SALTCONF19

Highly available architecture with SaltStack Enterprise

SaltStack Enterprise 6.1

SaltConf19: Pre-Conference Training

November 18-19, 2019

COPYRIGHT AND REPRODUCTION NOTICE

BE RESPECTFUL

The labs are setup in a cloud environment shared by all students.

It is your responsibility to act respectfully to your fellow students, SaltStack, and the cloud provider.

Any malicious, abusive, or careless actions outside of the instructions provided in the lab will not be tolerated and may result in being removed from the class without any refund, and such individual may be monetarily responsible for any additional expense.

- We will all be sharing the same API key so be careful.
- Do not create VMs larger than the lab states.

FEEDBACK

We want to hear from you! We are always looking for ways to improve our products and services. Please send any feedback relating to SaltStack training services to: *training@saltstack.com*

Session Introduction

WELCOME

Welcome to SaltConf19 Pre-Conference Training!

In this tutorial, you will be introduced to the basics of creating flexible, event-driven orchestration workflows with Salt and SaltStack Enterprise through discussion and hands-on lab exercises.

OBJECTIVES

In this session, you will learn to build a robust and resilient SaltStack Enterprise architecture:

- Understand the SaltStack Enterprise Architecture components
  - PostgreSQL
  - Redis
  - SaltStack Enterprise Service (RaaS)

- Configure POSTGRESQL STREAMING REPLICATION
- Configure PGPOOL to act as endpoint and failover control for POSTGRESQL
- Configure REDIS REPLICATION and REDIS SENTINEL
- Configure multiple RAAS heads connected to same DB
- Configure HAPROXY as Load Balancer for RAAS
- Configure HAPROXY to act as endpoint and failover control for REDIS
- Perform survivability tests

SESSION FORMAT

This is a two-hour session. There will be roughly 30 minutes of lecture and discussion, followed by 90 minutes available for completion of the lab exercises. Lab assistants will be available if you have any questions.

LAB ENVIRONMENT

This is a technically complex lab requiring you to perform multiple configuration tasks on multiple servers on a given order.

Please pay close attention to lab instructions provided for each service/server set up, failure to do so will prevent you from succeeding on lab goals.

A total of 8 RHEL 7.6 servers have been provisioned for your lab, in some cases some minor configuration steps have been performed on your behalf to save lab time.

Each server has a public IP address for SSH access, you will receive a paper with your servers public ip addresses and private ip address.

To complete the lab exercises, you will use SSH to connect to each server public ip address. For most configuration cases, internal/private ip addresses must be used, the instructions will mention those cases in which the use of public ip address is needed (commonly to access RAAS servers via https or via HAPROXY from your computer)

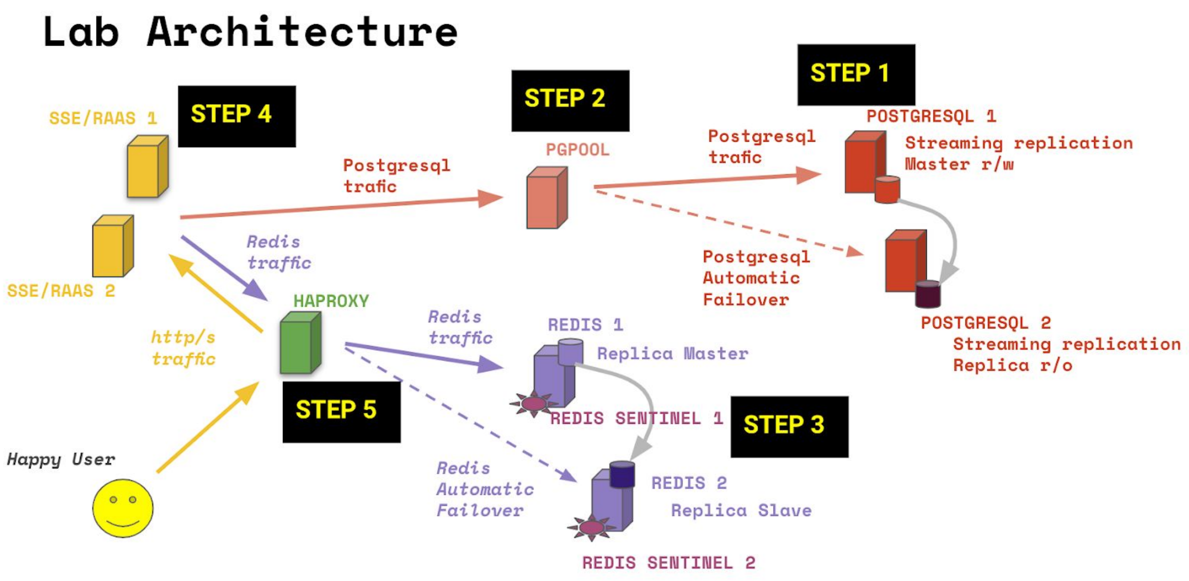
Servers List (in order of usage):

