Puppet

Çağrı Emer

Nisan, 2016

İçindekiler

1 Giriş

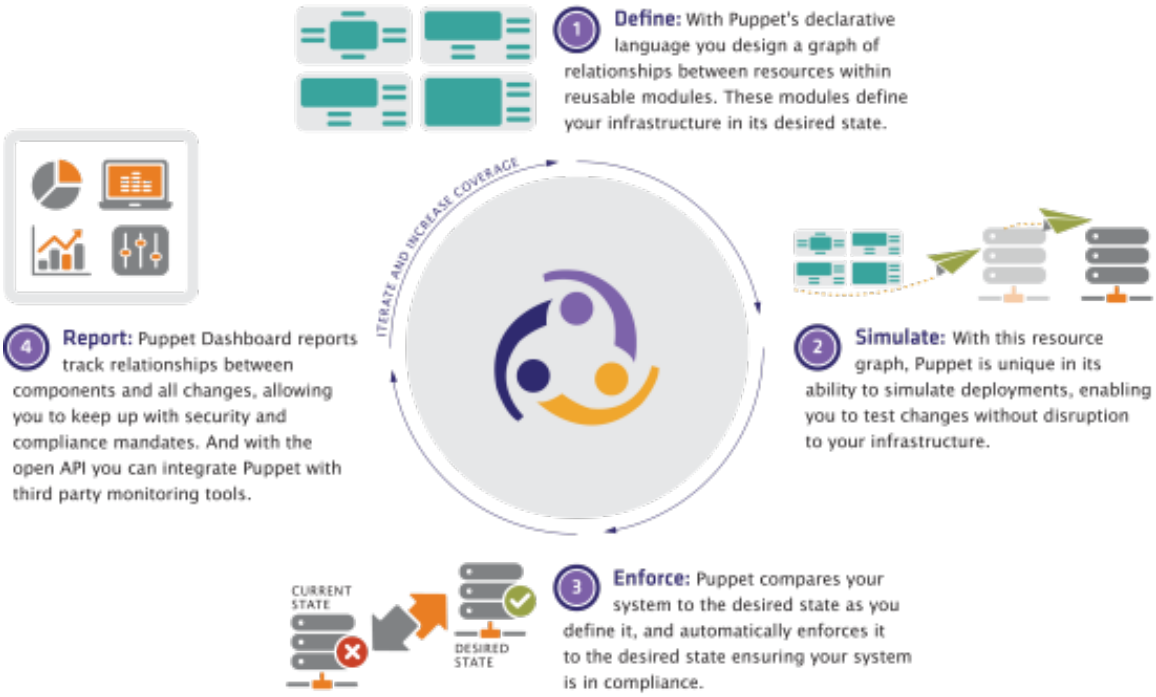
Puppet hem ticari hem de açık kaynaklı bir yazılım olarak sunulmaktadır. Bu ikisi arasındaki temel farklar şunlardır: Açık kaynak versiyon, grafik kullanıcı arayüzü ile gelmez. VMWare sanal makinelerinin provizyonunu yapamaz. Keşif ve Kopyalama [1] özelliği yoktur. Kullanıcı hesapları tanımlanamaz. Görev otomasyonu yapılamaz. Denetleme ve standartlara uyum için bir aracı yoktur. RBAC [2] desteği yoktur. Bütün bileşenlerini içeren platformun bağımsız tek bir yükleyicisi yoktur. Servis sözleşmesi ve destek yoktur. Puppet Labs mühendisleri tarafından test edilmemiştir. Ticari versiyondan bütün bu eksiklerine rağmen açık kaynak Puppet yeterince güçlü bir araçtır.

[1] *Sunuculardaki kaynakları otomatik olarak bulabilir ve yapılandırmaları kopyalayabilir.*
[2] *Role Başed Access Control (RBAC) değişik yetkilere sahip kullanıcıların tanımlanmasına izin verir.*

2 Nasıl Çalışır?

Puppet IT otomasyonunda model tabanlı, tanımlayıcı yaklaşımı kullanır.

