

saltstack 在运维平台上的应用

-----分享自競技世界_吴胜宝_2018 年 8 月 25 日

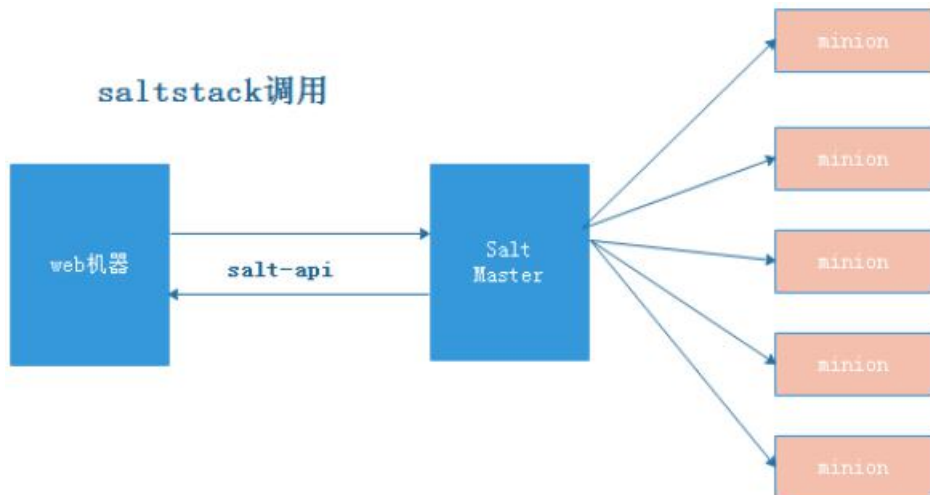
大概内容目录：

1	Salt 管理
1.1	key 管理
1.2	分组管理
2	Salt 命令执行
2.1	远程执行
2.2	部署安装
2.3	操作记录
3	安全限制
3.1	api 权限验证, saltapi 和 web ...
3.2	堡垒机鉴权
3.3	命令黑名单
4	实际运用
5	API 接口

分为 key 管理、分组管理、执行部分和安全部分，以及 api 和运用

一、key 管理：主要是管理连接 master 的主机，主要功能是接受秘钥，拒绝秘钥和删除秘钥。主要使用 Salt-key list 扫描节点并入库，更新新增节点和失效节点。

先说下 salt 调用的主要过程：



saltmaster 和 web 是分开的，master 有多台，web 机器和 master 之间通过 saltapi 来进行交互。先将指令下发给对应的 master，然后由 master 来往节点机器上下发执行。

二、分组管理

节点数量很多，有时候就需要分组，在页面上进行分组，存入本地 mysql，然后将分组信息通过服务同步到 master 机器上，具体做法是在 master 上写监听服务，web 机器发送同步请求时，会将分组信息传到 master 上，然后重启 master，让其生效。

+ 分组管理

tk-wu1 Salt主机分组			hb21-test Salt主机分组			tk-test Salt主机分组		
主机	状态	授权	主机	状态	授权	主机	状态	授权
HB21-LSP-PAR Salt主机分组			HB30-LSP-PAR Salt主机分组			HB10-LSP-PAR Salt主机分组		
主机	状态	授权	主机	状态	授权	主机	状态	授权
hb21-lsp-par	True	True	hb30-lsp-par	True	True	hb10-lsp-par	True	True
hb21-lsp-par	True	True	hb30-lsp-par	True	True	hb10-lsp-par	True	True
hb21-lsp-par	True	True	hb30-lsp-par	True	True	hb10-lsp-par	True	True
hb21-lsp-par	True	True	hb30-lsp-par	True	True	hb10-lsp-par	True	True

这是页面分组

编辑分组 | 分组管理

Salt分组: *

所有主机

Search...

