Infrastructure as Code

MICROSERVICES

Content

//WEEK 1 - Tobias

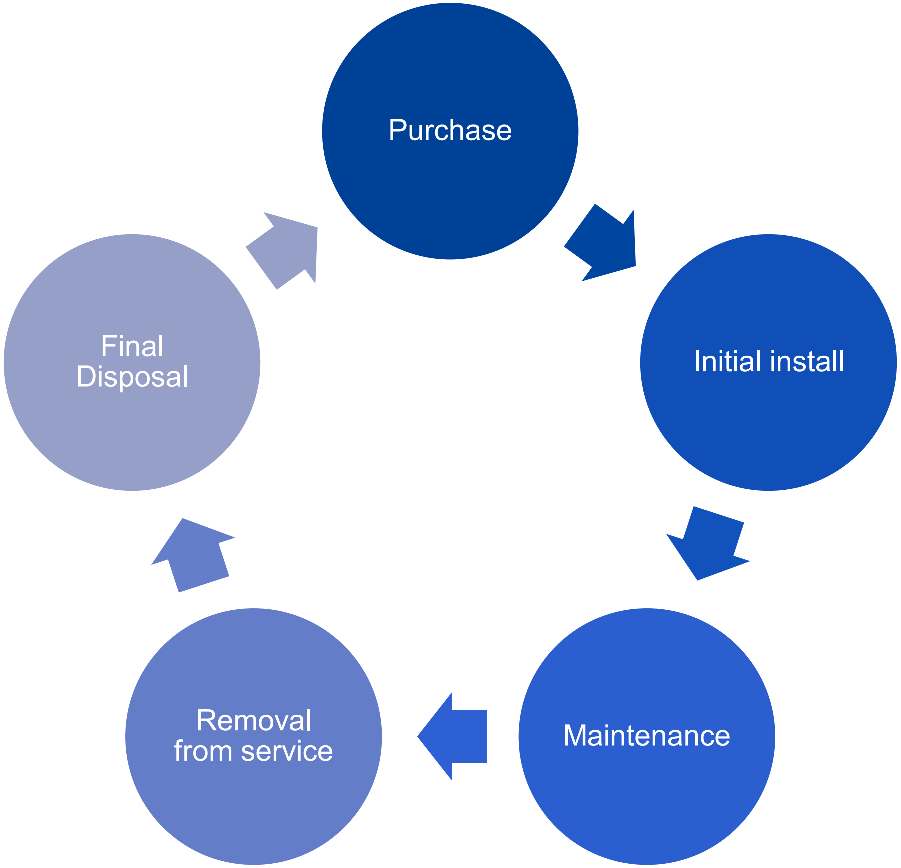
- DevOps Introduction
- Pipeline (continuous integration / continuous delivery / continuous deployment)
- Real World Examples

//WEEK 2 - Peter

- Infrastructure as Code
- Puppet, Chef, Ansible, Salt, …
- DNS as Code

History

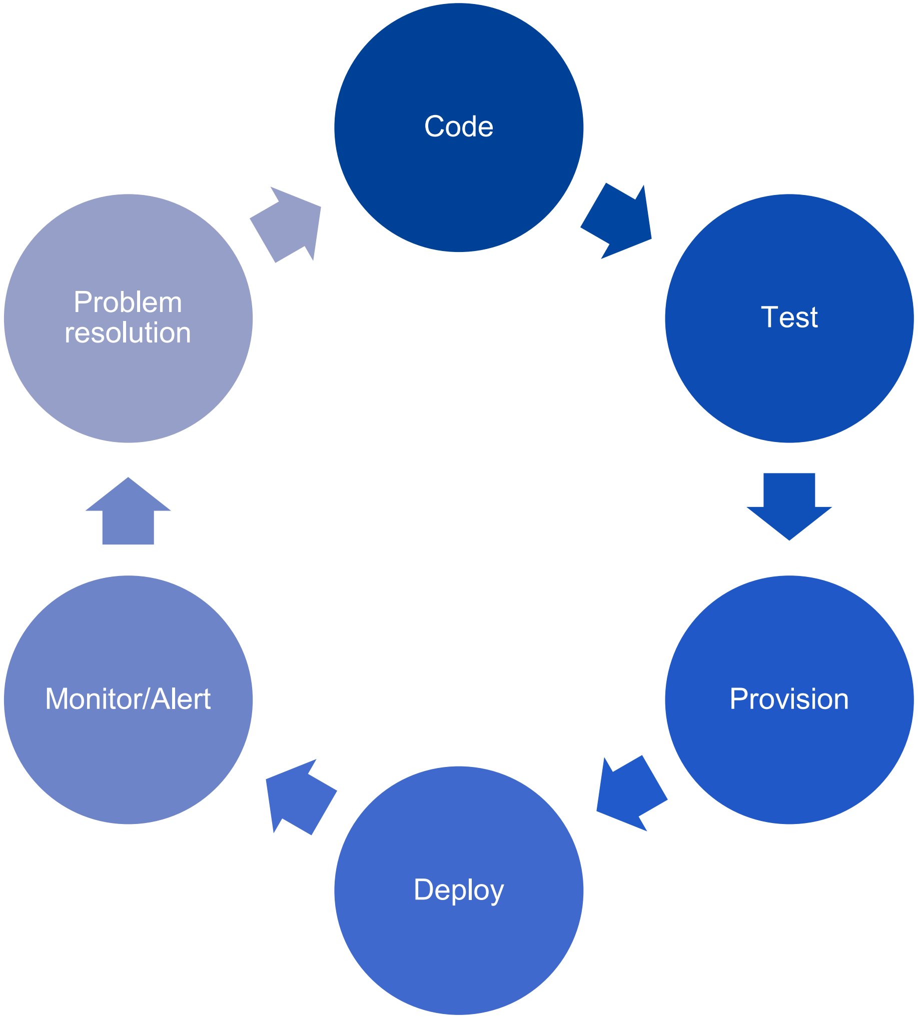
- In the past, administrators have taken care of each server for its entire lifecycle
- Every server was kind a „piece of art“
- Every server hosted a large number of services
- To be able to restore a server, administrators created full backups of every server (e.g. the /etc directory of Linux servers)