- **2 POSTGRESQL**
- **1 PGPOOL**
- **2 REDIS**
- **2 RAAS**
- **1 HAPROXY**

CONNECTING TO THE LAB ENVIRONMENT

Your instructor will provide you with a list of IP addresses, username, and password to connect to all your assigned instances.

LAB CONFIGURATION STEPS



- Step 1 - POSTGRESQL REPLICATION -- **TIME: 25 MINS**
  Connect to psql1 and psql2, follow instructions in /root/instructions_1-psql.txt

- Step 2 - PGPOOL -- **TIME: 15 MINS**
  Connect to pgpool1, follow the instructions in /root/instructions_2-pgpool.txt
  - *as part of the procedure, install pgpool_extensios in PSQL servers*

- Step 3 - REDIS REPLICATION AND REDIS SENTINEL -- **TIME: 10 MINS**
  Connect to redis1 and redis2, follow instructions in /root/instructions_3-redis.txt

- Step 4 - RAAS -- **TIME: 10 MINS**
  Connect to sse1 and sse2, follow instructions in /root/instructions_4-sse.txt

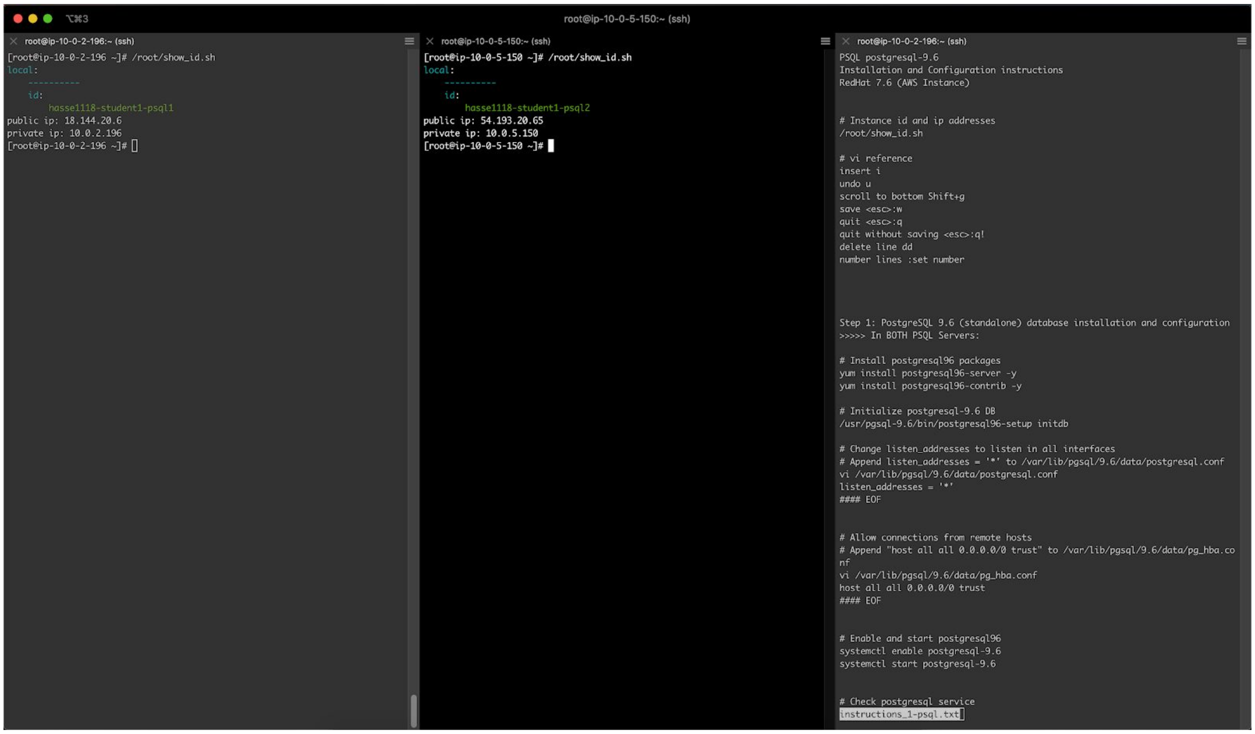
- Step 5 - HAPROXY -- **TIME: 10 MINS**
  Connect to haproxy1, follow instructions in /root/instructions_5-haproxy.txt