BJZL-LY-
HP-DAT-
HP-DAT-
HP-M-LG-
TK-DAT-161
TK-DAT-162
TK-DAT-163
TK-Dat-0
TK-MGR-
XG-HaDo
bjcp-mgr-
bjcp-mgr-6-241
bjzl-ly-buil
bjzl-ly-md-20
bjzl-ly-md-11

已选主机

Search...

hb21-lsp-
hb21-lsp-
hb21-lsp-
hb21-lsp-

节点和分组的关系是 manytomany 关系，一个节点可以存在于多个分组，一个分组可以拥有多个节点。分组完成后，要进入到执行阶段了，执行分为同步执行和异步执行、单机执行和组执行。这些 saltapi 里都有提供，自己封装一下即可。

目标主机 模块 远程命令 失败跳过

L0 HP-DAT-MyCat27 命令执行 ifconfig

运行 查看结果

运行结果 重新查询 展开所有

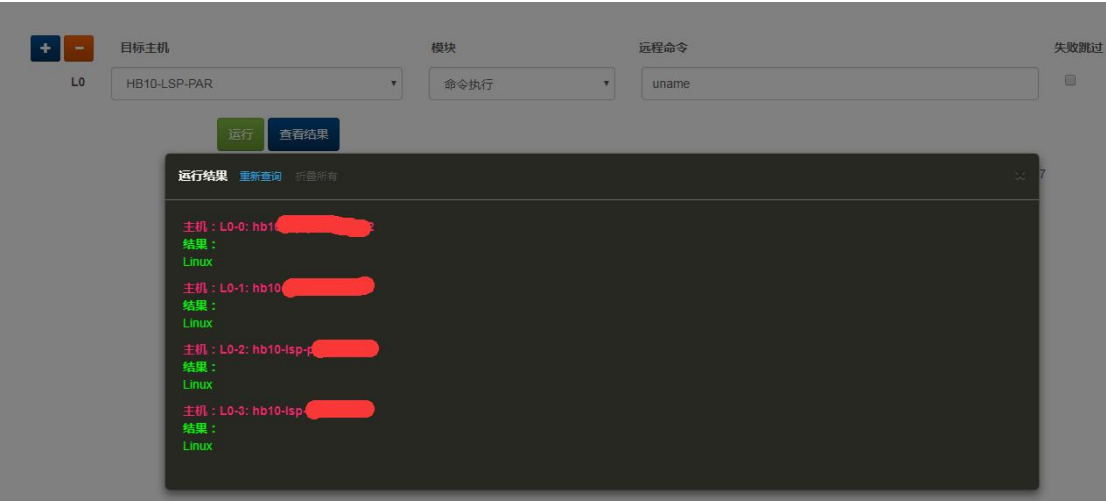
主机: L0: HP-DAT-MyCat27

结果:

```
eth0 Link encap:Ethernet HWaddr 00:50:56:91:3C:65
inet addr: 255.255.255.0 Bcast:255.255.255.0
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:219024167 errors:0 dropped:0 overruns:0 frame:0
TX packets:200286434 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:17208724393 (16.0 GiB) TX bytes:28384179118 (26.4 GiB)

lo Link encap:Local Loopback
inet addr:127.0.0.1 Mask:255.0.0.0
UP LOOPBACK RUNNING MTU:65536 Metric:1
RX packets:58372529 errors:0 dropped:0 overruns:0 frame:0
TX packets:58372529 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
RX bytes:15293167233 (14.2 GiB) TX bytes:15293167233 (14.2 GiB)
```

这是单机执行的页面



这是组执行的页面

模块部署

目标组 * HB21-LSP-PAR

包含主机 hb21-isp-par hb21-isp-par hb21-isp-par hb21-isp-par

安装模块 * 请选择

运行 查看结果

这是模块部署，部署原理就是上传相应的 sls 文件，然后调用 `state.sls` 执行即可，这点可以参照 saltapi 相关内容，比较简单。同时系统还会记录所有的操作记录，包括页面的操作记录和 saltstack 命令 的执行记录，方便以后回溯查询。

任务列表

任务ID: 20180821113313274253

25 records per page

用户	IP	操作时间	类别	操作	MasterIP	详细信息
吴胜宝	192.168.18.43	2018-08-21 11:33:15	远程命令	20180821113313274253	192.168.18.43	cmd.run ...
吴胜宝	192.168.18.43	2018-08-21 11:33:15	远程命令	20180821113313274253	192.168.18.43	cmd.run ...
吴胜宝	192.168.18.43	2018-08-21 11:33:15	远程命令	20180821113313274253	192.168.18.43	cmd.run ...
吴胜宝	192.168.18.43	2018-08-21 11:33:15	远程命令	20180821113313274253	192.168.18.43	cmd.run ...
吴胜宝	192.168.18.43	2018-08-21 11:32:40	远程命令	20180821113238043417	192.168.18.43	cmd.run ...
吴胜宝	192.168.18.43					cmd.run ...
吴胜宝	192.168.18.43					cmd.run ...
吴胜宝	192.168.18.43					cmd.run ...
吴胜宝	192.168.18.43					cmd.run ...
简武	192.168.18.19					tk-dat-mysql147,tk-dba-d...