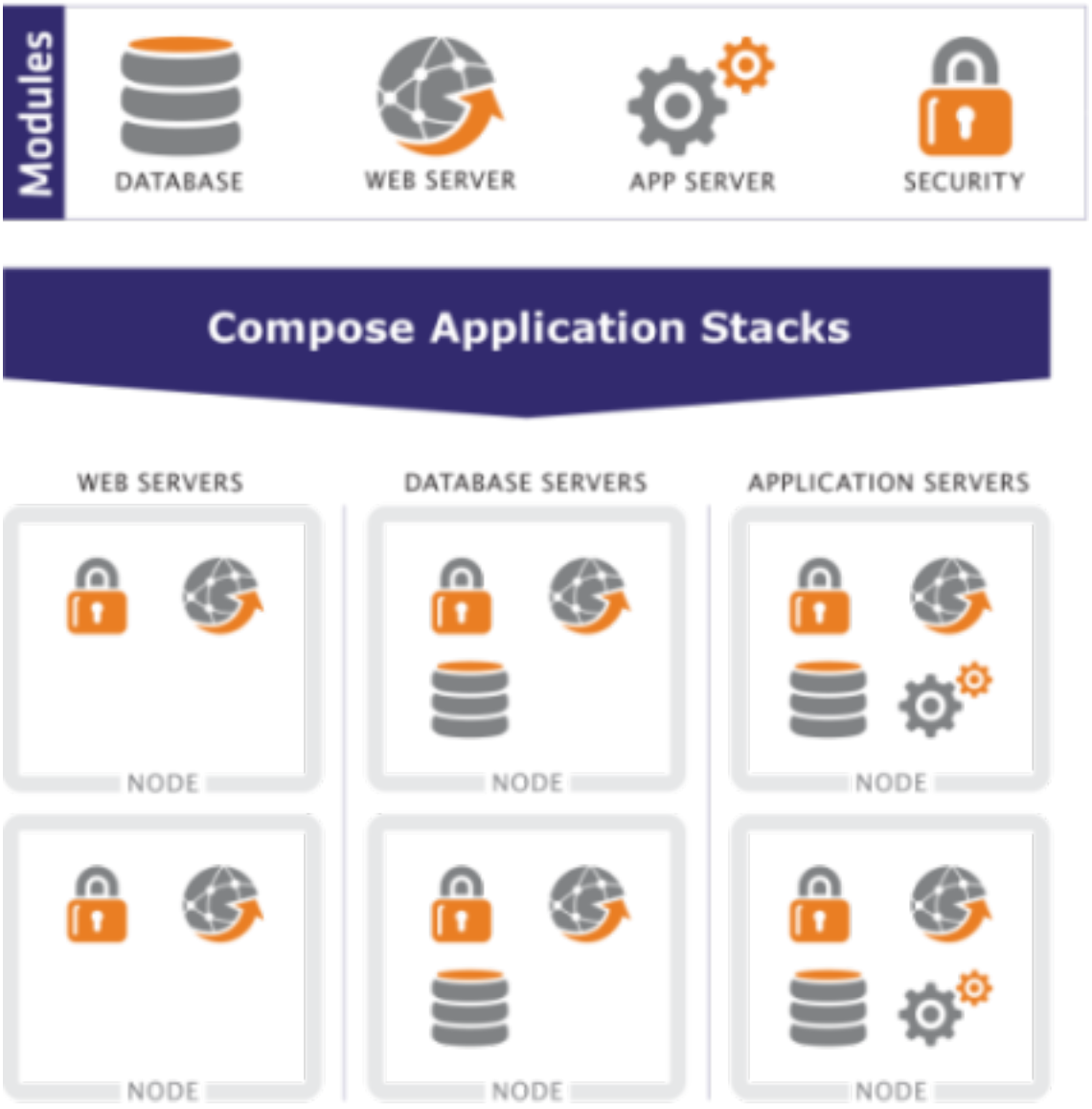
- **Tanımla:** Altyapının arzu edilen yapılandırmasını Puppet'ın yapılandırma dili ile tanımla.
- **Simüle et:** Yapılandırma değişikliklerini uygulanmadan önce simüle et.
- **Uygula:** Otomatik olarak istenen yapılandırmayı uygula böylelikle yapılandırmadaki küçük değişikliklerin önüne geç.
- **Raporla:** İstenen ve mevcut durum arasındaki farkları ve istenen duruma getirilmek için uygulanan değişiklikleri rapor et.



Şekil 1:

3 Yapılandırma ayarları nerede bulunur?

Puppet'i geliştiren firma olan Puppet Labs'in [3] PuppetForge [4] adını verdiği sistem 600'den fazla hazır yapılandırma modülü içermektedir. Buna ek olarak eğer özel ihtiyaçlarınız varsa Puppet'in yapılandırma dilinde kendi modülünüzü de yazmanıza olanak tanımaktadır. Bir kere yazıldıktan sonra bu modülleri ister fiziksel ister sanal sunucularda, isterseniz de bulut sunucu hizmet sağlayıcılarında kullanabilirsiniz. Dahası, ortak ayarlar içeren bütün bir uygulama yapılandırmasını değişik yapılandırma modüllerini birleştirerek oluşturabilirsiniz de.



Şekil 2:

[3] http://puppetlabs.com
[4] http://forge.puppetlabs.com

4 Elleri Kirletme Vakti

Bu yazıda basitçe sunucu/istemci modelinde çalışmaktansa sunucuya ihtiyaç duymayan Puppet manifestoları nasıl yazılır anlatmaya ve böylelikle Puppet'a genel bir giriş yapmaya çalışacağım. Kaynaklar ve RAL (kaynak soyutlama katmanı) nedir tanımlayarak başlayalım. Bir sistem hayal edin. Bu sistemdeki dosyalar, kullanıcılar, paketler, çalışan servisler, zamanlanmış görevler hatta tek bir kabuk komutu bile kaynak olarak tanımlanır Puppet gözlükleri ile baktığımızda. Her kaynağın bazı özellikleri vardır. Örneğin her dosya bir dosya yoluna ve dosya sahibine, her kullanıcı bir isime, bir kullanıcı numarasına ve bir gruba sahiptir. Bu da demek oluyor ki benzer kaynaklar bazı türler/modeller altında toplanabilir. Daha da ötesi, işletim sistemlerinin bu kaynakların kullanımını/yönetimini implemente etme şekli farklı olsa dahi konsept olarak aynı işi yaparlar. Örneğin, yum ınstall httpd ve apt-get ınstall apache2 komutlarının ikisi de Apache web sunucusunu kurma işini yaparlar. Fakat bu işi yapma şekilleri değişiktir. İşte az önce değinilen implementasyon farkı budur. Puppet'in RAL'ını oluşturan iki temel işte bu türler ve platforma özgü işlemlerdir. RAL sayesinde kaynakları işletim sisteminin ne olduğundan bağımsız şekilde yönetebilir ve kullanabiliriz.

