending on job.
4. Schedule remotely across many nodes.
5. Enable and Disable.

A few different options

at

```
echo "cc -o foo foo.c" | at 11:45 jan 31
```

Pros

- Available on most Linux & *BSD systems, even Windows and OS X.
- Simple syntax to schedule jobs
- Management tools: at,atq, and atrm
- Notifications via email

Cons

- One time run.
- Node specific management.

What We Want

1. Easily schedule a job. ✳
2. Notification of job completion. ✳
3. Different notification depending on job. ❌
4. Remotely across many nodes. ❌
5. Enable and Disable. ❌

cron

MAILTO: user@example.com
00 20 * * * /home/user/command.sh

MAILTO: user@example.com
00 20 * * * /home/user/command.sh
MAILTO: admin@example.com
59 23 * * * /usr/sbin/service apache restart

Pros

- Also available on most Linux & *BSD systems, and Windows and OS X.
- Relatively simple syntax to schedule jobs
- Management tools: crontab
- Notifications

Cons

- Still node specific management.
- Having to check the man page for which column is which 🙂

What We Want

1. Easily schedule a job. ✳
2. Notification of job completion. ✳
3. Different notification depending on job. ✳
4. Remotely across many nodes. ❌
5. Enable and Disable. ❌

Alternatives

Manage ‘at’ jobs with Salt

@garethgreenaway

Execution Module

‘at’

Schedule ‘at’ job

salt '*' at.at <timespec> <cmd> [tag=<tag>] [runas=<user>]

Example:

salt 'node1' at.at 12:05am '/sbin/reboot' tag=reboot

State Execution Module

‘at’

Schedule ‘at’ job.

```
rose:
  at.present:
    - job: 'echo "I love saltstack" > love'
    - timespec: '9:09 11/09/13'
    - tag: love
    - user: jam
```

Manage ‘cron’ with Salt

@garethgreenaway

Similar execution and state modules for cron

```
salt 'node1' cron.set_job root '*' '*' '*' '*' 1
/usr/local/weekly
```

@garethgreenaway

```
date > /tmp/crontest:
  cron.present:
    - user: root
    - minute: 5
```

Still Limitations of both *at* and *cron*

Simply Use Salt

@garethgreenaway

Powerful Scheduler

Schedule Configured on Minion

```
schedule:
  job1:
    function: state.sls
    seconds: 3600
    args:
      - httpd
    kwargs:
      test: True
```

More precise by mimicking Cron.

```
schedule:
  job1:
    function: state.sls
    cron: '*/5 * * * *'
    args:
      - httpd
    kwargs:
      test: True
```

And more clear.

```
schedule:
  job1:
    function: state.sls
    when: 'Monday 8:15pm'
    args:
      - httpd
    kwargs:
      test: True
```

@garethgreenaway

Multiple runs.

```
schedule:
 job1:
   function: state.sls
   when:
       - ‘Monday 8:15pm'
       - ‘Tuesday 3:00pm’
   args:
     - httpd
   kwargs:
     test: True
```

@garethgreenaway

Another example

```
schedule:
  job1:
    function: cmd.run
    when: 'Monday 8:15pm'
    args:
      - 'logger -t salt < /proc/loadavg'
    kwargs:
      stateful: False
      shell: /bin/sh
```

What We Want

1. Easily schedule a job. *
2. Notification of job completion. ?
3. Different notification depending on job. ?
4. Remotely across many nodes. ?
5. Enable, Disable, and Move Jobs. ?

Notifications

Salt Returners

@garethgreenaway

Examples of returners:
Syslog, MySQL, PostgreSQL, Redis
SMTP, XMPP, Slack, Nagios

Scheduler + Returners

Notifications

```
schedule:
  job1:
    function: status.procs
    when: ‘8:15pm'
    returner: xmpp
```

Returner Configuration on Minion

@garethgreenaway

XMPP Returner Configuration

xmpp.recipient: to-jid@gmail.com

xmpp.jid: from-jid@gmail.com/salt

xmpp.password: 12345

What We Want

1. Easily schedule a job. *
2. Notification of job completion. *
3. Different notification depending on job. ?
4. Remotely across many nodes. ?
5. Enable and Disable. ?

Different Notifications

Alternative Returner Configuration

XMPP Returner Configuration

alt.xmpp.recipient: different-jid@gmail.com

alt.xmpp.jid: from-jid@gmail.com/salt

alt.xmpp.password: 12345

Notifications

```
schedule:
  job1:
    function: status.procs
    when: '8:15pm'
    returner: xmpp
    return_config: alt
```

XMPP Returner Configuration

xmpp.jid: from-jid@gmail.com/salt

xmpp.password: 12345

john.xmpp.recipient: john@gmail.com

bob.xmpp.recipient: bob@gmail.com

dba.xmpp.recipient: dba@company.com

What We Want

1. Easily schedule a job. *
2. Notification of job completion. *
3. Different notification depending on job. *
4. Remotely across many nodes. ?
5. Enable and Disable. ?