- Step 6 - *Survivability tests* -- **TIME: 10 MINS**
  Follow the instructions in this guide

Step	Group	Naming Convention	IP Addresses	Configuration Completed ?
1	PSQL	{{ session }}-student{{ id }}-**psql1** {{ session }}-student{{ id }}-**psql2**	**Public:** 1- 2- **Private:** 1- 2-	
2	PGpool	{{ session }}-student{{ id }}-**pgpool1**	**Public:** **Private:**	
3	Redis	{{ session }}-student{{ id }}-**redis1** {{ session }}-student{{ id }}-**redis2**	**Public:** 1- 2- **Private:** 1- 2-	
4	SaltStack Enterprise	{{ session }}-student{{ id }}-**sse1** {{ session }}-student{{ id }}-**sse2**	**Public:** 1- 2- **Private:** 1- 2-	
5	HAproxy	{{ session }}-student{{ id }}-**haproxy1**	**Public:** **Private:**	

TIPS:

- **Keep multiple ssh sessions open to same group of servers side by side, even to same server, to follow instructions or logs in one screen and perform actions in other screen**


⌥⌘3
root@ip-10-0-5-150:~ (ssh)
root@ip-10-0-2-196:~ (ssh)
root@ip-10-0-5-150:~ (ssh)
root@ip-10-0-2-196:~ (ssh)

```
[root@ip-10-0-2-196 ~]# /root/show_id.sh
local:
    ----------
    id:
        hasse1118-student1-psql1
public ip: 18.144.20.6
private ip: 10.0.2.196
[root@ip-10-0-2-196 ~]#
```

```
[root@ip-10-0-5-150 ~]# /root/show_id.sh
local:
    ----------
    id:
        hasse1118-student1-psql2
public ip: 54.193.20.65
private ip: 10.0.5.150
[root@ip-10-0-5-150 ~]#
```

```
PSQL postgresql-9.6
Installation and Configuration instructions
RedHat 7.6 (AWS Instance)


# Instance id and ip addresses
/root/show_id.sh

# vi reference
insert i
undo u
scroll to bottom Shift+g
save <esc>:w
quit <esc>:q
quit without saving <esc>:q!
delete line dd
number lines :set number




Step 1: PostgreSQL 9.6 (standalone) database installation and configuration
>>>>> In BOTH PSQL Servers:

# Install postgresql96 packages
yum install postgresql96-server -y
yum install postgresql96-contrib -y

# Initialize postgresql-9.6 DB
/usr/pgsql-9.6/bin/postgresql96-setup initdb

# Change listen_addresses to listen in all interfaces
# Append listen_addresses = '*' to /var/lib/pgsql/9.6/data/postgresql.conf
vi /var/lib/pgsql/9.6/data/postgresql.conf
listen_addresses = '*'
#### EOF


# Allow connections from remote hosts
# Append "host all all 0.0.0.0/0 trust" to /var/lib/pgsql/9.6/data/pg_hba.co
nf
vi /var/lib/pgsql/9.6/data/pg_hba.conf
host all all 0.0.0.0/0 trust
#### EOF


# Enable and start postgresql96
systemctl enable postgresql-9.6
systemctl start postgresql-9.6


# Check postgresql service
instructions_1-psql.txt
```

- **Use** */root/show_id.sh* **to identify system name and public/private ip addresses**

```
[root@ip-10-0-5-150 ~]# /root/show_id.sh
local:
    ----------
    id:
        hasse1118-student1-psql2
public ip: 54.193.20.65
private ip: 10.0.5.150
```