Introduction

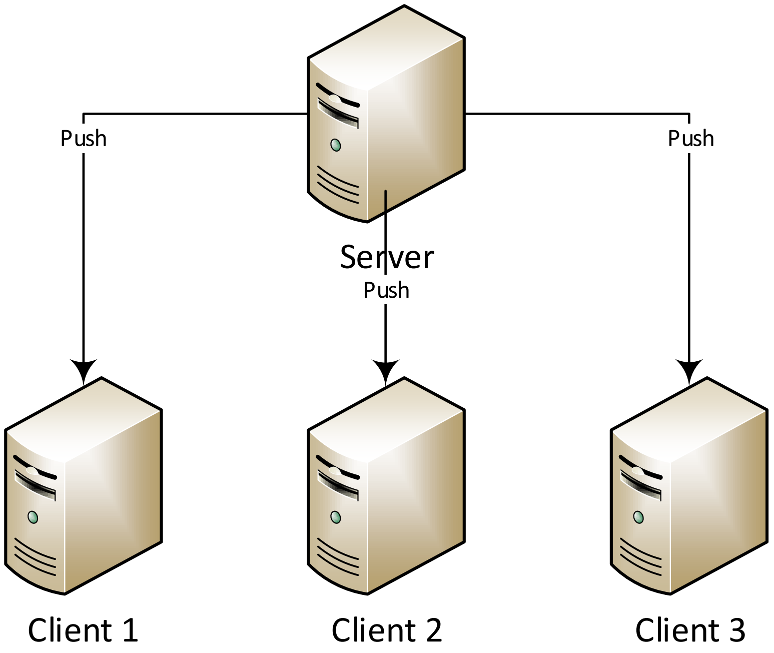
- Define the configuration of your whole infrastructure as code
- Whenever you don’t need a server any more delete it and restore it if needed (only data backup and code is required)
- It’s easy (and required) to put all your infrastructure code into a version control system
  - Test your infrastructure code as you test your program code!
  - New servers can be bootstrapped full-/ or semi-automatic
- It doesn’t matter if you’re building a Docker container or if you’re installing a virtual or physical server – infrastructure code may be applied to all of them
- Focused on managing a large number of servers (instead of building a container farm)

