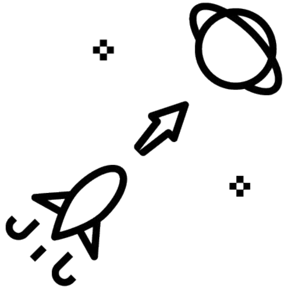
Puppet Server Tuning and Scaling

Martin Alfke
<ma@betadots.de>



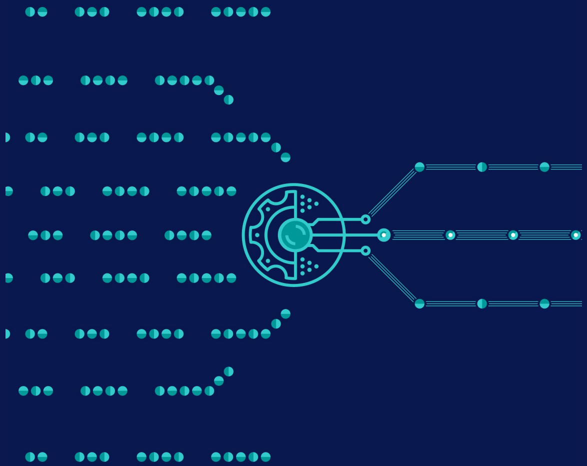
Martin Alfke
CEO/Consultant/Trainer at betadots GmbH
Berlin, Germany

- Puppet Trainer and Puppet Solution Engineer
- Platform Engineering, Consulting and Training
- Agile methods, Scrum
- tuxmea (Twitter, GitHub, Slack)

Puppet Server Tuning and Scaling

In large environments with many nodes one must take care of Puppet server scaling.

When and if scaling is needed, depends on the number of nodes and on code complexity and size.



Puppet Server tuning

Puppet Server tuning

We sometimes see the usage of horizontal scaling, even when there is no need to do it.

As scaling horizontally needs further infrastructure components like Load Balancer and Puppet Server Compilers, we usually first ask to do performance tuning on the single node instance.

A properly configured and optimized single Puppet server should be able to handle up to 2500 nodes using default runinterval of 30 minutes.

https://www.puppet.com/docs/pe/2023.5/hardware_requirements#hardware_requirements_standard

Puppet Server tuning - Java version

Puppet server package has a dependency on java openjdk (headless).

On most Linux distributions this will install Java 1.8.

Please ensure to upgrade to Java 17 and set the java alternative accordingly or set the JAVA path in `/etc/[sysconfig|default]/puppetserver`:

e.g (on Almalinux 8):

```
alternatives --set java \
/usr/lib/jvm/java-17-openjdk-17.0.9.0.9-2.el8.x86_64/bin/jav
a
```

Puppet Server tuning - JRuby instances

With its default configuration a Puppet Server spins up (number of CPU - 1) with a minimum of 1 and a maximum of 4 JRuby instances.

Each instance is able to handle a single Puppet Agent request.

When there are more requests, these get queued.

Each JRuby instance needs 512 MB of Java Heap RAM.

We were unable to run more than 32 JRuby instances on a single node!

Puppet Server tuning - JRuby instances

Adopting the number of JRuby instances takes place in `/etc/puppetlabs/puppetserver/conf.d/puppetserver.conf` within the `jruby-puppet` section by setting the `max-active-instances` parameter:

```
jruby-puppet: {
    ...
    max-active-instances: 8   # <------- JRuby instances
    ...
}
```

Please note that increasing the number of JRuby instances causes the Java process to need more Java HEAP RAM.

Puppet Server tuning - Java heap size

The Java engine should have enough RAM to be able to spin up and maintain the JRuby instances. The default value is set to 2 GB.

If you forget to increase Java Heap size, you will see Java out-of-memory error messages in the Puppet server logfile.

Increasing Java Heapsize takes place in `/etc/default/puppetserver`(Debian based) or `/etc/sysconfig/puppetserver`(RedHat based).

The Java Heap size is provided as a Java argument:

# /etc/[sysconfig|default]/puppetserver

```
JAVA_ARGS="-Xms18g -Xmx18g
-Djruby.logger.class=com.puppetlabs.jruby_utils.jruby.Slf4jLogger ..."
```

Puppet Server tuning - Java heap size

It is considered best practice to set upper and lower limit to the same value, so Java reserves the RAM upon start up.