- **Pay very close attention to instructions, specially to what server execute each task, look for entries like this:**
  >>>>> In BOTH PSQL Servers:

- **Pay attention to replace** *ipaddress_servername* **with required server ip address**

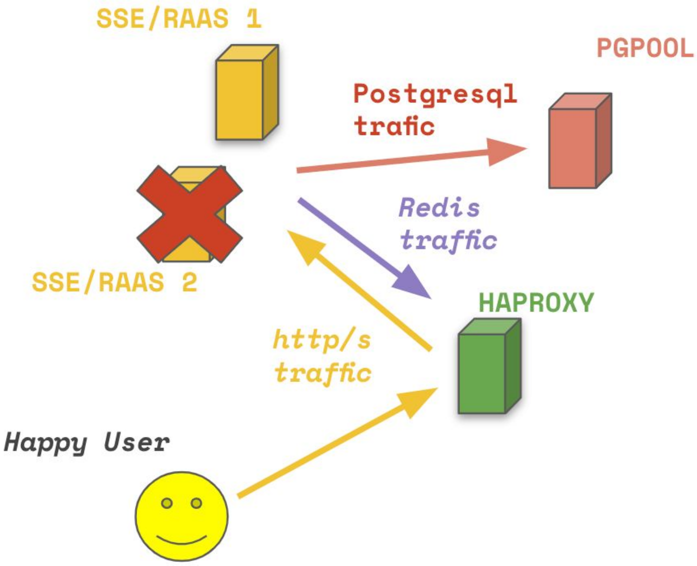
Step 6 - Survivability Tests

In this section we will test our configuration by simulating components failures

- 1- SSE server down
- 2- REDIS server down
- 3- POSTGRESQL server down

1- SSE server down
This simulates the failure of a RAAS servers.

EXPECTED RESULT:
SaltStack Enterprise console remains available via HAPROXY redirecting traffic to remaining SSE server



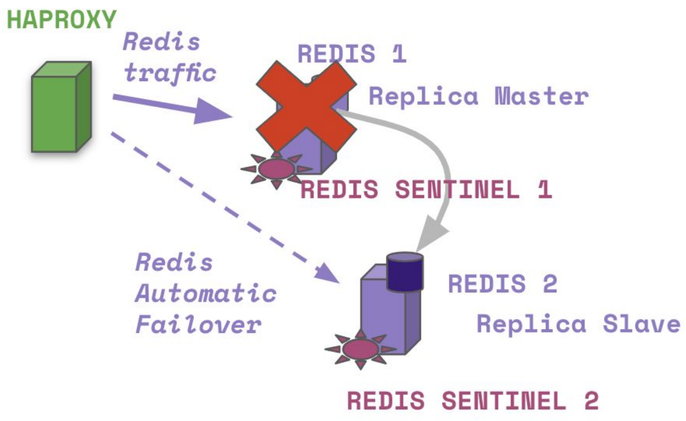
ACTIONS:

1. Stop raas service in sse-2 (or sse-1) server

```
$ systemctl stop raas
```

2. Check if SSE console remains up, connect to SSE via HAPROXY
   `https://ip_address_haproxy (public ip)`

2 - SSE server down
Simulate failure of REDIS MASTER server



EXPECTED RESULT:
Redis Sentinel to detect failure of Redis Master and trigger failover to slave replica server, converting it to Master.
HAPROXY to detect the change and route REDIS traffic to failover server
SSE to remain available up and running

ACTIONS:

1. Stop redis service in REDIS-1

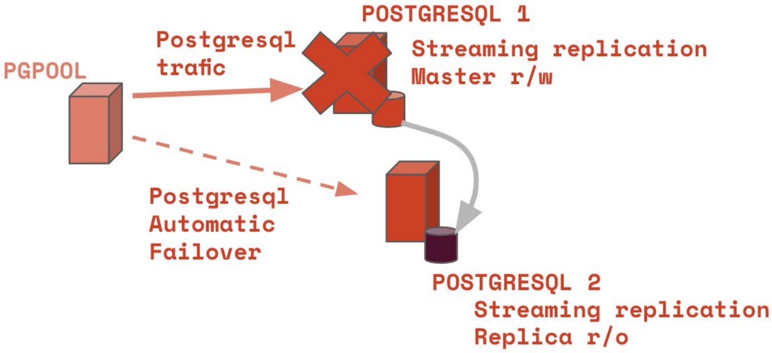
   ```
   $ systemctl stop redis
   ```

2. Check if SSE console remains up, connect to SSE via HAPROXY
   `https://ip_address_haproxy (public ip)`

TIPS:
Watch REDIS-2 log or MONITOR

3- POSTGRESQL server down
Simulate failure of POSTGRESQL server

EXPECTED RESULT:
PGPOOL to detect failure of POSTGRESQL server and trigger failover to stand by server, converting it to Master and directing traffic to this server. SSE to remain available up and running