Workflow

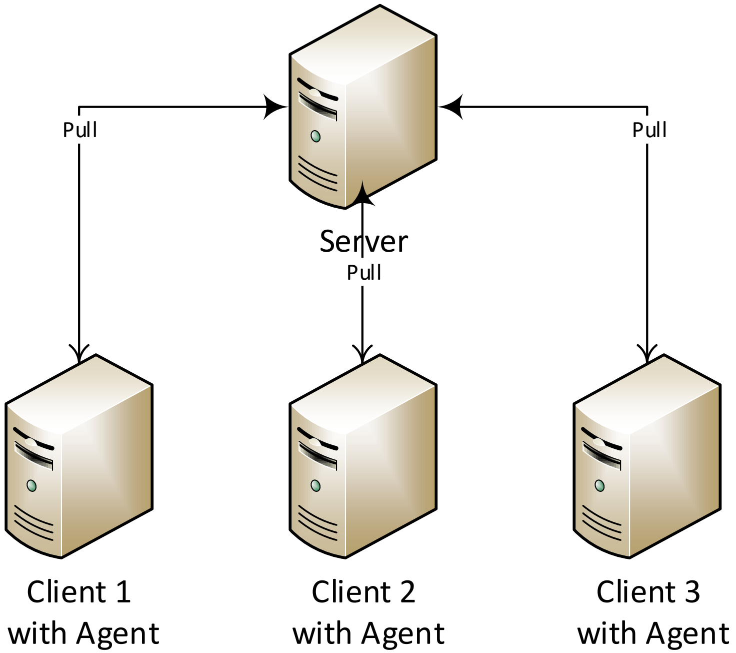

Code
Test
Provision
Deploy
Monitor/Alert
Problem resolution

Architecture

Server only



Client – Server



Fundamentals

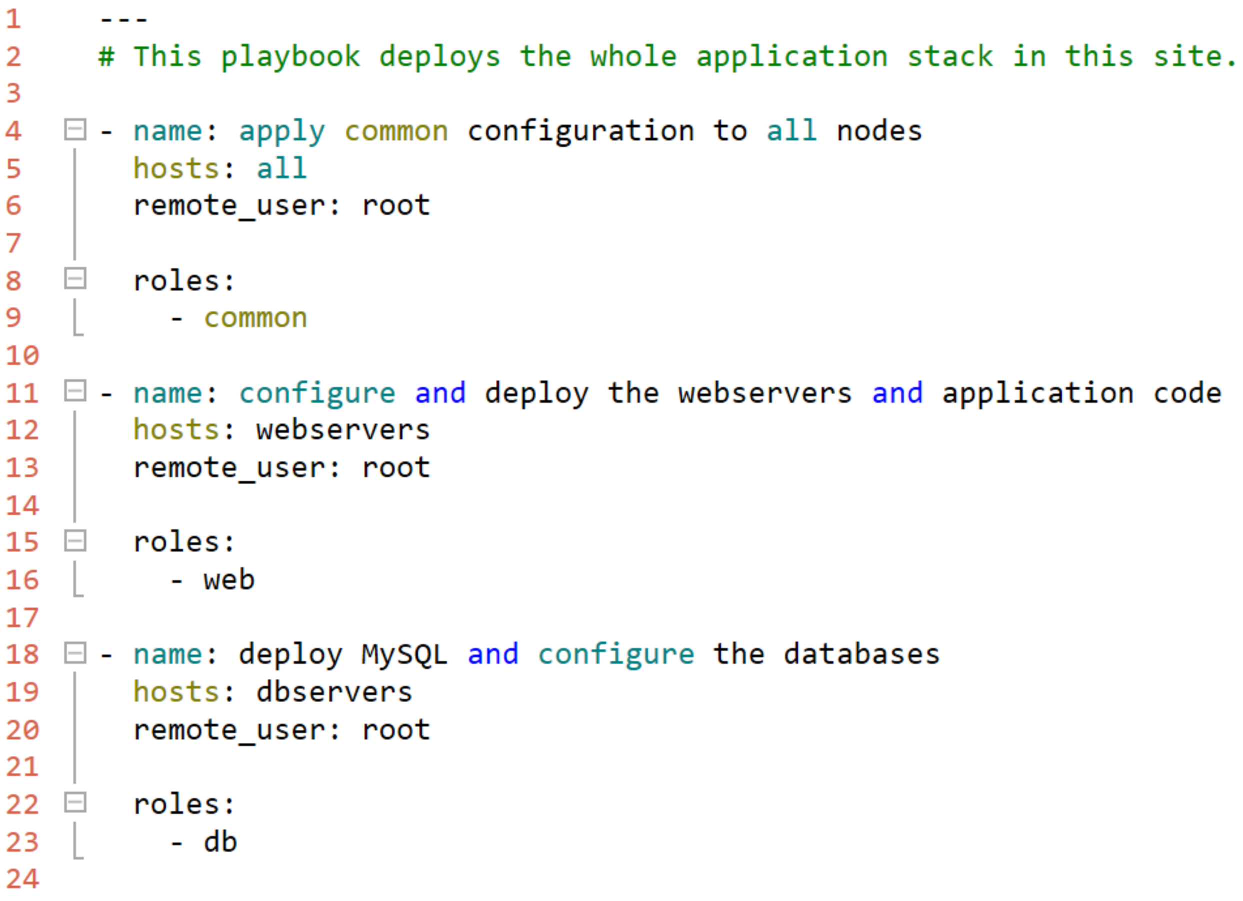
- Server or agent collects facts about target system
- Configuration may be applied multiple times leading to the same result (means no change if not necessary)
- Process of applying configuration may be forced
- Configuration describes a desired state of a machine (configuration files, installed packages, existence of users, running services, …)
- Most of the systems abstract the concrete operating system (e.g. the concrete package manager)
- Most of the systems are resource orientated to describe:
  - Packages
  - Files
  - Users/Groups
  - Services

