Saltstack

Martin Bracher
Trivadis AG
Bern / Zürich

Schlüsselworte
Konfigurationsmanagement, Standardisierung, Remote Execution, Oracle Insallation

Einleitung

Saltstack ist einerseits ein **Remote-Execution Tool**. Es erlaubt es, von einem Master-Server aus auf einer Gruppe von entfernten Servern den gleichen Befehl abzusetzen, ohne sich selbst auf jedem einloggen zu müssen.

Saltstack ist andererseits ein **Konfigurationsmanagementsystem**, welches es erlaubt, Servern Rollen zuzuordnen und sie dadurch in bestimmte Stati zu versetzen. Beispielsweise um zu gewährleisten, dass bestimmte Softwarepakete installiert sind, oder dass gewisse Dienste laufen.

Saltstack ist ein Opensource Produkt, welches weitgehend in Python geschrieben ist. Es sind für viele Plattformen (auch Windows (ohne GUI...)) fertige Pakete verfügbar. Das Projekt hat eine sehr aktive Community und wird laufend weiterentwickelt.

Es gibt noch weitere Tools mit dem Fokus Konfigurationsmanagement, wie beispielsweise „Puppet“ oder „Chef“. Dieser Vortrag soll kein Vergleich der Tools sein, sondern es wird, aufgrund der gemachten Erfahrungen in einem grösseren Oracle-Umfeld, gezeigt, wie sich mit Saltstack die Installation der Grid-Infrastructure und der Datenbanksoftware (inklusive PSU) automatisieren lässt, sowohl für Standalone-Systeme als auch für RAC.

Terminologie

Master: Dies ist der zentrale Server, der Zugriff auf die anderen Server hat, dort Befehle ausführen kann, und die Konfigurationsfiles speichert.

Minion: Die vom Master-Server verwalteten Clients

Pillar: Eine globale Variable, die für alle Minions gilt

Grain: eine lokale Variable, deren Inhalt nur für einen Minion gilt. Beispielsweise die Standard-Grains CPU-Typ oder Hostname.

Installation

Für Oracle Enterprise Linux muss noch ein weiteres YUM Repository angefügt werden, danach lassen sich die fertigen Saltstack-Pakete inklusive der Abhängigkeiten installieren. Die Beschreibung befindet sich auf https://docs.saltstack.com/en/latest/topics/installation/rhel.html

```
yum install salt-minion #auf jedem Server
yum install salt-master #nur auf dem Master-Server
```

Sollte dabei der Fehler „Not Found“ auftreten, ist in der Repo-Datei „rhel$releasever“ durch den effektiven Release, z.B. „rhel6“ zu ersetzen.

Nach der Installation muss die Kommunikation zwischen Master und den Minions konfiguriert werden. In der Datei /etc/salt/minion muss der Name des Salt-Masters definiert werden. Bei der Gelegenheit sollte man dann in /etc/salt/minion_id den Namen des Minions (sinnvollerweise sein Hostname) eintragen. Danach kann man den Daemon auf dem Minion starten.

```
salt-minion -l debug  #initial mal interaktiv mit Debugging-Information
service salt-minion start #wenns funktioniert dann als Daemon
chkconfig salt-minion on
```

Auf dem Master ist dann der Key vom Minion zu akzeptieren, danach können die beiden Server miteinander kommunizieren.

```
salt-key –A name_of_minion
```

Stati

Ein Status ist ein bestimmter Setup eines Servers, also ein bestimmer Stand an Software, Konfiguration und laufender Dienste.

Stati werden in sogenannten „sls“ Dateien definiert, und befinden sich standardmässig unterhalb von /srv/salt. Die für unser Oracle-Setup benötigten Dateien kann man beispielsweise unter /srv/salt/roles/oracle ablegen. Wir machen beispielsweise eine create_grinf_rsp.sls Datei zum Erzeugen des Response-Files für die Grid-Infrastructure Installation.