运行结果 展开所有

```

主机: data
结果:
hb10-isp-par: Linux
hb10-isp-par: Linux
hb10-isp-par: Linux
hb10-isp-par: Linux

```

操作时间	2018-08-16 21:28:51
类别	更新Pillar
操作	更新Pillar
详细信息	hb30-dba-mysql BUSINESS: mysql DEPARTMENT: sscdba IDCNAME: HB30 IPADDR: 10 LOGTYPE: - lnxsys_cmdhis - mysql_error

记录操作时间，操作人，操作的节点以及操作的命令和结果。根据执行的 `jid` 可以回查结果，这个功能是 `salt` 自带的。



这是整个系统的一部分功能

还有一个脚本上传预览和执行的功能

上传文件

25 records per page

Search:

用户	目标主机	类型	本地路径	远程路径	备注	操作
吴胜宝	tk-mgr-opt-plt189	list	.saltconfig/fileupload/20180323172456/pingcheck_Tb2n251	/tmp	测试	🔗 ▶ 🗑
吴胜宝	tk-mgr-cmdb10...	list	.saltconfig/fileupload/20180327173721/wutest.sh	/tmp	test	🔗 ▶ 🗑
吴胜宝	tk-wu1	nodegroup	.saltconfig/fileupload/20180327173804/wutest.sh	/tmp	测试	🔗 ▶ 🗑
吴胜宝	hb21-k8s-nod				测试	🔗 ▶ 🗑

Showing 1 to 4 of 4 entries

Demo of ACE Editor - Google Chrome

安全 | https://ops.ijweb.cn/deploy/file_view/1

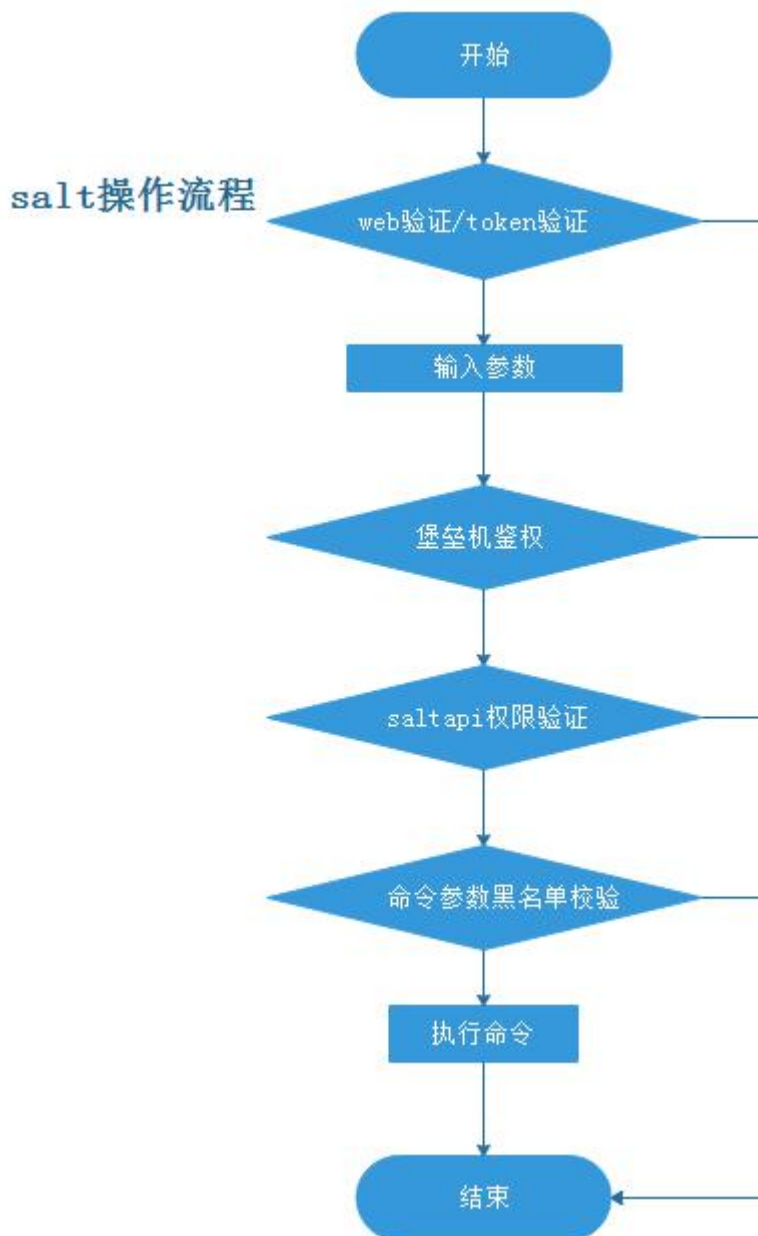
```
12 from IDC.models import *
13 from django.db.models import Count, Min, Sum, Avg
14
15 from threading import Thread
16 import subprocess
17 from Queue import Queue
18 import time
19 from IPy import IP
20 def genips(net):
21     ips=[]
22     for ip in IP(net):
23         ips.append(str(ip))
24     return ips
```