Ansible

- Open-source and commercial editions available
- Available for Linux/Unix systems with Python 2.7 environment
- Server oriented – no agent available, changes are executed through SSH commands
- Configuration (called Playbooks) are written in YAML files
- Modules can be created in every language which can create JSON
- Information about the target system are collected by the setup module (or Facter or Ohai)
- Ansible Galaxy as central module registry
- Modules are created by the Ansible company and the community

https://www.ansible.com/

Ansible Sample



```yaml
---
# This playbook deploys the whole application stack in this site.

- name: apply common configuration to all nodes
  hosts: all
  remote_user: root

  roles:
    - common

- name: configure and deploy the webservers and application code
  hosts: webservers
  remote_user: root

  roles:
    - web

- name: deploy MySQL and configure the databases
  hosts: dbservers
  remote_user: root

  roles:
    - db

```

Ansible – known customers

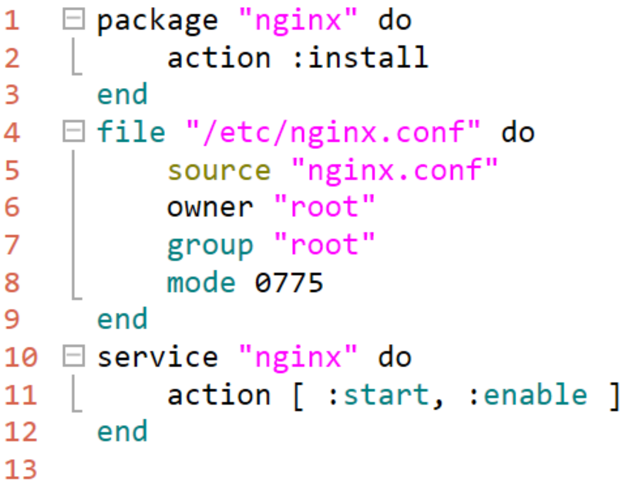
- Atlassian
- Cisco
- EA Sports
- Evernote
- Gartner
- GoPro
- Juniper
- NASA
- NEC
- Twitter

Chef

- Open-source and commercial editions available
- Available for most common Linux Systems, OS X, AIX and Windows
- Server oriented – standalone mode available
- Configurations are written in Ruby
- Recipes are written in Ruby
- Central Chef Supermarket for recipes

https://www.chef.io/chef/

Chef Sample



```ruby
package "nginx" do
    action :install
end
file "/etc/nginx.conf" do
    source "nginx.conf"
    owner "root"
    group "root"
    mode 0775
end
service "nginx" do
    action [ :start, :enable ]
end
```

