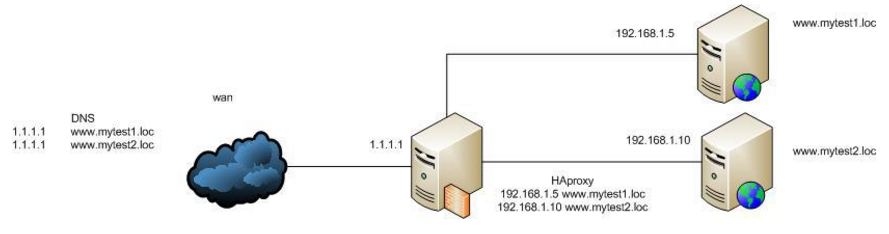
**Consul** - система визначення сервісів і розприділеного сховища ключ-значення. Використовується для взаємного визначення зв’язаних сервісів. Розповсюджується одним бінарним файлом, тому немає проблем з кросплатформенністю. Будь-який софт, що використовує Consul робить запити на локалхост. Використовує Gossip (pc-to-pc protocol) в якості протоколу обміну даними, що робить Consul швидким, відмовостійким. Для роботи з Consul не використовуються клієнтські бібліотеки, а вся робота йде чрез простий і зрозумілий HTTP REST API (близько 20 API функцій) або через service-definition.

Всі сервіси які можуть підтримувати самореєстрацію повинні використовувати відповідні бібліотеки ЯП. Окрім реєстрації сервіса потрібно грамотно написати скріпти перевірки доступності сервісів бо є ризик отримати зареєстрований в системі але недоступний сервіс.

Consul дозволяє відмовитися від конфігураційних файлів за рахунок SD (реєстрація сервісів і знаходження сервісів). Замість цього використовується KV-сховище для обміну конфігураційною інформацією.

**ConsulTemplate** - засіб для підключення до Consul систем які не підтримують SD, наприклад, HAProxy. Його задача - це відслідковувати реєстрацію\дереєстрацію сервісів і змінювати конфігурацію підлеглих сервісів

**HAProxy** використовують для балансування нагрузки на TCP-, HTTP-, HTTPS-додатки завдяки розприділенню вхідних запитів між "живими" обслуговуючими серверами (періодично перевіряє доступність серверів). Може також закріпляти конкретні ресурси за конкретними серверами. Стабільний, швидкий і ефективний в плані використання ресурсів пк. Має веб-інтерфейс.

Приклад конфігурації проксі:

frontend http\_frontend  
bind \*:80  
mode http  
option httpclose  
acl is\_mytest1 hdr\_end(host) -i mytest1.loc  
use\_backend mytest1\_web if is\_mytest1  
acl is\_mytest2 hdr\_end(host) -i mytest2.loc  
use\_backend mytest2\_web if is\_mytest2  
  
backend mytest1\_web  
mode http  
cookie SERVERID insert indirect nocache  
server mytestweb1 192.168.1.5:80 check cookie mytestweb1  
  
backend mytest2\_web  
mode http  
cookie SERVERID insert indirect nocache  
server mytestweb2 192.168.1.10:80 check cookie mytestweb2

Слухається **80** порт, розбираються вхідні запити. Якщо запитується **mytest1.loc** , то він попадає в  **access-list is\_mytest1**, в цьому випадку використовується **backend mytest1\_web**, в якому ми перенаправляємо трафік на внутрішній хост **192.168.1.5:80**, де у нас лежить сайт. Аналогічно на **mytest2.loc**

Команда **curl -** консольна утиліта для передачі даних використовуючи URL синтаксис.

**Ansible** - система віддаленого керування конфігураціями (її аналоги Puppet, Shef, Salt). Бере на себе всю роботу по приведенню віддалених серверів до потрібного стану. Їй потрібно тільки описати як це зробити за допомогою сценаріїв (утиліта ansible-playbook) які пишуться на YAML (\*.yml). Плейбуки ще можна розцінювати як набір вказівників на потрібні ролі. Таким чином можна швидко переконфігуровувати сервера добавляючи лише декілька нових рядків коду в сценарій. Плюс, на керовані вузли не потрібно встановлювати додаткове ПО, підключення виконується через SSH, а Python який потрібний ансіблу для роботи передвстановлений у всіх linux-like системах. В рідких випадках потрібні модулі можна викачати з офіційного репозиторія. Якщо треба написат исвій додатковий модуль, то його можна писати на будь-якій мові, головне щоб написаний модуль вмів приймати і віддавати JSON.

**Ролі** в Ансібл - це набір певних параметрів, тригерів, хендлерів і задач для одного чи декількох серверів. Використовуються, коли нам до сервера чи групі серверів потрібно застосувати типовий набір команд - тоді серверу назначається роль.