ACTIONS:

1. Stop postgresql service in PSQL-1

```
$ systemctl stop postgresql-9.6
```

2. Check if SSE console remains up, connect to SSE via HAPROXY
   `https://ip_address_haproxy (public ip)`

TIPS:
Watch PSQL-2 log

```
[root@ip-10-0-5-150 ~]# tail -f /var/lib/pgsql/9.6/data/pg_log/postgresql-Wed.log
< 2019-10-30 17:37:21.124 UTC > LOG:  database system was interrupted; last known up at 2019-10-30 17:33:19 UTC
< 2019-10-30 17:37:21.134 UTC > LOG:  entering standby mode
< 2019-10-30 17:37:21.136 UTC > LOG:  redo starts at 0/2000028
< 2019-10-30 17:37:21.138 UTC > LOG:  consistent recovery state reached at 0/2000130
< 2019-10-30 17:37:21.138 UTC > LOG:  database system is ready to accept read only connections
< 2019-10-30 17:37:21.160 UTC > LOG:  started streaming WAL from primary at 0/3000000 on timeline 1
< 2019-10-30 18:53:43.549 UTC > LOG:  replication terminated by primary server
< 2019-10-30 18:53:43.549 UTC > DETAIL:  End of WAL reached on timeline 1 at 0/4000098.
< 2019-10-30 18:53:43.549 UTC > FATAL:  could not send end-of-streaming message to primary: no COPY in progress
< 2019-10-30 18:53:43.550 UTC > LOG:  record with incorrect prev-link 10000/6D90000
at 0/4000098
< 2019-10-30 18:53:43.608 UTC > FATAL:  could not connect to the primary server:
could not connect to server: Connection refused
                Is the server running on host "10.0.2.196" and accepting
                TCP/IP connections on port 5432?

< 2019-10-30 18:53:48.555 UTC > LOG:  trigger file found: /tmp/postgresql.trigger.5432
< 2019-10-30 18:53:48.556 UTC > LOG:  redo done at 0/4000028
< 2019-10-30 18:53:48.556 UTC > LOG:  last completed transaction was at log time
2019-10-30 18:50:01.421008+00
< 2019-10-30 18:53:48.558 UTC > LOG:  selected new timeline ID: 2
< 2019-10-30 18:53:48.609 UTC > LOG:  archive recovery complete
< 2019-10-30 18:53:48.614 UTC > LOG:  MultiXact member wraparound protections are
now enabled
< 2019-10-30 18:53:48.616 UTC > LOG:  database system is ready to accept
connections
< 2019-10-30 18:53:48.619 UTC > LOG:  autovacuum launcher started
```

TEST CONCLUSION:

If all went well, your SSE should have remained available up and running despite all the server failures.

HOMEWORK CHALLENGE:

Now that you know the manual steps to deploy the entire infrastructure, here you go a little challenge, grab the base state files used to build this lab environment and use your knowledge of Salt to improve each state to fully automate the installation of each component.

TIPS
1: Observe how pillar data is used in SSE installer files to provide configuration parameters for each service in your environment. Alternatively, you can use yaml map files or simply Jinja variables.
2: You can use execution of local states, similar to step 1 and step 3 and step 4 for Postgresql
3. Add requisites, and/or replace cmd.run commands by other Salt states to improve stateful execution
4. Build orchestration

State files download links:
https://tinyurl.com/w9az2wf **(UPDATED Nov19th: https://github.com/amalaguti/SC19)**

https://drive.google.com/file/d/10ILqOZ4Dniyawp_UrkdVV2UhqEr0egOD/view?usp=sharing

Email your instructor amalaguti@saltstack.com

-

THANK YOU

We hope that this session has been valuable to you and that you enjoy your time here at SaltConf19!