Chef – known customers

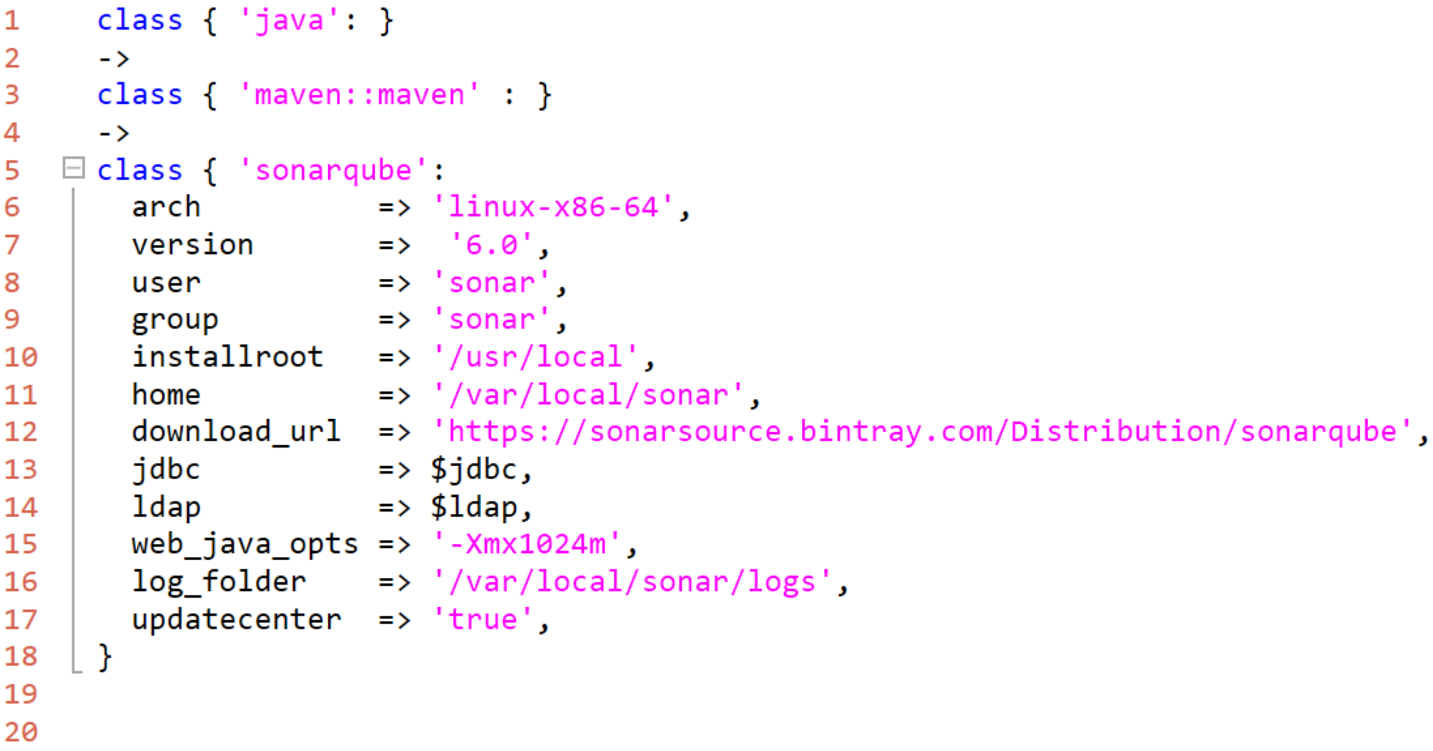
- Bloomberg
- Facebook
- HP Enterprise
- IBM
- Microsoft
- Prezi
- Yahoo
- …

Puppet

- Open-source and commercial editions available
- Available for most Linux systems, few Unix systems and Windows Server
- Client – Server oriented, standalone mode available
- Configuration (called Manifests) are written in custom DSL
- Modules are written in Ruby
- Information about systems are collected by a component called “Facter”
- Central module registry maintained by vendor Puppet Labs
- Modules are created by Puppet Labs and the community
- …

https://puppet.com

Puppet Samples



```
class { 'java': }
->
class { 'maven::maven' : }
->
class { 'sonarqube':
  arch          => 'linux-x86-64',
  version       =>  '6.0',
  user          => 'sonar',
  group         => 'sonar',
  installroot   => '/usr/local',
  home          => '/var/local/sonar',
  download_url  => 'https://sonarsource.bintray.com/Distribution/sonarqube',
  jdbc          => $jdbc,
  ldap          => $ldap,
  web_java_opts => '-Xmx1024m',
  log_folder    => '/var/local/sonar/logs',
  updatecenter  => 'true',
}
```