Ein solches „state“ File lässt sich einzeln aufrufen, entweder direkt vom Minion aus, oder vom Master:

```
salt-call state.sls /roles/oracle/create_install_grinf_rsp  #Minion (host1)
salt host1 state.sls /roles/oracle/create_install_grinf_rsp #Master
```

YAML

State-Files sind in **YAML** (Yet Another Markup Language) geschrieben und werden vor der Ausführung gerendert.

Am Einfachsten lässt sich die Syntax wohl durch ein paar Beispiele erklären:

```
my_key:
  my_value:
```

Die einzelnen Ebenen müssen mit Leerzeichen eingerückt werden.

Listen-Werte sind mit einem „-“ zu kennzeichnen.

```
my_dictionary:
  - list_value_one
  - list_value_two
  - list_value_three
```

Je nach Schreibweise werden Kommentare ebenfalls gerendert, oder eben nicht:

```
{# comment that is not rendered {{var1}} #}
# comment that is rendered {{var1}}
```

In der ersten Zeile wird {{var1}} nicht interpretiert, in der zweiten Zeile wird es durch den entsprechenden Variablenwert ersetzt.

Vor dem Aufruf wurden folgende Grains gesetzt:

```
salt-call grains.setval grid_version 12.1.0.2
salt-call grains.setval db_version 11.2.0.4
```

Die sls Datei sieht wie folgt aus:

```
{% set grid_version = grains['grid_version'] %}     # error if undefined
{% set db_version = grains.get('db_version','') %}  # null if undefined
create_install_grinf_rsp:
  file.managed:
    - name:  /u00/app/grid/grid.{{grains['grid_version']}}.rsp
install_rdbms:
  cmd.run:
    - name: runInstaller -responseFile /tmp/rdbms.{{db_version}}.rsp -
silent
```

Das gerenderte Resultat dieses YAML Codes ist dann Folgendes:

```
create_install_grinf_rsp:
  file.managed:
    - name:  /u00/app/grid/grid.12.1.0.2.rsp
install_rdbms:
  cmd.run:
    - name: runInstaller -responseFile /tmp/rdbms.11.2.0.4.rsp -silent
```

Definition eines Loops:

```
{% for node in grains.get('ora_cluster_nodes', [] ) %}
...
{% endfor %}
```

Definition von if-then:

```
{% if grains.get('ora_cluster_name','') %}
  {% set asmdisks = salt['cmd.script']('salt://disk.pl').get('stdout') %}
{% endif %}
```

Module

Für viele Anwendungsfälle gibt es fertige Module, welche in den Stati verwendet werden können.

http://docs.saltstack.com/en/latest/ref/states/all/

Einige Beispiele:

- cmd: Betriebssystem-Kommandos auf dem Minion ausführen
- file: Erstellen, kopieren, herunterladen, bearbeiten von Dateien, insbesondere in Templates Variablen durch Werte zu ersetzen
- ssh_auth: Verwalten der authorized_keys Datei von ssh
- sysctl: Bearbeiten von sysctl Werten

Nach dieser kurzen Einführung in Saltstack kommen wir nun zum eigentlichen Thema, wie man Oracle-Server damit aufsetzen kann.

Im Vortrag wird erläutert, wie man RAC-Systeme, oder auch Single-Systeme aufsetzt. Nicht Bestandteil des Vortrages ist die Vorbereitung des Servers, also die Sysadmin-Arbeiten wie Storage bereitstellen, notwendige rpm-Pakete installieren, Kernel-Parameter setzen und der Oracle

Useraccount erstellen. Der Vortrag setzt also ab dem Zeitpunkt auf, wo der DBA normalerweise beginnt.

Für die Installation selbst müssen nur wenige Informationen konfiguriert werden:

- Die zu installierende Version der Grid-Infrasturcture- und Datenbank-Software
- Die Namen der Cluster-Nodes
- Der Cluster/SCAN Name

Bei einer manuellen Installation eines RAC Clusters ist es relativ einfach, die SSH-Keys zu verteilen. Man wartet, bis alle Server bereit stehen, loggt sich ein, generiert dann die Keys und verteilt sie per Copy/Paste von Session zu Session oder per scp mit Passwort-Eingabe.

