Infraestrutura como Código

Outras Ferramentas

Michel Vasconcelos
michel.vasconcelos@gmail.com

Puppet

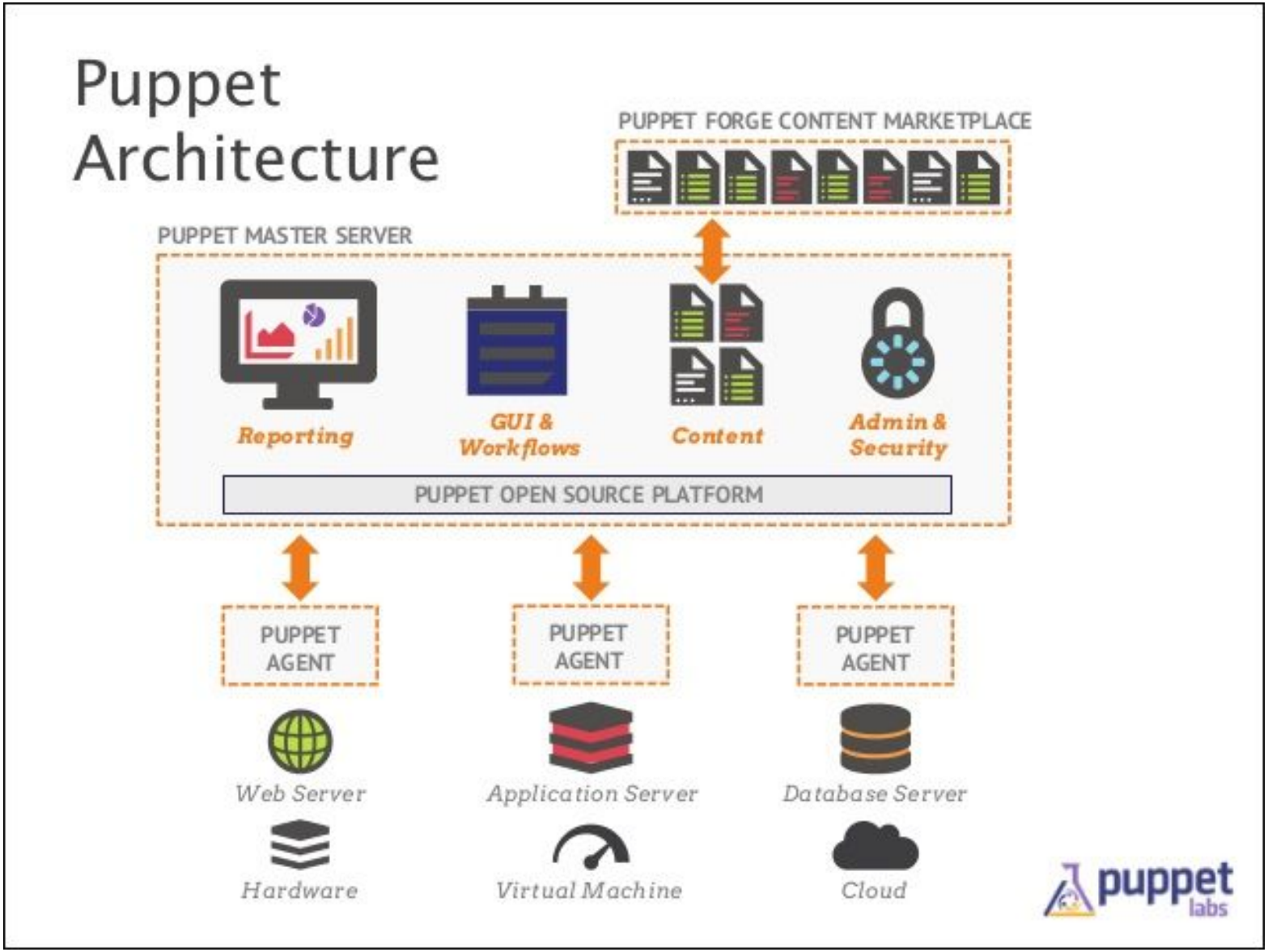
Puppet

- Gestão de Configuração e automação de infraestrutura
- Ruby + DSL
- Declarativo
- Idempotência e Consistência
- Master vs Standalone