Puppet – known customers

- at&t
- Citrix
- EMC
- GitHub
- HP
- Jenkins
- NetApp
- Sun
- …

Product overview

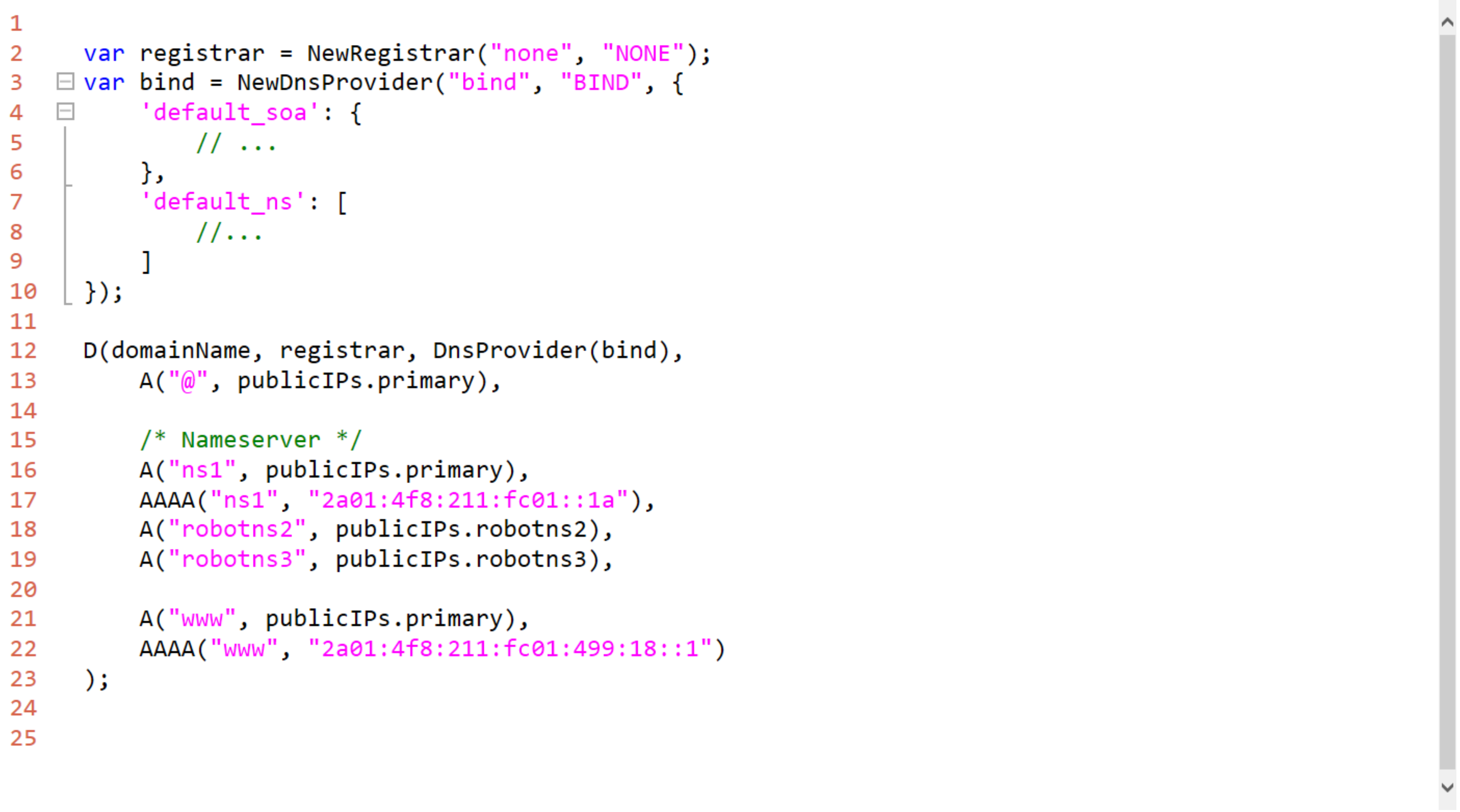
Name	Ansible	CFEngine	Chef	Machinery	Puppet	Salt
Vendor	Ansible HQ	CFEngine	Chef	SUSE	Puppet Labs	SaltStack
Release year	2012	1993	2008	2014	2005	2011
Operating system	Unix/Linux Python 2.7	SLES, Debian, CentOS, Windows	Debian, RHEL, Ubuntu, OSX	openSUSE, SLES, Fedora, CentOS, Ubuntu, Debian	Debian, Fedora, Ubuntu, CentOS, OSX, RHEL,…	Unix/Linux Python 2.6
Architecture	Server only	Client – server	Chef server, Chef local	CLI	Client – server, standalone	Client – server
Data model	Hosts, Playbooks, Roles, Modules, Attribute	Bundles, Policies, Promises	Environments, Nodes, Roles, Cookbooks, Attributes	-	Variables, Parameter, Manifests, Classes, Hiera	Pillars, States, Formulas, Modules
Inventory	Local only	Server	Local	Server	Local	Local

DNS as Code

- Solution to manage DNS entries in an independant format
- Enables administrators (or so called devops) to test the DNS settings and entries through continuous integration
- Adapters available for:
  - Active Directory
  - BIND
  - CloudFlare
  - Google
  - …