Remotely Across Many Nodes

@garethgreenaway

Remote Execution System

@garethgreenaway

Schedule Execution Module

```
salt -G 'role:webserver' schedule.add
apache_restart function='apache.signal'
args=”restart” seconds=3600
salt 'cache*' schedule.add varnish_purge
function=varnish.purge'
when=”[’10:00am’,’10:00pm’]”
```

Configuration Management System

@garethgreenaway

Schedule State Module

```
apache_restart:
  schedule.present:
    - function: apache.signal
    - args: restart
    - seconds: 3600
```

Schedule Jobs

```
job1:
  schedule.present:
    - function: state.sls
    - args:
      - httpd
    - kwargs:
      test: True
    - when:
        - Monday 5:00pm
        - Tuesday 3:00pm
        - Wednesday 5:00pm
```

@garethgreenaway

What We Want

1. Easily schedule a job. *
2. Notification of job completion. *
3. Different notification depending on job. *
4. Remotely across many nodes. *
5. Enable and Disable Jobs. ?

Disable, Enable and Move

Schedule Execution Module

@garethgreenaway

schedule.disable_job

salt -G ‘role:webserver’ schedule.disable_job apache_restart

schedule.enable_job

salt -G ‘role:webserver’ schedule.enable_job apache_restart

schedule.move_job

salt ‘webserver_new’ schedule.move_job apache_restart webserver_new

Other Available Functions

- schedule.delete
- schedule.disable
- schedule.enable
- schedule.list
- schedule.run_job
- schedule.modify
- schedule.purge
- schedule.reload
- schedule.save

What We Want

1. Easily schedule a job. *
2. Notification of job completion. *
3. Different notification depending on job. *
4. Remotely across many nodes. *
5. Enable, Disable, and Move Jobs. *


VIA 9GAG.COM


@garethgreenaway

So what can we schedule?

Salt has almost 300* modules
and roughly
3000* module functions.

* not all modules and functions available on all systems.

Other Interesting Scheduler Features

Splay

```
schedule:
  job1:
    function: state.sls
    seconds: 3600
    args:
      - httpd
    kwargs:
      test: True
    splay: 15
```

```
schedule:
  job1:
    function: state.sls
    seconds: 3600
    args:
      - httpd
    kwargs:
      test: True
    splay:
      start: 10
      end: 15
```

@garethgreenaway

Range

```
schedule:
  job1:
    function: state.sls
    seconds: 3600
    args:
      - httpd
    kwargs:
      test: True
    range:
        start: 8:00am
        end: 5:00pm
```

@garethgreenaway

Inverted Range

```
schedule:
  job1:
    function: state.sls
    seconds: 3600
    args:
      - httpd
    kwargs:
      test: True
    range:
        invert: True
        start: 8:00am
        end: 5:00pm
```

@garethgreenaway

return_job

```
schedule:
  job1:
    function: state.sls
    seconds: 3600
    args:
      - httpd
    kwargs:
      test: True
    return_job: True
```

metadata

```
schedule:
  job1:
    function: state.sls
    seconds: 3600
    args:
      - httpd
    kwargs:
      test: True
    return_job: True
    metadata:
      foo: bar
```

until

```
schedule:
  job1:
    function: state.sls
    seconds: 3600
    args:
      - httpd
    kwargs:
      test: True
    until: '12/31/2015 11:59pm'
```

@garethgreenaway

run_on_start

```
schedule:
  job1:
    function: state.sls
    seconds: 3600
    args:
      - httpd
    kwargs:
      test: True
    run_on_start: True
```

skip_during_range

```
schedule:
  job1:
    function: state.sls
    seconds: 3600
    args:
      - httpd
    skip_during_range:
      start: 8:00am
      end: 5:00pm
```

@garethgreenaway

Postponing jobs

Postponing Jobs

```
salt -G 'role:webserver' schedule.postpone_job
apache_restart current_time new_time
(time format %Y-%m-%dT%H:%M:%S)
salt -G 'role:webserver' schedule.postpone_job
apache_restart current_time new_time
time_fmt='%Y-%m-%dT%H:%M:%S'
```

@garethgreenaway

Skipping jobs

Skipping Jobs

```
salt -G 'role:webserver' schedule.skip_job
apache_restart time
(time format %Y-%m-%dT%H:%M:%S)
salt -G 'role:webserver' schedule.skip_job
apache_restart time
time_fmt='%Y-%m-%dT%H:%M:%S'
```

Next Scheduled Time

Skipping Jobs

```
salt -G 'role:webserver'
schedule.show_next_fire_time apache_restart
```

@garet

Any questions?

@garethgreenaway