Principais Elementos

- Puppet Master
- Agentes
- Puppet DB
- Recursos
- Manifesto
- Módulo
- Catálogo
- Fatos

Arquitetura (Master)



Funcionamento

- Todo o modelo de trabalho se baseia na declaração de recursos

- Vários recursos podem ser agrupados em classes que, por sua vez, são agrupados em manifestos
  - Módulos são reusáveis
  - Puppet Forge

- Manifestos são os arquivos de programação do puppet
  - DSL Ruby

Manifesto

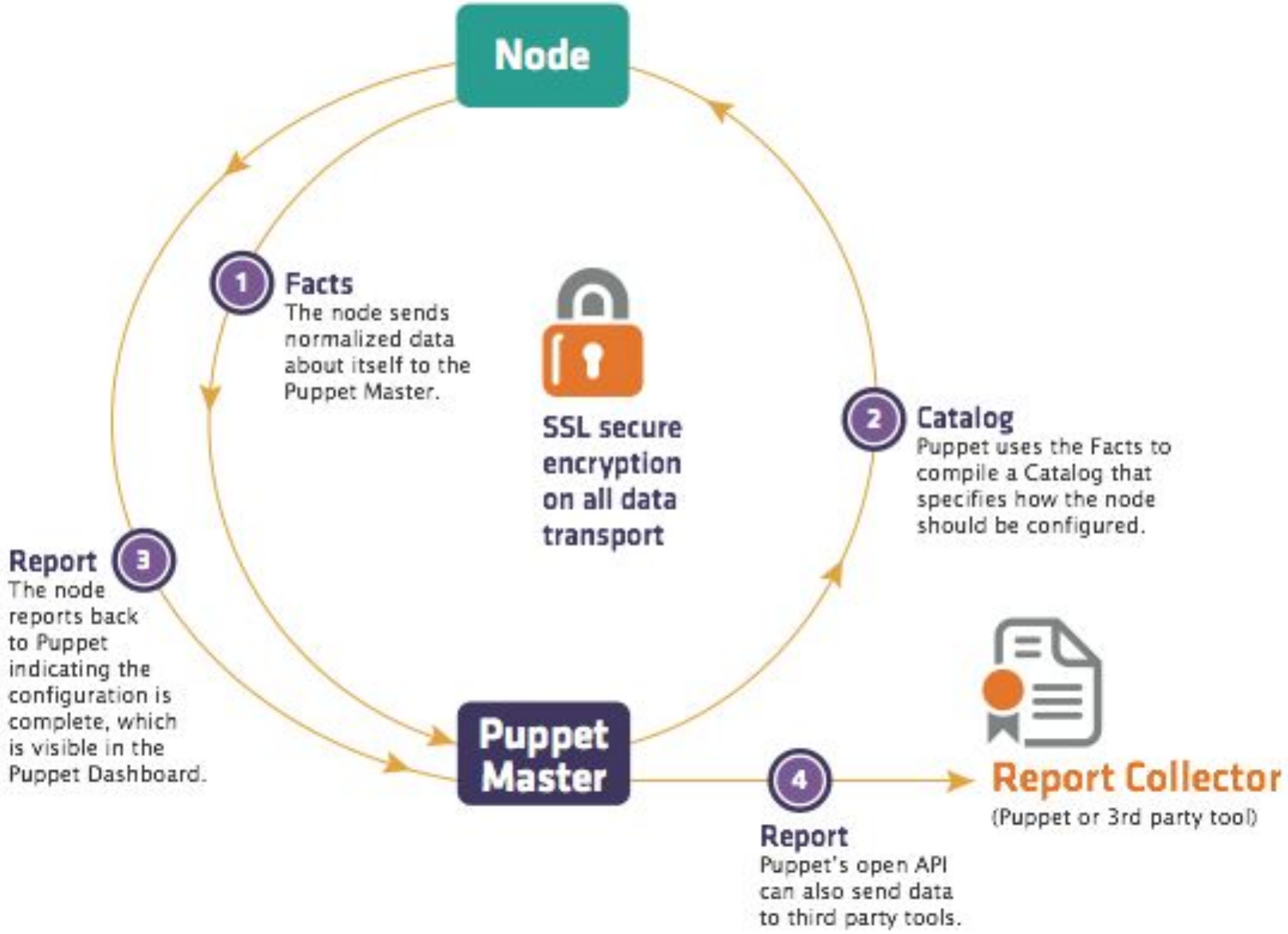
```
case $operatingsystem {
  centos, redhat: { $service_name = 'ntpd' }
  debian, ubuntu: { $service_name = 'ntp' }
}

package { 'ntp':
  ensure => installed,
}

service { 'ntp':
  name      => $service_name,
  ensure    => running,
  enable    => true,
  subscribe => File['ntp.conf'],
}

file { 'ntp.conf':
  path    => '/etc/ntp.conf',
  ensure  => file,
  require => Package['ntp'],
  source  => "puppet:///modules/ntp/ntp.conf",
  # This source file would be located on the Puppet master at
  # /etc/puppetlabs/code/modules/ntp/files/ntp.conf
}
```