Please note that the maximum possible value is limited by the amount of the system RAM.

If you increase the Java heap size beyond system RAM, you will find Kernel out-of-memory errors in the journal.

Puppet Server tuning - Java heap size

Besides this one must be aware that Java switches memory handling when using more than 32 GB of heap, which reduce the amount of objects.

Also see the blog post from Codecentric and Atlassian:

- https://www.codecentric.de/wissens-hub/blog/35gb-heap-less-32gb-java-jvm-memory-oddities
- https://confluence.atlassian.com/jirakb/do-not-use-heap-sizes-between-32-gb-and-47-gb-in-jira-compressed-oops-1167745277.html

Puppet Server tuning - Reserved Code cache

When Puppet servers receive a request from a node the Puppet Code is loaded into code cache in memory.

The default value is set to 512 MB RAM.

A larger Puppet code base or complex code might need a larger Code cache setting.

Code cache is configured as Java argument in `/etc/[sysconfig|default]/puppetserver`:

# /etc/[sysconfig|default]/puppetserver

```
JAVA_ARGS="-Xms18g -Xmx18g -XX:ReservedCodeCacheSize=1024m
-Djruby.logger.class=com.puppetlabs.jruby_utils.jruby.Slf4jLogger
..."
```

Puppet Server tuning - Reserved Code cache

Please note that the maximum possible value is at `2048m`.

Please ensure that the Puppet server process was restarted after setting all the required configuration options.

A short mathematical sizing rule of thumb:

```
Java Heap size(M) = ( No of JRuby instances * 512M ) + 512M

Reserved Code Cache Size (M) = No of JRuby instances * 128M
```

Puppet DB tuning

In most environments PuppetDB is used to store facts, reports, catalogs and exported resources.

The more nodes you have in your infrastructure, the higher the load and the memory requirement on the PuppetDB process.

PuppetDB is a HTTP(S) Rest API in front of a PostgreSQL database.

One can say that PuppetDB also is a kind of web service.

Puppet DB tuning - Java heap size

The most important setting is Java heap size.

Per default a PuppetDB is configured to use 512MB Heap size.

Configuration takes place in `/etc/sysconfig/puppetdb` (RedHat/SLES) or `/etc/default/puppetdb` (Debian).

The Java Heap size is provided as a Java argument:

# /etc/[sysconfig|default]/puppetdb

```
JAVA_ARGS="-Xms1g -Xmx1g ..."
```

Puppet DB tuning - DB connection pool

Within the PuppetDB configuration file, which is located at `/etc/puppetlabs/puppetdb/conf.d/database.conf`, one can set the maximum numbers of idle and active PostgreSQL connections within the `database` section

```
maximum-pool-size = 25 # default
```

Please note that PuppetDB uses two connection pools:

- read pool

- write pool

Puppet DB tuning - DB connection pool

The read pool can be set individually by setting `maximum-pool-size` in the `read-database` section. The default value is 10.

The `maximum-pool-size` should be set to the sum of both.

Please check your PostgreSQL configuration (especially the max connection setting) prior increasing the connection pool.

Puppet DB tuning - Block facts

Another possibility is to lower the number of facts stored.

This can be achieved by setting `facts-blocklist`.

```
facts-blocklist = ["fact1", "fact2", "fact3"]
```

Puppet DB tuning - Threads

Besides this PuppetDB lets you configure the number of threads by setting `threads` within the `command-processing` section.

As default PuppetDB will use half the number of cores of the system.

Last but least one can set the `max-threads` in `/etc/puppetlabs/puppetdb/conf.d/jetty.ini`.

This specifies the number if parallel possible HTTP and HTTPS connections.

Puppet Server scaling

Puppet Server scaling

If a single Puppet server is not able to handle the amount of requests - e.g. in large infrastructures exceeding 2000 nodes - one has the option to set up additional compilers.

Please note that each Puppet server infrastructure component should receive the performance tuning settings!

Puppet Server scaling - Infrastructure

A high performance Puppet infrastructure consists of the following components:

- CA Server
- Compiler(s)
- Load balancer

A Puppet agent sends the request to the Load balancer.