5 Bir Kaynağın Anatomisi

Puppet'ta her kaynak, bu kaynağı sağlayan türün bir örneğidir ve "title" ile nitelenir. Türün belirlediği özelliklere "attribute" sahiptir ve bu özelliklerin bir değeri "value" vardır. Puppet dili bir kaynağı şu şekilde ifade eder:

```
1 user { 'sudodergi':
2     ensure       => present,
3     shell        => '/bin/bash',
4     home         => '/home/sudodergi',
5     managehome   => true,
6     }
```

Puppet, puppet resource adı verilen bir araç ile gelir. RAL'ı kullanan bu kabuk sayesinde sisteminiz hakkında bilgi edinebilir ve değişiklikler yapabilirsiniz. Puppet resource aracının ilk argümanı türdür. Eğer başka bir argüman almadan şu komut çalıştırılırsa;

```
1 puppet resource user
```

sistem user türündeki kaynaklar için sorgulanacak ve o türe ait olan tüm bilgiler gösterilecektir. İkinci argümanı ekleyerek spesifik bir kaynağı sorgulamak mümkündür.

```
1 puppet resource user www-data
```

Örneğin yukarıdaki sorgu benim bilgisayarımda şu sonucu döndürmektedir;

```
1 user { 'www-data':
2   ensure  => 'present',
3   comment => 'www-data',
4   gid     => '33',
5   home    => '/var/www',
6   shell   => '/bin/sh',
7   uid     => '33',
8 }
```

Eğer puppet resource'a verilen türe ait bir özellik ve o özelliğe ait değer argüman olarak verilirse o kaynağın o özelliği değiştirilecek ya da yaratılacaktır. Örneğin sisteme üstteki sudodergi kullanıcısı eklemek için şöyle bir komut verilebilir.

```
1 puppet resource user sudodergi ensure=present shell="/bin/bash" home="/home/sudodergi"
      managehome=true
```

6 Çekirdek Türler

Puppet'in birkaç öntanımlı türü vardır ve yeni türler dağıtılan modüllere eklenebilir. İlk aşamada aşina olunacak türler şunlardır: "notify", "file", "package", "service", "exec", "cron", "user" ve "group". Puppet belgelendirmesi bu türler için bir de "kopya kâğıdı"na sahiptir.[5] Bu kopya kâğıdı sayesinde türleri ve özelliklerini ezberlemenize gerek kalmayacaktır. Derinlemesine bilgi almak için yine tür belgelendirmesine bakılabilir.[6] Puppet manifestoları yazarken dilinizin ucunda olan fakat çıkaramadığınız özellikleri de puppet describe -s komutu yardımıyla uçbirimden hızlıca kontrol edebilirsiniz. puppet help komutu da yardım alabileceğiniz başka bir komuttur.

[5] http://docs.puppetlabs.com/puppet_core_types_cheatsheet.pdf
[6] http://docs.puppetlabs.com/references/stable/type.html

7 Manifestolar

Puppet programlarına “manifesto” adı verilir ve .pp dosya uzantısını kullanırlar. Yazılmış bir manifestoyu şu şekilde sisteminize uygulayabilirsiniz:

```
puppet apply cok_guzel_dosyalar_yarat.pp
```

Şimdi bu isminden ne yapacağı hiç anlaşılmayan çok_güzel_dosyalar_yarat manifestosunu yazalım.

```
file {'/tmp/cok':
      ensure  => present,
      content => "Merhaba",
    }

    file {'/tmp/guzel':
      ensure => directory,
      mode   => 0644,
    }

    file {'/tmp/dosyadedimamalinkbu':
      ensure => link,
      target => '/tmp/cok',
    }
```

Eğer bu manifestoyu çalıştırırsanız tmp dizininde çok dosyası, güzel dizini ve dosyadedimamalinkbu isimli çok dosyasını gösteren bir link oluşturulacaktır.

Bütün türlerin özelliklerine daha önce de söylediğim üzere puppet describe -s ile ulaşmak mümkün. Ardından bu özellikleri [7]'dan inceleyip ne yapacağını çıkarttığınızda Puppet manifestolarınızı rahatlıkla yazabilirsiniz.

Puppet'a giriş yapmaya çalıştığım bu yazı umarım aklınızdaki soru işaretlerini yanıtlamanıza yardımcı olmuştur.

[7] http://docs.puppetlabs.com/references/stable/type.html

8 Kaynak: