Kolmo

Making automation & configuration easier

bert hubert

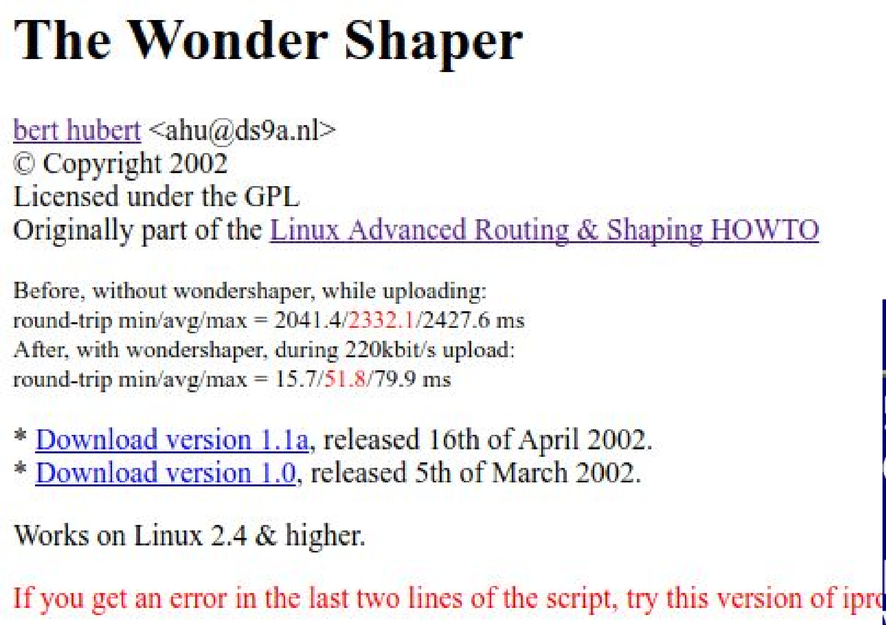
https://kolmo.org/ - http://tinyurl.org/kolmopreso

@PowerDNS_Bert

POWERDNS
AN OX COMPANY

UNHCR
The UN Refugee Agency

128Kbit/s, 500ms latency.
Also, Biddinghuizen



The Wonder Shaper

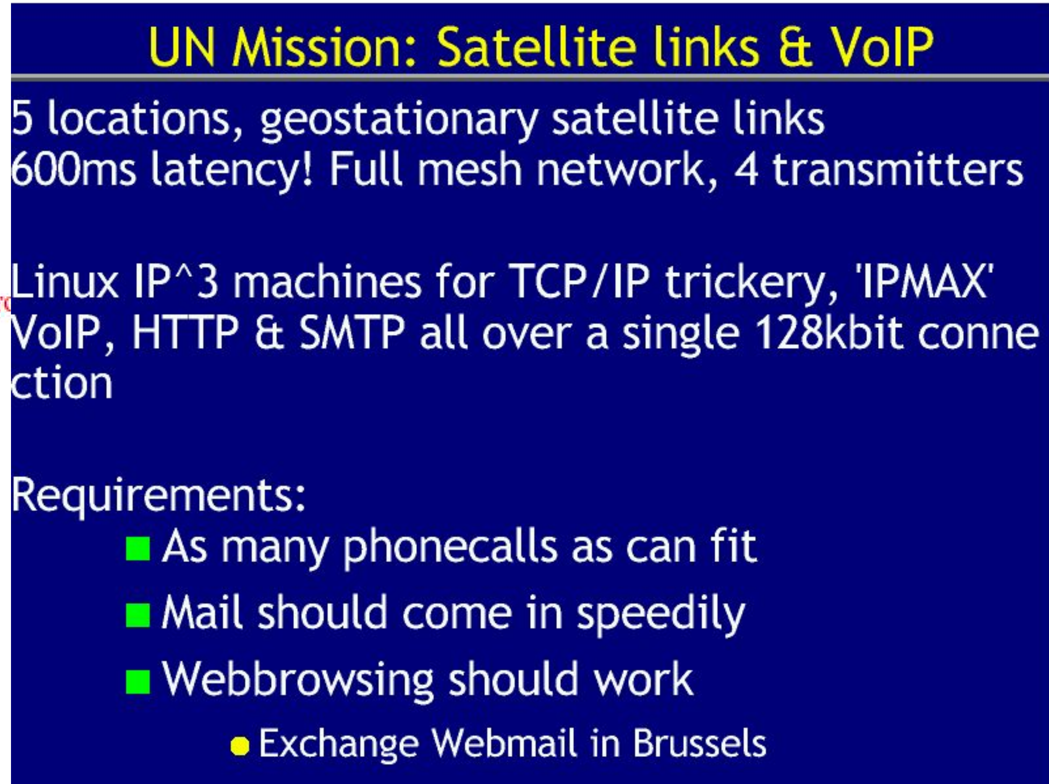
bert hubert <ahu@ds9a.nl>
© Copyright 2002
Licensed under the GPL
Originally part of the Linux Advanced Routing & Shaping HOWTO

```
Before, without wondershaper, while uploading:
round-trip min/avg/max = 2041.4/2332.1/2427.6 ms
After, with wondershaper, during 220kbit/s upload:
round-trip min/avg/max = 15.7/51.8/79.9 ms
```

* Download version 1.1a, released 16th of April 2002.
* Download version 1.0, released 5th of March 2002.