ijworld 男孩世界 @2017

这是执行方面的一些功能，当然还有文件回滚等等功能目前还没有集成进来。

三、下面来说说安全方面做的一些工作

由于 salt 是批量操作，稍有不慎，会造成很大的影响，所以在执行之前，需要做一些限制。



首先是 web 的页面验证，页面点击需要登录验证，api 调用需要 token 验证，保证随便就能操作的。这一步过去了，然后就验证操作人是否包含要操作的节点权限，具体做法是去堡垒机里面进行鉴权，每个用户在堡垒机里都会有权限列表，如果他要执行的节点没有权限就会直接打回请求。

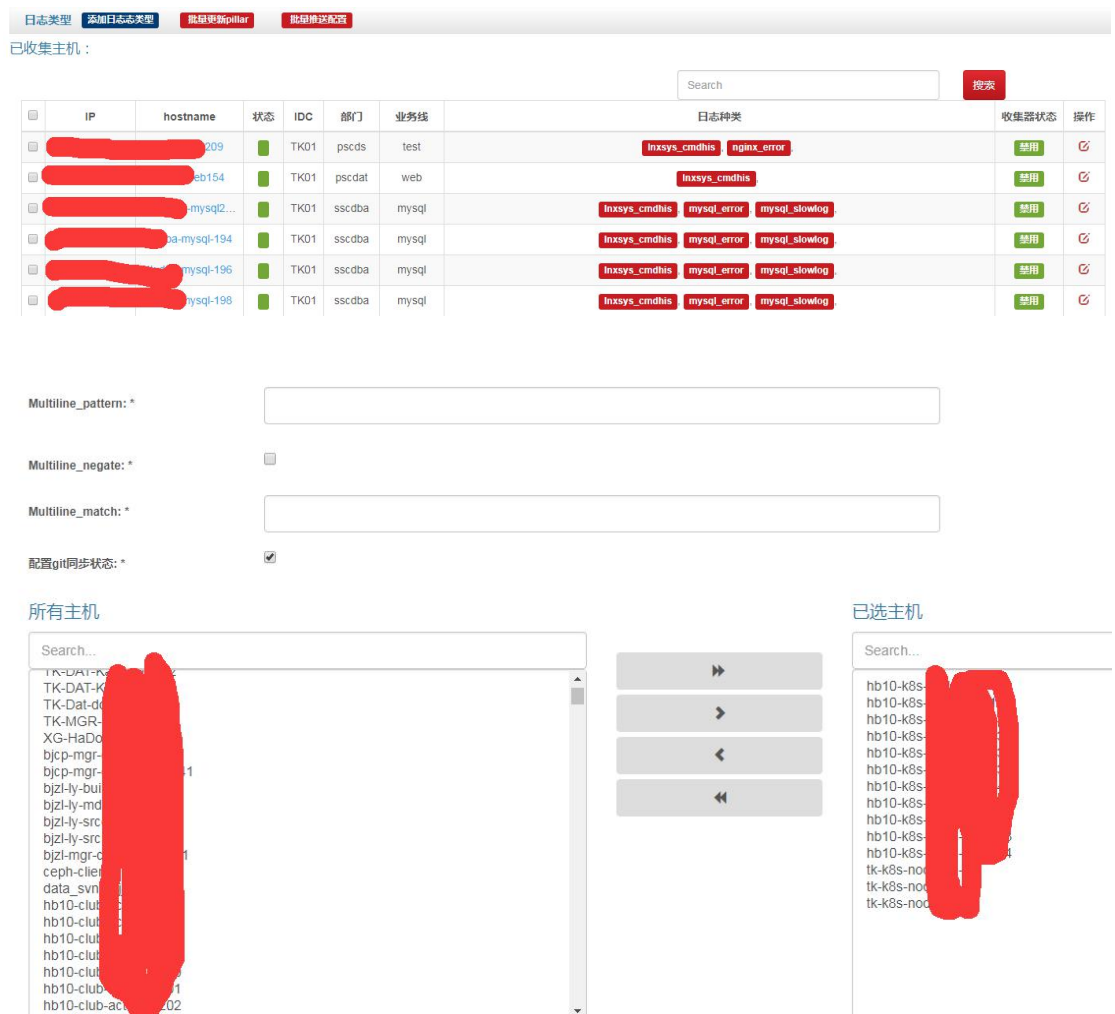
```
47 print getresponse(url,token,{'username':username,'hostname':'zw-clu', 'module':'cmd.run','cmd':'ifconfig','resul
48
49
50
{'u'token': u'62e4b04d7c5858ddd80d4eeb9a34b6ec7dc3ce6e'}
```