The load balancer passes the requests to a free compiler.

Puppet Server scaling - Infrastructure

From a Puppet agent point of view, the request is sent to the Load balancer and the response is received from the compiler.

This has a special meaning when it comes to SSL certificates and strict SSL validation.

We will discuss this when we come to the compiler setup.

Puppet Server scaling - Infrastructure - CA Server

The Puppet CA server is a standalone single instance, which is used to spin up and maintain additional Puppet compilers.

From CA server point of view a compiler is just a node.

All new CSR's must be signed on the CA server.

When you want to scale your Puppet infrastructure the CA Server will be your existing Puppet server.

Puppet Server scaling - Infrastructure - CA Server

When we want to sign the compilers certificates including dns_alt_names, we must configure the CA instance, to be able to do this by modifying the `/etc/puppetlabs/puppetserver/conf.d/ca.conf` file:

We must allow subject alt names setting:

```
# /etc/puppetlabs/puppetserver/conf.d/ca.conf
certificate-authority: {
    # allow CA to sign certificate requests that have subject alternative names.
    allow-subject-alt-names: true  # <----- enable SAN cert signing
    # allow CA to sign certificate requests that have authorization extensions.
    # allow-authorization-extensions: false
    # enable the separate CRL for Puppet infrastructure nodes
    # enable-infra-crl: false
}
```

Puppet Server scaling - Infrastructure - Compilers

Compilers should **not** act as CA Server!
The CA functionality is managed in `/etc/puppetlabs/puppetserver/services.d/ca.cfg`

Now we need to change the settings so that a compiler does not act as CA server, but pass all CA related requests to the CA server:

```
# /etc/puppetlabs/puppetserver/services.d/ca.cfg
# To enable the CA service, leave the following line uncommented
# puppetlabs.services.ca.certificate-authority-service/certificate-authority-service
# To disable the CA service, comment out the above line and uncomment the line below
puppetlabs.services.ca.certificate-authority-disabled-service/certificate-authority-di
sabled-service
puppetlabs.trapperkeeper.services.watcher.filesystem-watch-service/filesystem-watch-se
rvice
```

Puppet Server scaling - Infrastructure - Compilers

The Puppet agent will not connect to a compiler but to the load balancer.

If we keep the default setup, the Puppet agent will refuse to connect to the load-balancer, if DNS ALT Names are missing in the compilers certificate.

The compiler **must** have the Load balancer DNS name configured, prior generating the CSR.

# /etc/puppetlabs/puppet/puppet.conf

```
[agent]
dns_alt_names = loadbalancer.domain.tld,compiler-fqdn.domain.tld
```

Puppet Server scaling - Infrastructure - Load balancer

Each request to a Puppet Server starts a compilation process with different runtimes. One should **not** configure the roundrobin distribution algorithm on the load balancer.

Instead we want to distribute the connections to the Puppet compiler with the least work to do - which means the one with the least connections. In HAproxy this setting is called **leastconn**.

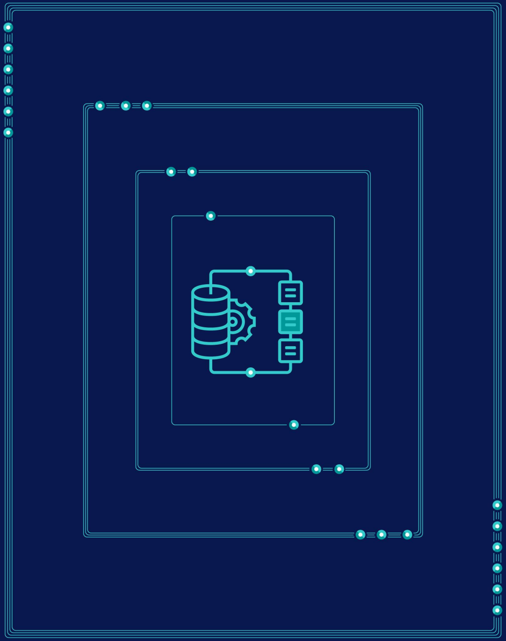
Besides this, you do not want to rely on Layer 3 IP connection, but on Layer 7 functionality. Within HAproxy one should add the **SSL** directive to each backend.