Works on Linux 2.4 & higher.

If you get an error in the last two lines of the script, try this version of ipr



UN Mission: Satellite links & VoIP

5 locations, geostationary satellite links
600ms latency! Full mesh network, 4 transmitters

Linux IP^3 machines for TCP/IP trickery, 'IPMAX'
VoIP, HTTP & SMTP all over a single 128kbit conne
ction

Requirements:

- As many phonecalls as can fit
- Mail should come in speedily
- Webbrowsing should work
    - Exchange Webmail in Brussels

Even Linux is not magic, but..

Network was broken by design and the requirements where conflicting.

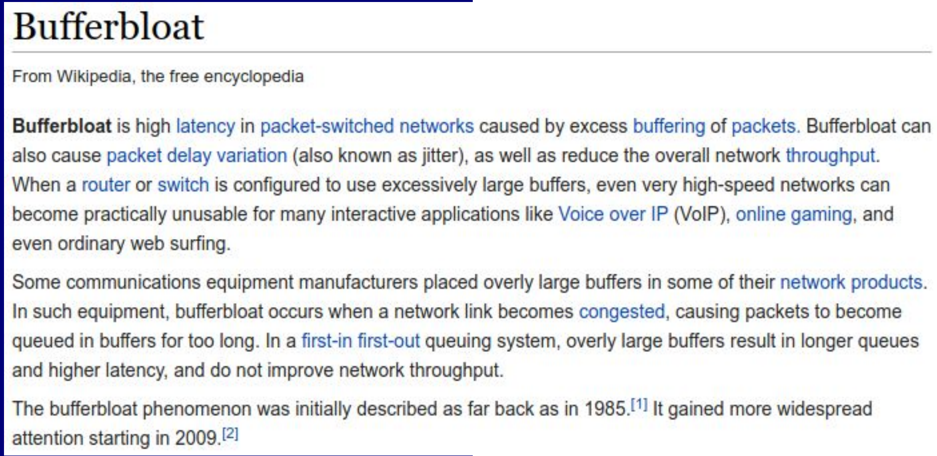
However, following tricks helped:

* Removal of Cisco cruft

* 5 additional Linux machines

- Iptables MSS clamp
- Agressive Queuecontrol
- Shaping (CBQ+TBF)
- Iptables driven MRTG

* Adjustment of expectations



Bufferbloat

From Wikipedia, the free encyclopedia

Bufferbloat is high latency in packet-switched networks caused by excess buffering of packets. Bufferbloat can also cause packet delay variation (also known as jitter), as well as reduce the overall network throughput. When a router or switch is configured to use excessively large buffers, even very high-speed networks can become practically unusable for many interactive applications like Voice over IP (VoIP), online gaming, and even ordinary web surfing.

Some communications equipment manufacturers placed overly large buffers in some of their network products. In such equipment, bufferbloat occurs when a network link becomes congested, causing packets to become queued in buffers for too long. In a first-in first-out queuing system, overly large buffers result in longer queues and higher latency, and do not improve network throughput.

The bufferbloat phenomenon was initially described as far back as in 1985.[1] It gained more widespread attention starting in 2009.[2]

You can’t download
the configuration?!?!

You can’t even commit
the configuration?!?!

“Never underestimate the bandwidth of an armoured helicopter in Bosnia” - no one ever