Funcionamento

- **Módulo** é a principal unidade de agrupamento do Puppet. É composto por um arquivo de manifesto principal e um conjunto de diretórios

- <MODULE NAME>
- manifests
- files
- templates
- lib
    - facter
    - puppet
        - functions
        - parser/functions
        - type
        - provider
- facts.d
- examples
- spec
- functions
- types
- tasks

Funcionamento



Standalone mode

- Cada nó contém todas as informações necessárias para compilação e aplicação do catálogo

- O processo de aplicação pode ser sob demanda or por meio de job

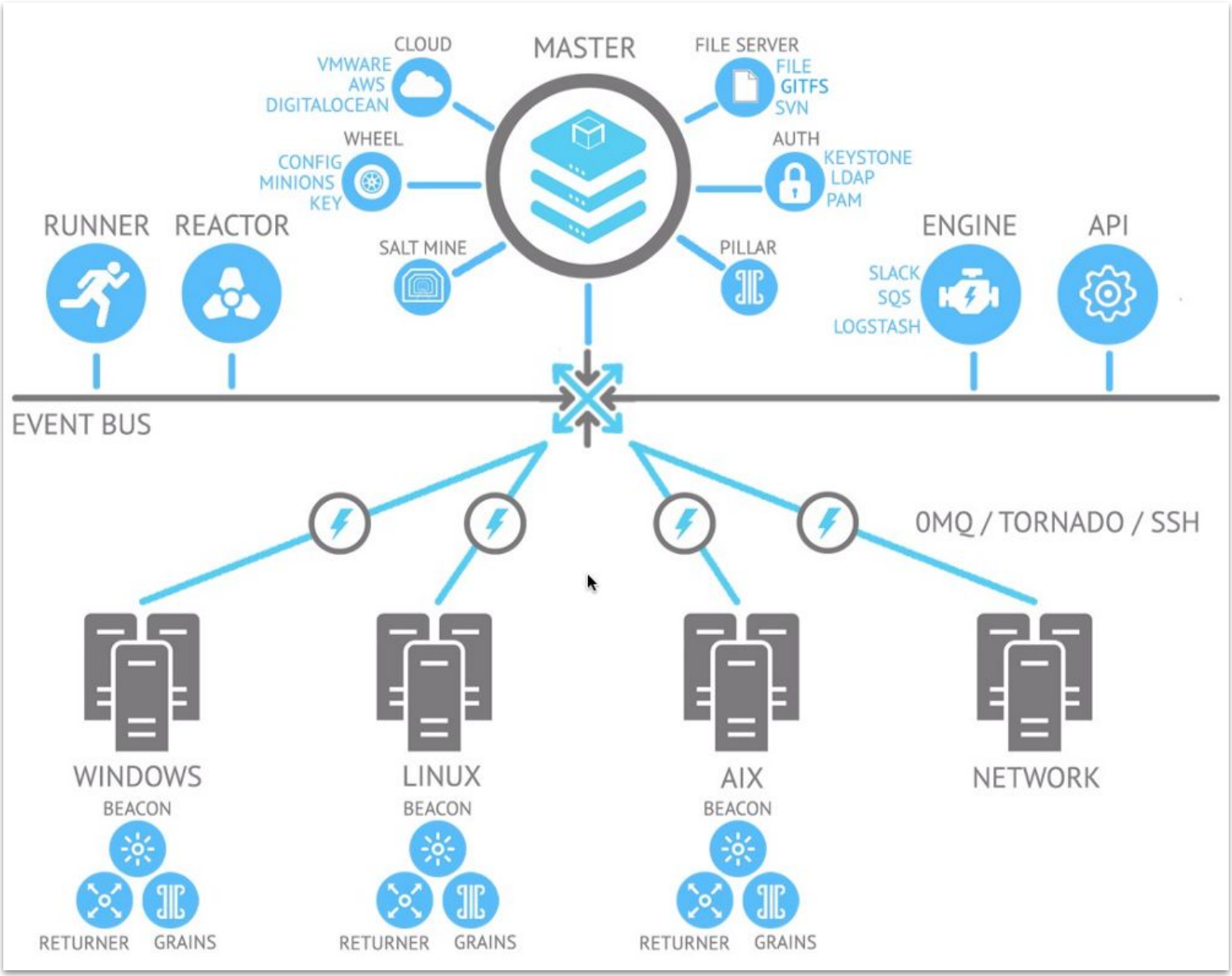
- O relatório de aplicação é salvo em disco