Puppet Server scaling - Infrastructure - PuppetDB

In former times, it was best practice to have a single PuppetDB instance on the Puppet Server.

If you have the need for scaling Puppet Servers, one should consider having PuppetDB on each Puppet infrastructure node, which connect to a centralized PostgreSQL database.

Analyzing performance win



Analyzing Performance win - Compile times

Every time one does performance tuning, one also wants to get proof that the new setting has improved performance.

The most simple check is by getting the compile times from puppetserver logfile.

```
awk '/Compiled catalog/ {print $12" "$14}'
/var/log/puppetlabs/puppetserver/puppetserver.log
```

```
production 0.07

production 0.05

production 0.04

production 0.03
```

Analyzing Performance win - Compile times

You can also get highest and lowest compile time

```
awk '/Compiled catalog/ {print $12" "$14}'
/var/log/puppetlabs/puppetserver/puppetserver.log | sort | awk 'NR==1; END{print}'
```

production 0.03

production 0.07

If you need an average compile time the following command will be helpful

```
awk '/Compiled catalog/ {print $12" "$14}'
/var/log/puppetlabs/puppetserver/puppetserver.log | sort | awk '{total += $2; count++
} END { print total/count }'
```

0.0442857

Analyzing Performance win - Metrics

Another option is provided by the Puppet Operational Dashboards module

(https://forge.puppet.com/modules/puppetlabs/puppet_operational_dashboards/readme).

This will setup a Grafana frontend which reads data from an InfluxDB, which gets its content from a Telegraf agent which connects via HTTPS to the Puppet server metrics endpoints.

In Puppet Enterprise the PostgreSQL database also uses SSL certificates, which also allows fetching PostgreSQL metrics.

When running Puppet Open Source, one needs to add an additional telegraf configuration to query the PostgreSQL database using a user and password and push the data into InfluxDB.

Distributed Puppet Agent runs

Distributed Puppet agent runs

In the introduction we mentioned that a single server can handle up to 2500 nodes.

This number of nodes requires that Puppet agent runs must be evenly distributed over time.

The default way for a Puppet agent is running as a daemon, which will schedule a Puppet agent run upon start and triggers the next runs every 30 minutes (`runinterval` config option).

Due to the reason that systems might be starting in parallel, one will still see overloaded Puppet server infrastructure.

As a result one will recognize that the Puppet agent run takes very long time. If the Agent did not receive a reply from Puppet server within 5 minutes (`http_read_timeout`config option), it will stop the actual request and repeat again in 30 minutes.

Distributed Puppet agent runs

The config options can be modified in `puppet.conf` config file in section `agent`.

Please reconsider if Puppet should run every 30 minutes.

In some infrastructure Puppet is running in `noop` mode as changes may only be deployed during maintenance window.

In this case one still wants the node to regular check its configuration, but maybe 4 times a day only, so people can identify within the reporting that the systems are still in desired state.

Doubling the `runinterval` option reduces the load on the Puppet server to its half, so the Puppet server can handle double number of nodes.

Distributed Puppet agent runs

One might still see load spikes on the Puppet server.

It still can happen that too many nodes try to request for their configuration in parallel.

We (except for Robert) usually recommend to customers to run the agent via cron.

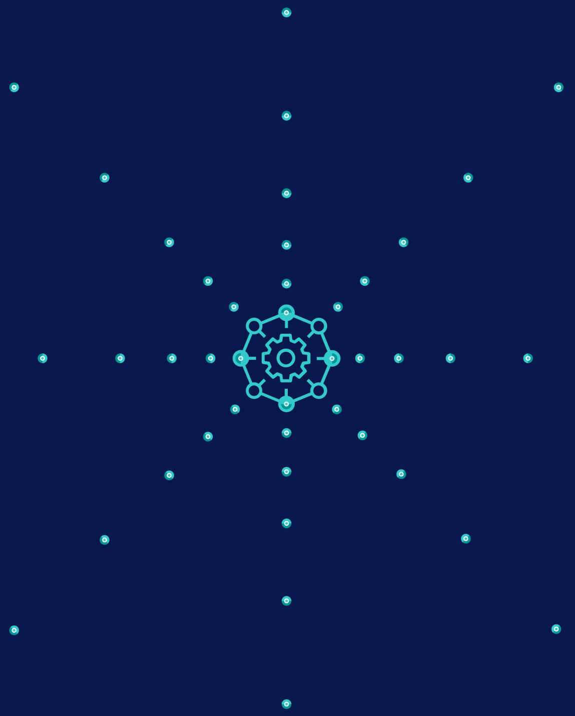
One run will be done at reboot and the other runs will take place as recurring cron entries.