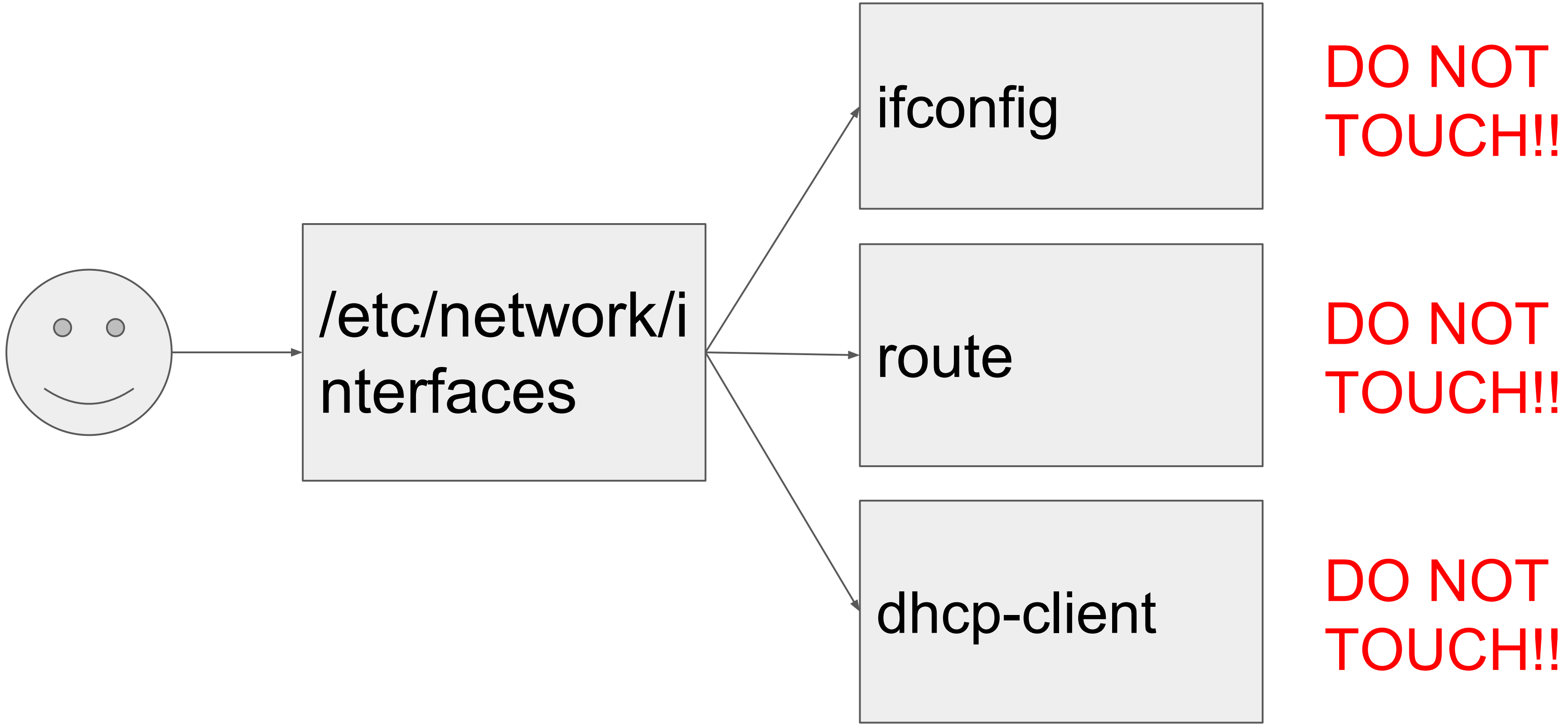
Configuration

- Make runtime/boot time changes to:
    - iptables
    - ifconfig txqueuelen
    - /proc/sys/
    - bridge (brctl)
    - Actual QoS (tc, htb, cbq)

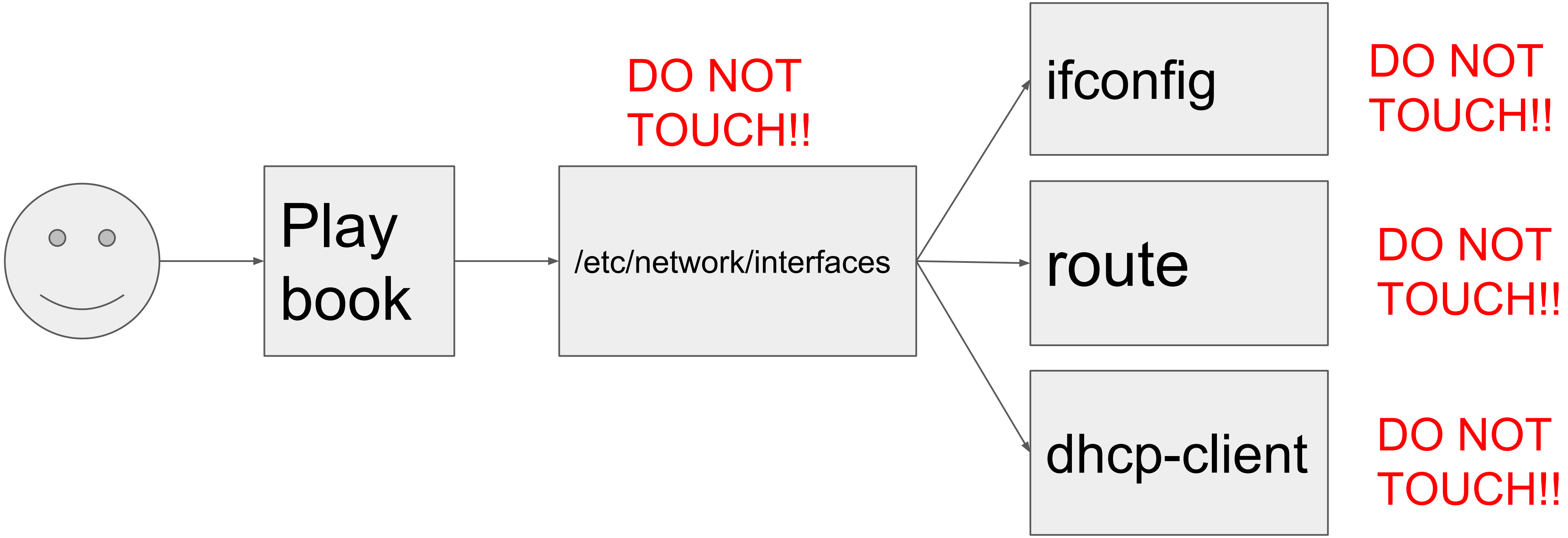
- ALL OF THESE could not be serialized
    - Pretty output is available - for humans
    - Most of them to this day can’t be saved/serialized
    - Exception: iptables-save / iptables-restore