SaltStack

Características

- Modelo Master e Agentes
  - Minions
- Modelo declarativo
- Arquivo de definição de configuração em YAML
  - states
- Deployment em larga escala

Arquitetura



Principais conceitos

- Master e Minions
- Grains
- State Files

Comparativo

	Chef	Puppet	Ansible	SaltStack	CloudFormation	Terraform
Code	Open source	Open source	Open source	Open source	Closed source	Open source
Cloud	All	All	All	All	AWS only	All
Type	Config Mgmt	Config Mgmt	Config Mgmt	Config Mgmt	Orchestration	Orchestration
Infrastructure	Mutable	Mutable	Mutable	Mutable	Immutable	Immutable
Language	Procedural	Declarative	Procedural	Declarative	Declarative	Declarative
Architecture	Client/Server	Client/Server	Client-Only	Client/Server	Client-Only	Client-Only

Comparativo

- A divisão entre provisionamento e configuração está cada vez mais nebulosa
  - Alguns chamam o provisionamento de *deploy* (de infra)

- Chef e Puppet são (de longe) as ferramentas mais usadas e com maior estabilidade
  - Ansible correndo atrás

- Se você pensa em flexibilidade, fique de olho no SaltStack (longo prazo)

Comparativo

- Todos estão correndo para ter um servidor que forneça governança
  - Chef e Puppet na frente
  - Ansible Tower ainda não é tão amigável quanto os outros
  - Há ferramentas Open Source
    - Foreman
    - Semaphore
    - etc

Considerações Finais

- Qual a importância de se ter CM?

- Provisionamento vs Configuração

- E o Deploy?!
  - Octopus
  - Urban Code
  - Electric Flow
  - GoCD
  - etc

Trabalho

Utilizando as ferramentas Terraform, Ansible e /ou Chef, provisione e configure uma infraestrutura para aplicações web contendo:

- 2 servidores apache para fornecer as paǵinas estáticas
  - Deve estar configurado com uma página inicial (index.html) sob a url <servidor-web>/espec-iac/index.html
- 2 servidores tomcat
  - Respondendo na porta 9090
- 1 Servidor de Banco de Dados Postgresql
  - Respondendo na porta 5433

Trabalho

O trabalho será entregue na forma de um repositório git e na sua raiz deverá haver um arquivo README.md descrevendo os artefatos do repositório e como eles devem ser utilizados. O arquivo também deverá conter os integrantes da equipe.