https://github.com/StackExchange/dnscontrol

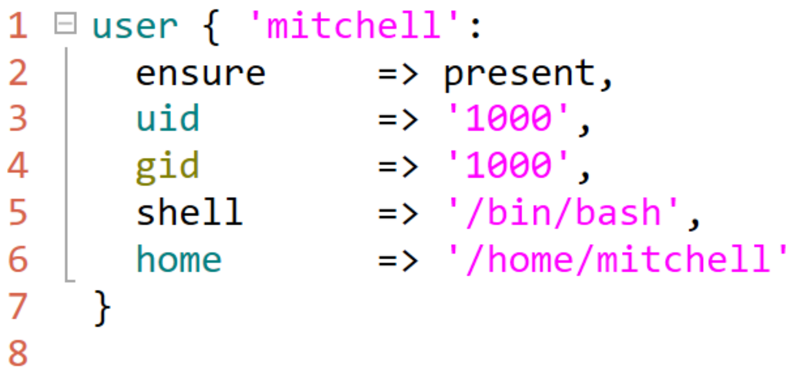
DNSControl – Sample



```

var registrar = NewRegistrar("none", "NONE");
var bind = NewDnsProvider("bind", "BIND", {
    'default_soa': {
        // ...
    },
    'default_ns': [
        //...
    ]
});

D(domainName, registrar, DnsProvider(bind),
    A("@", publicIPs.primary),

    /* Nameserver */
    A("ns1", publicIPs.primary),
    AAAA("ns1", "2a01:4f8:211:fc01::1a"),
    A("robotns2", publicIPs.robotns2),
    A("robotns3", publicIPs.robotns3),

    A("www", publicIPs.primary),
    AAAA("www", "2a01:4f8:211:fc01:499:18::1")
);
```

Puppet – Resources

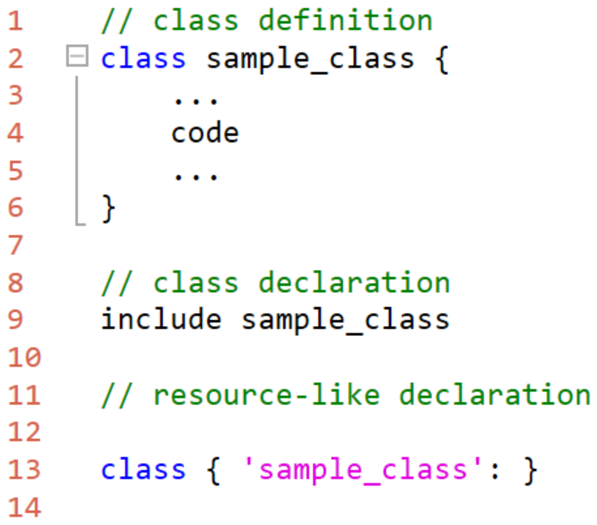
- Puppet is pure declarative
- Everything is a resource
- Resources may have parameters



```
user { 'mitchell':
  ensure      => present,
  uid         => '1000',
  gid         => '1000',
  shell       => '/bin/bash',
  home        => '/home/mitchell'
}

```

Puppet – Classes

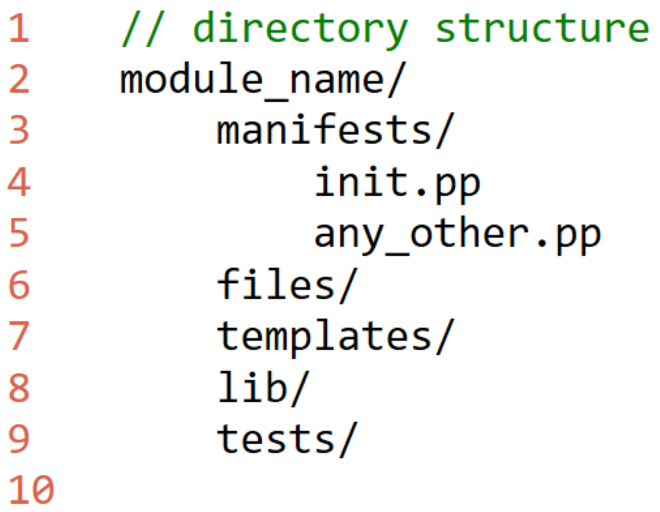
- Classes encapsulate code like in any other programming language
- Classes are a construct to model reusable code
- Classes are used in manifests
- When a class is included, Puppet evaluates the code in the class
- Classes may be used as resources to create “new” resources like an Apache web server or anything else



```
// class definition
class sample_class {
    ...
    code
    ...
}

// class declaration
include sample_class

// resource-like declaration

class { 'sample_class': }