Central problem: hard to figure out server/system state

2000 solution: generation



2014 solution: generation



Automation workflow

- Create playbook
- Run playbook on servers
- Modify playbook bit in response to new needs
- Deploy to servers
- Happiness

- **Lies!**

Actual automation workflow

- Create playbook, run playbook on server
- **Server does nothing like you want it to**
- Inspect server
- Find Ansible ‘lineinfile’ triggers on a comment block
- Change playbook, rerun
- Find further problems, change playbook, rerun
- Webserver does not do what you want, work on server to figure out how to get it to do what you want
- Attempt to put those changes in playbook, finally it works
- **Rerun playbook against fresh server: nothing works**

Automation dream workflow

- Deploy fresh server
- Configure everything, on that server, until it is exactly how you like it
- **Download server state**
- Insert state in playbook
- Happiness

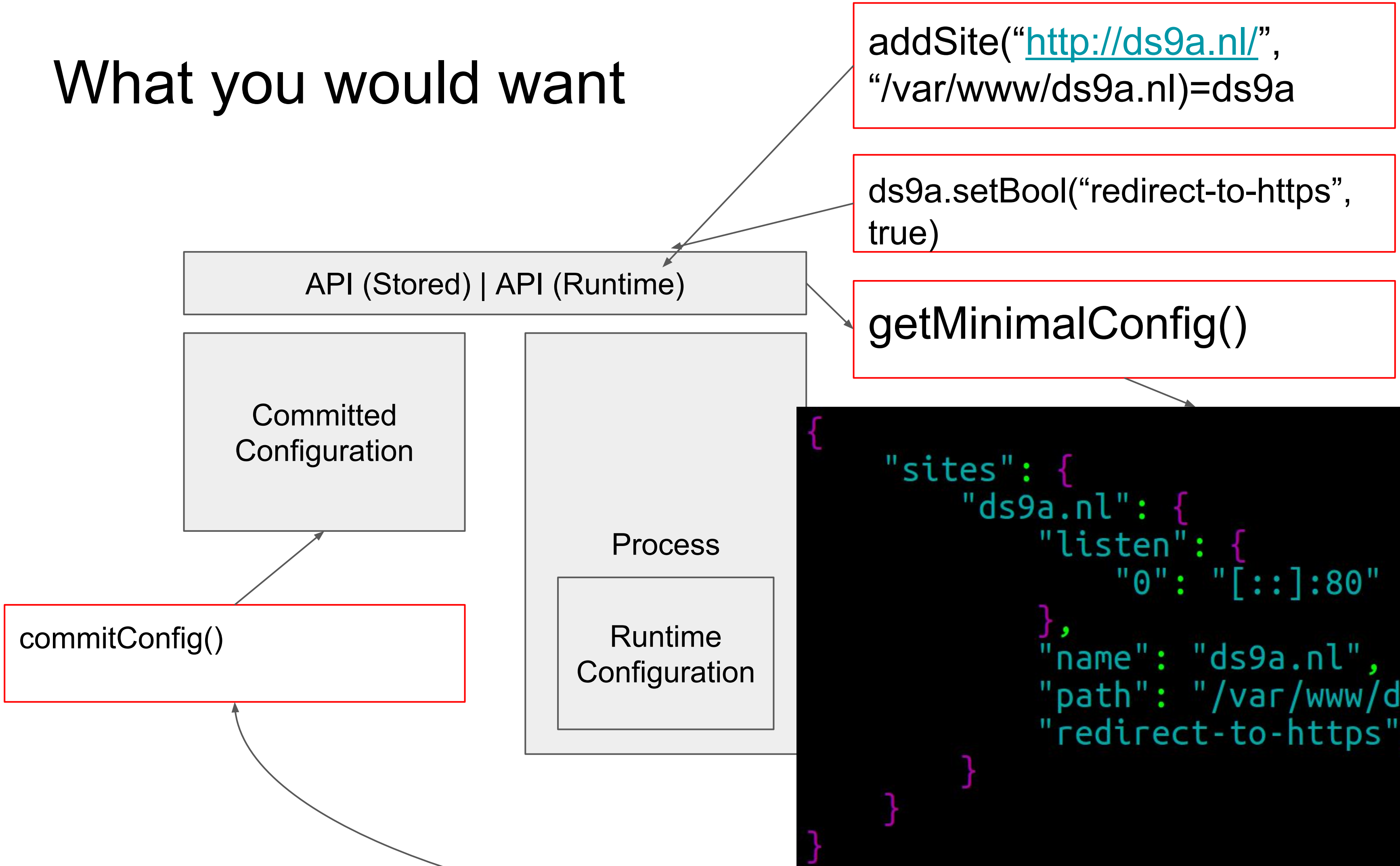
Why it can’t be done

- A typical process if configured using several configuration files
- Some of them under distribution/operating system control
- Some of them explicitly meant for you to change them
    - Include.d directories, sites-enabled, sites-available
- Some of them are TEH HUGE and actually contain large parts of the software logic (Exim)
- There is no easy way to “extract your changes” from this set of configuration files
- There are only very painful ways to “insert your changes” into existing configuration files
    - (sometimes there is an ansible module that helps)