Графічна оболочка Ansible - **Ansibleworks AWX**.

**Jinja/Jinja2** - шаблонізатор для Python. Дозволяє налаштовувати теги, фільтри і глобальні змінні. Приклад:

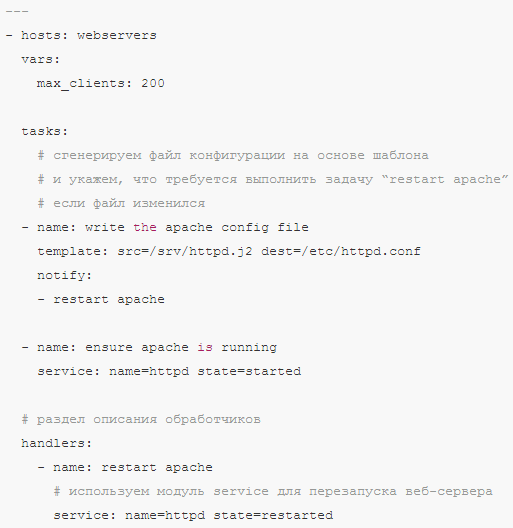
*# -\*- coding: utf-8 -\*-*

**from** **jinja2** **import** Template

template = Template('Hello {{ name }}!')

**print**(template.render(name=u'Вася')

Приклад сценарія з хендлером (вказується в notify) на YAML:



**Fiddler** — прокси, который работает с трафиком между Вашим компьютером и удаленным сервером, и позволяет инспектировать и менять его.

**Ендпоінт** - клас, який обробляє закінечення урла запиту. Приклад, аннотація над класом RequestMapping(“/health”) і методи в ньому помаркані також, наприклад RequestMapping(“/heartBeat”)

**Аутентифікація** ідентифікує юзера. Тобто хто він є.

**Авторизація** визначає до чого юзер має доступ.

**OAuth** – протокол авторизації.

Учасники OAuth: сервер з userdata, сервер авторизації, клієнт, який хоче отримати доступ до userdata.

**Polling** – це коли клієнт кожних 2-3 секунди запитує сервер чи з’явилися якісь нові дані для нього і бере їх якшо вони є. Тобо модель connect\disconnect.

**Long polling** – це коли клієнт відкриває коннект і висить на ньому очікуючи нових даних з сервера. Якщо дані прийшли, то робиться реконнект.

**HTTP Streaming** – це коли відправляється реквест, створюється об’єкт респонсу і він ніколи не закриваєть. Він і пушає дані клієнту.

**Server Side Events** – це коли реєструються server side listeners в браузері, а сервер в свою чергу відправляє івенти на клієнт.

**Web Socket** – Юзають для аплікух які динамічно міняюь якісь дані. Наприклад, контроль показників медичного обладнання, спортивні портали тощо.

Різниця між Web Scoket i HTTP:

* НТТР – stateless. Тобто він робить реквест, отримує респонс і закривається таким чином забуває що було в попередньому реквесті
* Request\response model – тобто ми можемо бігти тільки в один бік. Зробили реквест, чекаємо респонс. WebSecket є дуплексним і ним можна бігати в дві сторони.

ElasticSearch:   
Index in ES – the same as Database in RM.   
Type in ES – Tables in RM.  
Document in ES – Record in RM.  
Щоб створити новий індекс достатньо просто зробити запит до ES, а він створить новий індекс на льоту.

Масштабування ES:

Кластер може містити:

- master node (займається роутингом і делегуванням обробки запитів і отримання результату)

- data node (містить тільки дані і ніякої логіки)

-- може містити декілька шардів (shard) в яких знаходять різні типи даних. 1 індекс = 5 шардів по замовчуванню = 5 файлів на диску.

- search balancer node (роутінг тільки пошукових запитів)

Replica factor - це число яке значить скільки раз задубльовано інформацію з одного дата вузла.

Збережені дані не одразу доступні на читання. Є дефолтний таймаут 1 секунда. (як фінт використовують client.refresh() і дані обновляються зразу)

ES при індексації і пошуку використовує механізми аналізаторів і токенайзерів.

- аналізатор - перетворює вхідний текст в сукупність термінів по якому ES виконує пошук. (приклад реалізації аналалізатора Snowbal Analyzer)

- токенайзер

**Лямбда** - анонімна функція, яка не прив’язана до конкретного класу.

Лямбда буде мати тип і приведена до типу взятому з контексту її використання.

Основні бенефіти лямбд і стрімів - менше коду, відповідно важче зробити помилку, краща читабельність коду,