To distribute the agent runs over time one can make use of the `fqdnrand` function. This function generates a unique number for a hostame or FQDN.

One can use the reidmv puppet_run_scheduler module to configure Puppet accordingly.

https://forge.puppet.com/modules/reidmv/puppet_run_scheduler/readme

Please note that within Puppet Enterprise environments the Puppet agent on the Puppet servers **must** be running as agent daemon service!



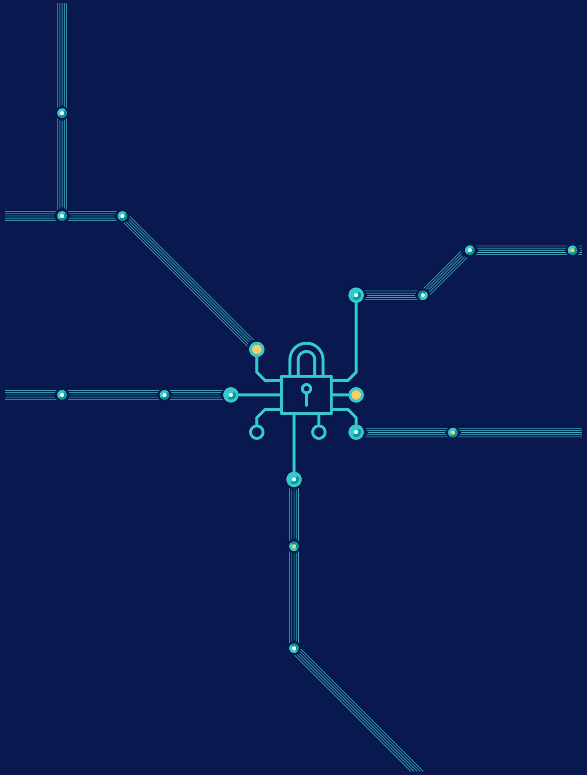
Summary

Tuning and scaling a Puppet server is very simple, as it is a single Java process and a web service only.

Tuning is mostly related to the Java process and memory handling and is usually limited by the amount of RAM and numbers of CPUs being available.

Scaling can be achieved by using simple, already established solutions to distribute web traffic amongst several servers.

Distributing the Puppet agent runs ensures evenly used Puppet servers which allows more nodes to get managed.



"Fun" part: Scaling beyond JRuby limits

Scaling beyond JRuby limits

We already mentioned that you can not spin up more than 32 JRuby instances.

We tried it. Be sure. It is not working at all.

But what to do if you have high end (e.g. HPC) hardware?

- 72 cores

- 348 GB RAM

Usually you want to slice the big hardware e.g using KVM or Docker and run multiple VMs or Puppet server containers for example by using Voxpupuli CRAFTY.

https://dev.to/betadots/puppet-containers-are-back-alive-5c2n

https://github.com/voxpupuli/crafty

Scaling beyond JRuby limits

Normally we see such a hardware only within virtualization or container based environments or within a HPC platform.

In our specific use case the usage of KVM, Docker, Podman or Kubernetes was not possible. Don't ask.

We talked about this in the Puppet community Slack channel and people told us that the only option is to spin up multiple Puppet server instances by "some specific symlinking and copying of config files".

Just to be sure: this is not something you can configure out-of-the-box, but needs lots of changes also to Puppet internal scripts.

Scaling beyond JRuby limits

Providing all information here would take too long.

We have created a blog-posting where you can find the required configurations and changes to run multiple Puppet server instances from a single Puppet server installation:

https://dev.to/betadots/scaling-puppet-infrastructure-3p2

Puppet Server Tuning and Scaling

Thank you.