Secret 2017 solution

- “We’re doing automation!”
- “We can compose our infrastructure at will!”
- **Why are you stuck on CentOS 6.low?**
- Um, well, our playbooks simply copy lots of files from a 6.0 installation to the target
    - And those files create errors on newer releases
    - Not actually “configuring” services, only distributing large amounts of configuration
- Yeah, you are doing “automation”

What you would want

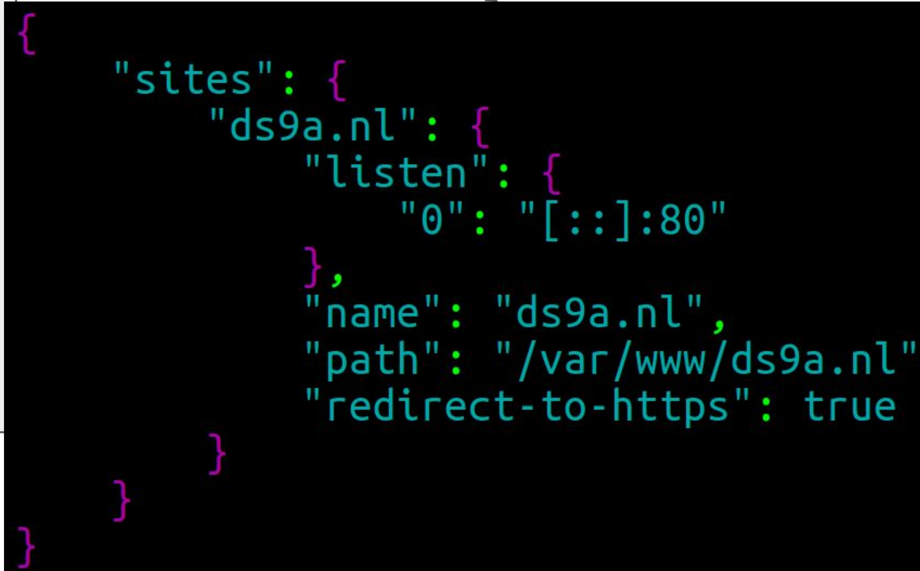


```
addSite("http://ds9a.nl/",
"/var/www/ds9a.nl)=ds9a
```

```
ds9a.setBool("redirect-to-https",
true)
```

```
getMinimalConfig()
```

```
commitConfig()
```

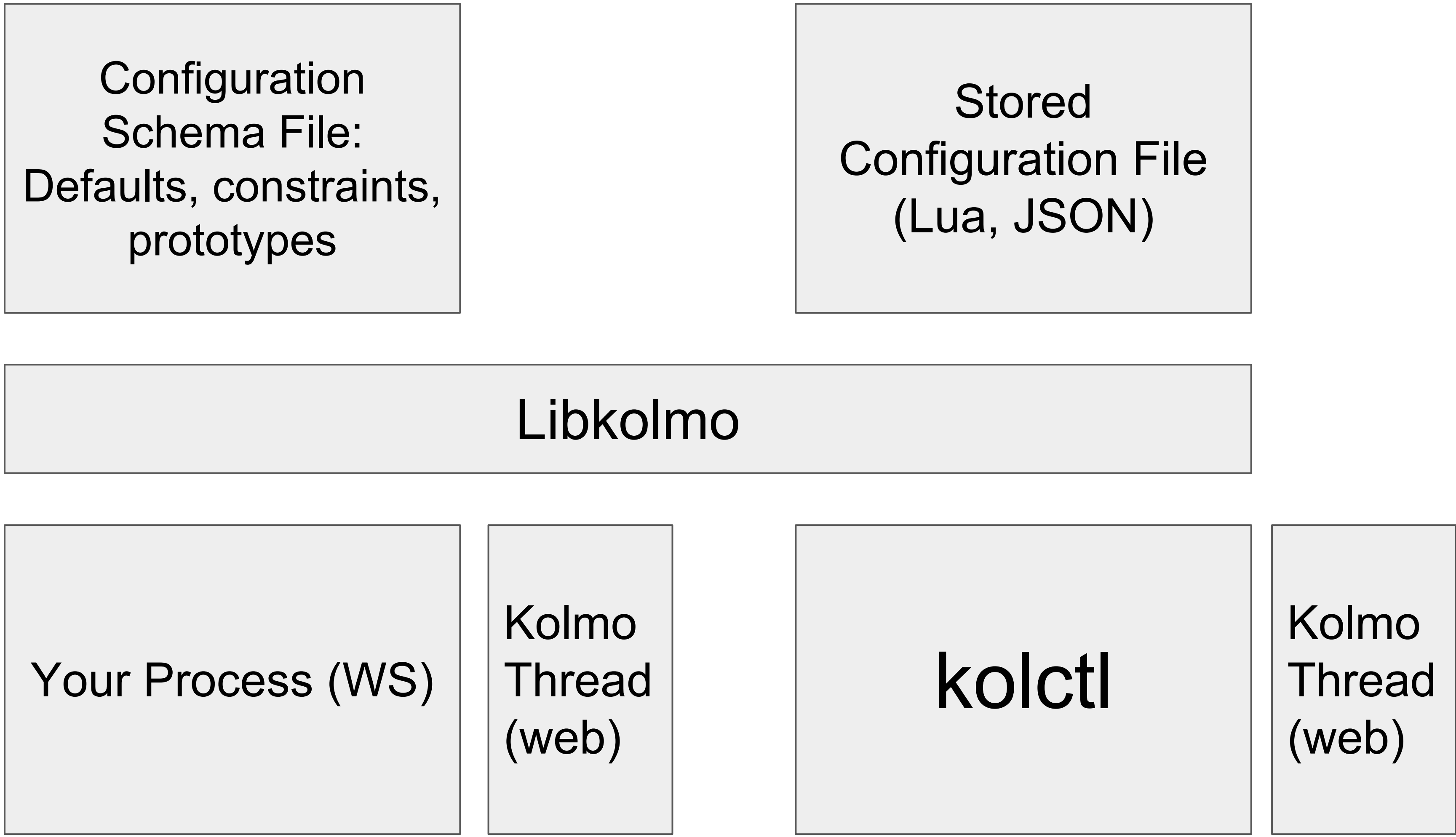


```
{
    "sites": {
        "ds9a.nl": {
            "listen": {
                "0": "[::]:80"
            },
            "name": "ds9a.nl",
            "path": "/var/www/ds9a.nl"
            "redirect-to-https": true
        }
    }
}
```