Bei der Automatisierung ist das etwas schwieriger, da man keine Passwörter hinterlegen möchte, und da auch nicht sicher ist, dass alle Server gleichzeitig zur Verfügung stehen.

Die gewählte Lösung beruht nun auf einem Vorbereitungs-Status, welcher auf jedem Server denselben initialen SSH-Key installiert, Erst danach kann die eigentliche Installation starten. Die weitere Installation erfolgt dann vom ersten Node aus, der nun erforderliche Remote-Befehle selbst per SSH ausführen kann. Übrigens: den temporären Key tauscht er dann später Passwort-frei per scp durch einen definitiven, neu generierten Key aus.

Dieser erste Node wartet nach dem Vorbereitungs-Status, bis sämtliche anderen Nodes fertig vorbereitet sind.

```
check_if_rac_nodes_are_prepared:
  cmd.run:
    - name: while [ $(ls /tmp/*.ok |wc -l) -lt {{
grains['cluster_nodes']|length() }} ] ; do sleep 10; done
```

Dies erfährt er vom Master-Server, über einen sog. „Reaktor“ Mechanismus. Dieser schreibt auf dem ersten Host die Datei /tmp/<node>.ok, sobald sich ein Node gemeldet hat, dass er fertig vorbereitet sei; Danach erfolgt die Installation ziemlich „straight-forward“.

- Download und Entpacken der benötigten Software in der neusten Version

  Die Software steht auf einem Webserver zur Verfügung. Durch Einhaltung einer bestimmten Namenskonvention lässt sich automatisiert die neuste Version bestimmen. Hierfür wurde ein kleines Perl-Script entwickelt, welches aus dem html Directory Listing den aktuellsten Dateinamen zurückliefert.

- Erstellen der Response-Files

  Das Response-File steht als sogenanntes Template zur Verfügung, bei dem die wechselnden Werte wie etwa Cluster- oder Host-Name als Variable gekennzeichnet sind. Saltstack lädt dann dieses Template herunter und ersetzt die Variablen durch die für diesen Cluster gültigen Werte.

```
create_grinf.rsp:
  file.managed:
    - name: /u00/app/grid/grid.{{ grains['ora_grid_version'] }}.rsp
```

```
- source: salt:///templates/grid.{{grains['ora_grid_version']}}.rsp
- user: oracle
- template: jinja
- context:
  ORACLE_HOSTNAME: {{grains['fqdn']}}
  ...
```

Die entsprechende Zeile im Template sieht so aus:
`ORACLE_HOSTNAME=`**`{{ORACLE_HOSTNAME}}`**

- Installation der Grid-Infrastructure, inkl. root.sh auf allen Nodes
  Eine Besonderheit hier ist, dass sich „root“ nicht direkt Remote einloggen kann. Jedoch hat der Oracle-User „sudo“ Rechte. Vom oracle Account des ersten Nodes aus wird deshalb per ssh auf die anderen Nodes eingeloggt und dort via sudo das root.sh aufgerufen.

- Installation der Datenbank-Software

- Applizierung des aktuellsten PSU

- Erstellen der Standard ASM Diskgruppen

Im Vortrag werden entsprechende Syntax-Beispiele zu diesen Schritten gezeigt und erläutert.

Ebenso werde ich auf die Stolpersteine hinweisen, auf welche ich im Verlauf des Projekts gestossen bin, und wie man sie umgehen kann.

Kontaktadresse:
Martin Bracher
Trivadis AG
Europa-Strasse 5
CH-8152 Glattbrugg

Telefon: +41 58 459 56 56
Fax: +41 58 459 56 66
E-Mail martin. bracher@trivadis.com
Internet: www.trivadis.com