```

Puppet – Modules

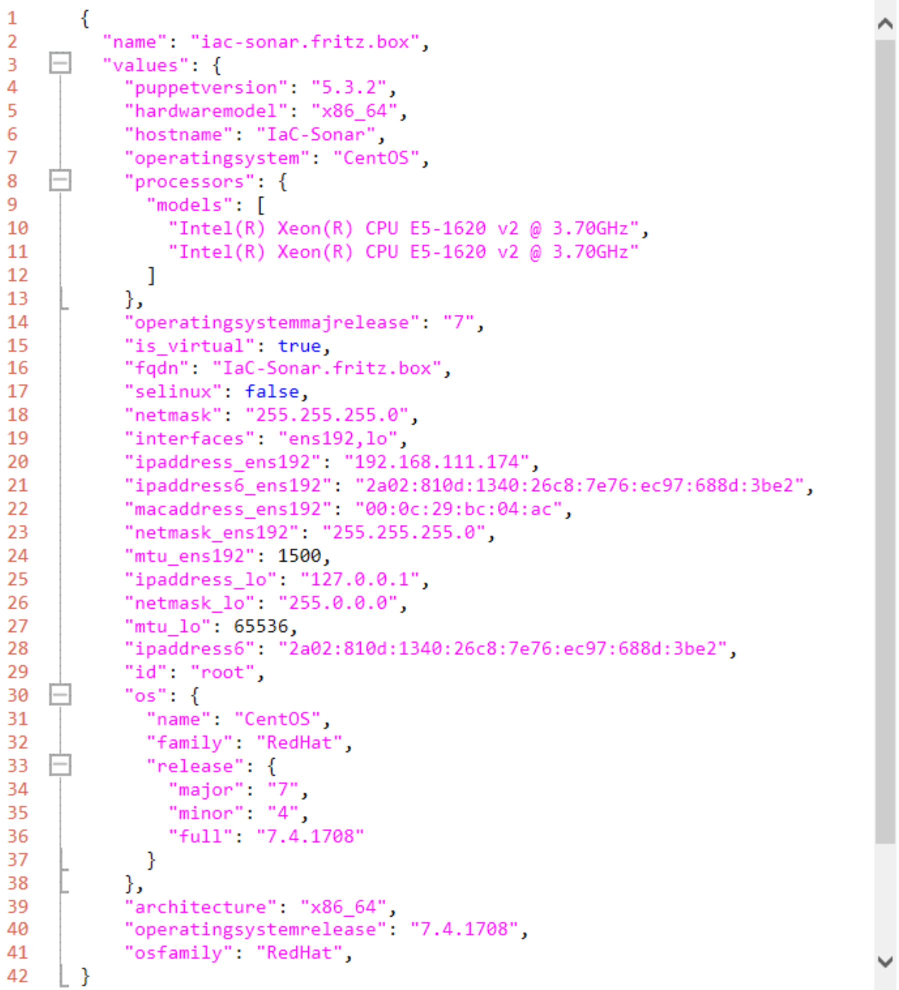
- Modules aggregate manifests
- Contains:
  - Files
  - Templates
  - Classes
  - Ruby code for processing
- Encapsulates management of a special resource or application (like SonarQube)



```
// directory structure
module_name/
    manifests/
        init.pp
        any_other.pp
    files/
    templates/
    lib/
    tests/

```

Puppet – Facts

- Facter collects information about the local system
- Collected information is used by modules, classes and manifests
- Facter is very thoroughly!



```
{
  "name": "iac-sonar.fritz.box",
  "values": {
    "puppetversion": "5.3.2",
    "hardwaremodel": "x86_64",
    "hostname": "IaC-Sonar",
    "operatingsystem": "CentOS",
    "processors": {
      "models": [
        "Intel(R) Xeon(R) CPU E5-1620 v2 @ 3.70GHz",
        "Intel(R) Xeon(R) CPU E5-1620 v2 @ 3.70GHz"
      ]
    },
    "operatingsystemmajrelease": "7",
    "is_virtual": true,
    "fqdn": "IaC-Sonar.fritz.box",
    "selinux": false,
    "netmask": "255.255.255.0",
    "interfaces": "ens192,lo",
    "ipaddress_ens192": "192.168.111.174",
    "ipaddress6_ens192": "2a02:810d:1340:26c8:7e76:ec97:688d:3be2",
    "macaddress_ens192": "00:0c:29:bc:04:ac",
    "netmask_ens192": "255.255.255.0",
    "mtu_ens192": 1500,
    "ipaddress_lo": "127.0.0.1",
    "netmask_lo": "255.0.0.0",
    "mtu_lo": 65536,
    "ipaddress6": "2a02:810d:1340:26c8:7e76:ec97:688d:3be2",
    "id": "root",
    "os": {
      "name": "CentOS",
      "family": "RedHat",
      "release": {
        "major": "7",
        "minor": "4",
        "full": "7.4.1708"
      }
    },
    "architecture": "x86_64",
    "operatingsystemrelease": "7.4.1708",
    "osfamily": "RedHat",
}
```

Puppet – CLI

CMDlet	Explanation
puppet apply <path/to/file>	Applies given manifest to local host system
puppet module install <module name>	Installs a module to the local system
puppet module list	List all local installed modules
puppet facts	List local
puppet help	Displays the help of the Puppet CLI