Introducing: Kolmo

Inspect, retrieve, modify & deploy configuration safely

- Programmatic access to configuration
    - Read/Write
- Self-documenting
- Client-side, offline, validity checking
- Meant to ease “automation”
- **Library & tools to support all kinds of systems/services**


Configuration Schema File: Defaults, constraints, prototypes
Stored Configuration File (Lua, JSON)
Libkolmo
Your Process (WS)
Kolmo Thread (web)
kolctl
Kolmo Thread (web)

The Schema: Welcome to the default free zone

- Defines all configuration settings
  - Type (YES IT IS TYPESAFE)
  - Defaults (ALL defaults)
  - Mandatory / optional
  - **DESCRIPTION**
  - **UNITS**
  - **CONSTRAINTS**

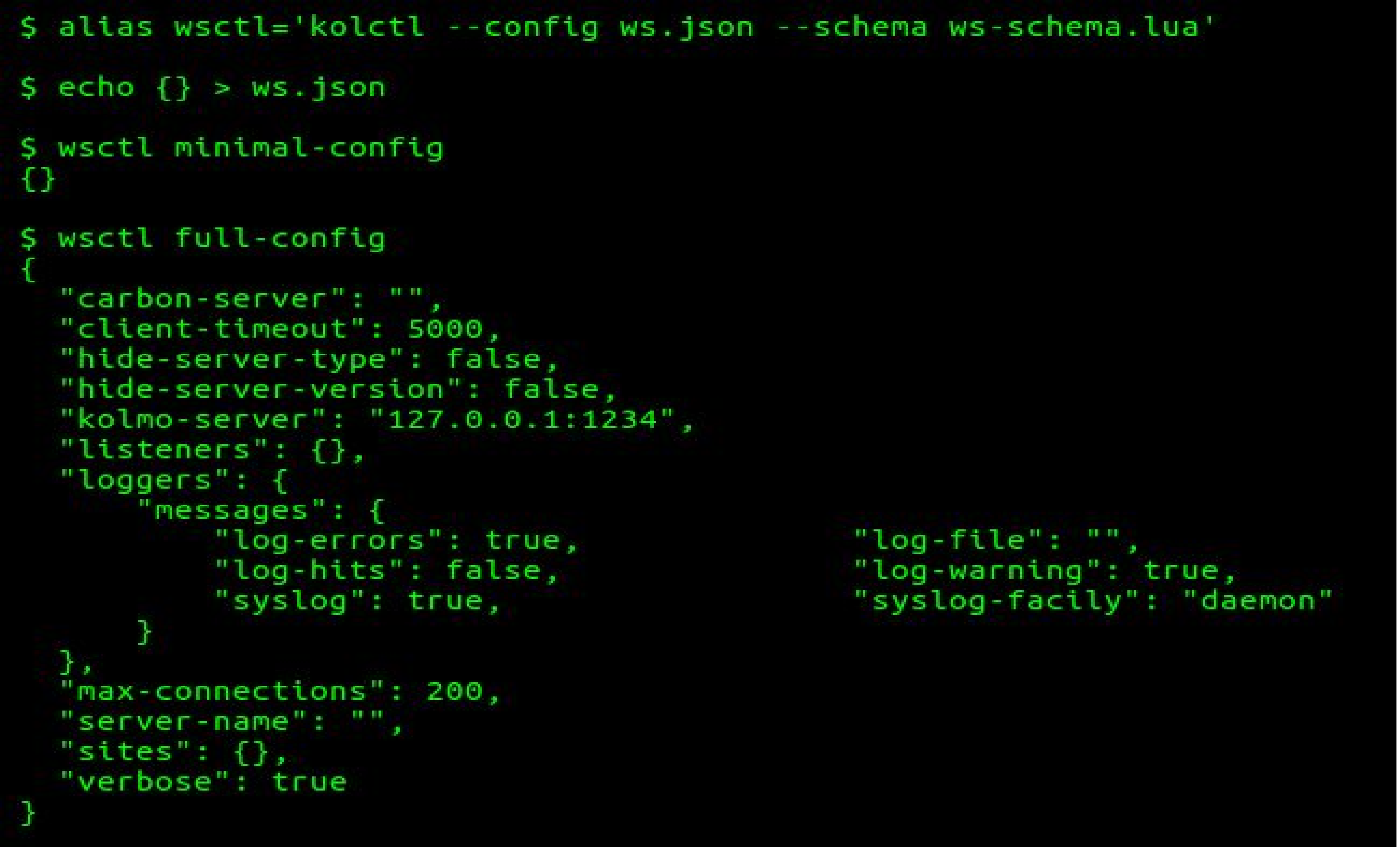
- If nothing else, the configuration schema is GREAT documentation of your configuration file and all defaults!
  - **“The problem with documentation is that the compiler does not read it”**