wushengbao 您好:
 您没有zw-clu 机器的操作权限，如果需要权限请向系统部申请！

这样进一步限制了安全范围，操作人只能操作自己有权限的机器，其它机器无法执行任何命令。这两步过去了，还要对参数进行简单的检查，会设定一个命令黑名单，像 rm、shutdown、init、等敏感命令都无法执行，黑名单可以根据自己需要自行加减。



如图，我在黑名单里加了 `uptime` 命令，当我要执行的命令行里含有 `uptime`，就不会执行，会返回提示信息。以上是 `salt` 相关的一些操作的集成展示，下面说说其他的一些应用。



这两图是用 `salt` 来推送 `elk` 日志配置的功能页面。根据节点不同的日志类型，会生成不同的日志文件，并进行推送。日志类型和类型中的节点都可以自由分配。先是根据主要信息生成 `pillar` 配置，然后根据 `pillar` 配置往下推送，日志配置的 `conf` 信息放在 `git` 上。要推送的时候，`slat` 会从 `git` 上拉取。

最后就是 `api` 接口了，目前只提供了两种接口，一种是执行接口，一种是根据 `jid` 查询结果的查询接口。执行接口根据参数设置分为同步执行和异步执行，参数为 `sync` 时会同步执行，直接返回结果，参数为 `async` 时会异步执行，只返回任务的 `jid`，然后用

户根据 `jid` 再调用查询接口去查询自己执行的任务结果。由于要经过层层校验，执行效率并不是太好，所以一般推荐用异步执行的方式，同时支持多个节点同时执行，`api` 不支持组执行，因为组权限不好控制。

以上就是我这次的主要分享内容了，欢迎探讨！谢谢！

ASK01:saltapi 走的 `py` 还是 `http`?

AN:应该是 `https`

ASK02:看图用的 `vue` 框架，又感觉页面 `ui` 有两个人系统?

AN:是两个.第一个是之前做的，没有用 `vue`.后来前后分离了，前端用 `vue`，后端还是 `django`.马上不搞 `vue` 了，准备转型 `react` 了.为了统一，可以寻求更多的技术支持.公司 `web` 都是用的 `react`，以后有复杂的功能页面可以使用他们的组件.或者直接寻求他们的支持.我们毕竟不是专业前端，复杂的的东西实现起来还是有点费力

ASK03:主机命名规范可以解读一下吗?

AN:我可以大概讲一下,因为这个都是部分很早就制定了.主机名规则是按照所在 `idc`、业务/部门、服务应用、`ip` 的后两位.目前只有物理服务器的主机名走 `dns`，映射的也是管理 `ip`，而不是业务 `ip`.命名规范因公司而异.

ASK:Django Token 走的 `JWT` 吗?

AN:像分组配置的同步和 `master` 的重启都是走的 `rpc`.我用的是 `rypc` 模块.没有，目前就是用的最基本的 `token` 验证，只是限制了一下 `ip` 和访问频次.