“Ws”: the Kolmo “Hello, World” application

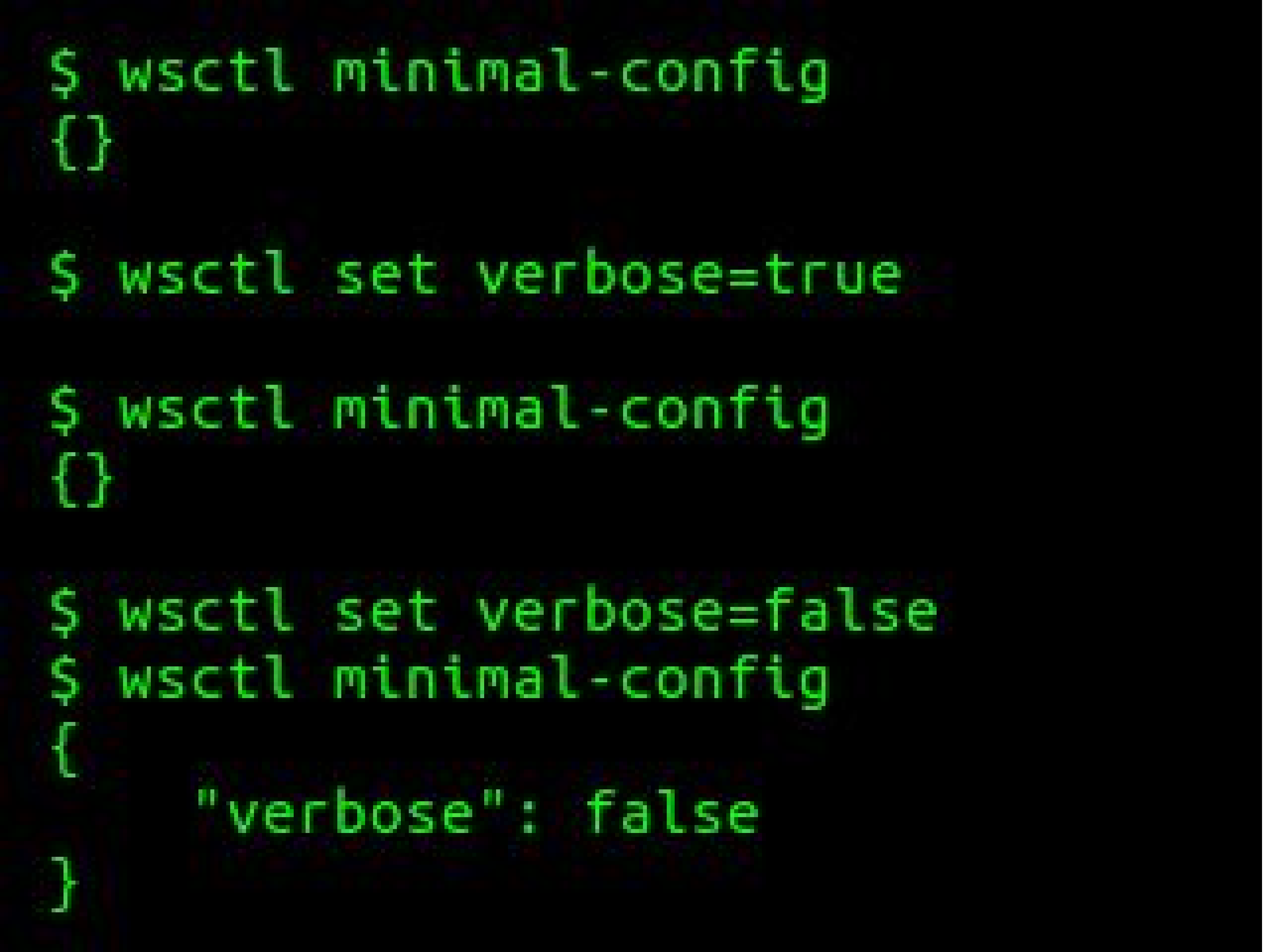
(that powers https://kolmo.org/)

```
main:registerVariable("verbose", "bool",
      { default="true",
        runtime="true",
        cmdline="-v",
        description="Perform verbose logging"
      })

site=createClass("site", {})
site:registerVariable("name", "string",
      { runtime="false", description="Hostname of this website"})

site:registerVariable("enabled", "bool",
      { runtime="false", default="true", description="If this site is enabled"})

site:registerVariable("path", "string",
      { runtime="true", description="Path on filesystem where content is hosted"})

site:registerVariable("listen", "struct",
      { member_type="ipendpoint", runtime="false", description="IP endpoints we listen on"})

site:registerVariable("redirect-to-https", "bool",
      { default="false", runtime="true",
        description="If all http requests should be redirected to https"})
```

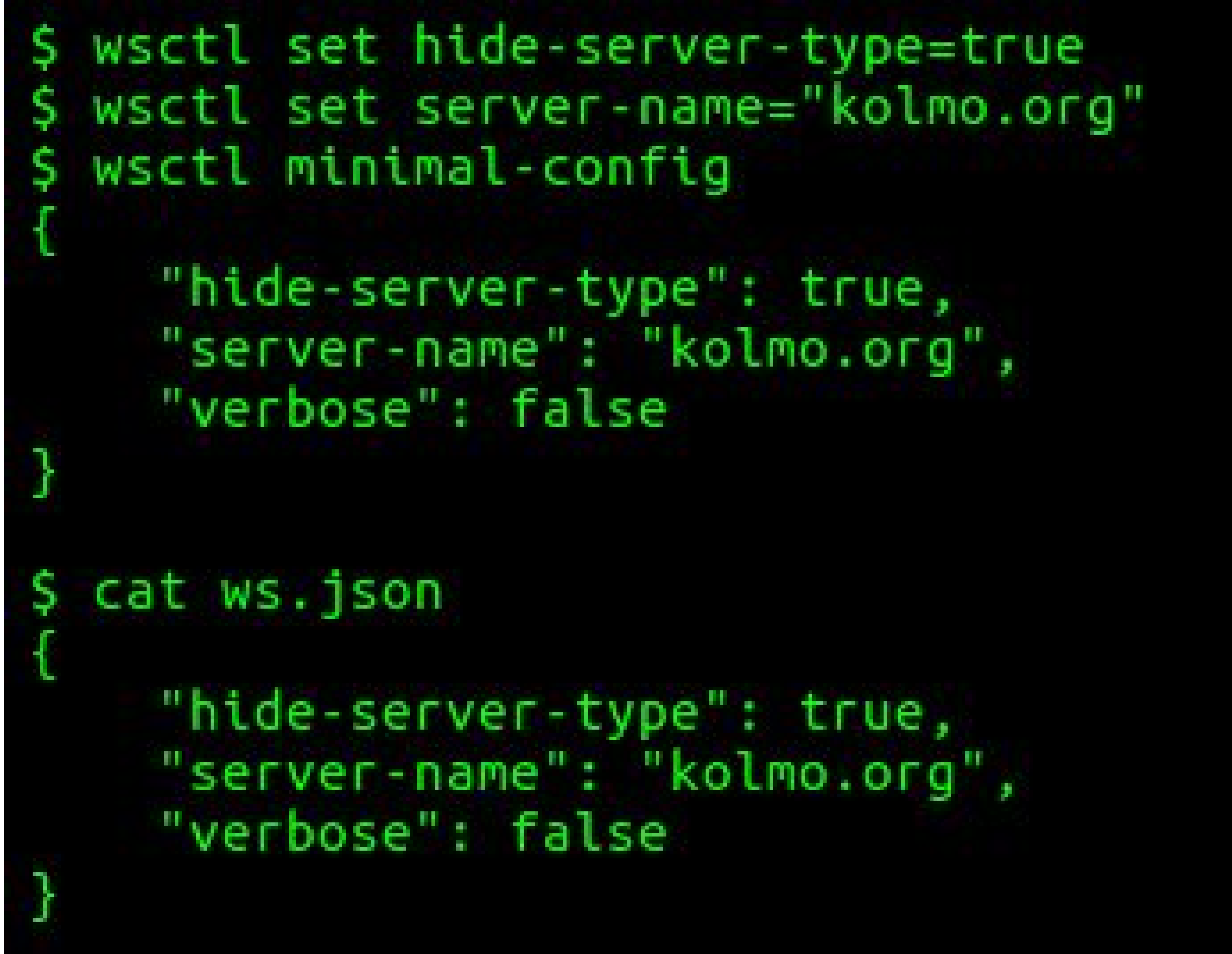
```
$ alias wsctl='kolctl --config ws.json --schema ws-schema.lua'

$ wsctl ls
carbon-server                                Send performance metrics to this IP address
client-timeout           5000                Timeout before client gets disconnected
hide-server-type         false               If we should hide server type
hide-server-version      false               If we should hide server version number
kolmo-server             127.0.0.1:1234      If we should launch a kolmo server
listeners                {struct}            Optional configurations per IP address listener
loggers                  {struct}            Loggers that log events and hits
max-connections          200                 Maximum number of versions
server-name              kolmo.org           Name this server reports as by default
sites                    {struct}            Sites we serve
verbose                  false               Perform verbose logging

$ wsctl ls sites
ds9a.nl                  {struct}
kolmo                    {struct}

$ wsctl ls sites/ds9a.nl
enabled                  true                If this site is enabled
listen                   {struct}            IP endpoints we listen on
name                     ds9a.nl             Hostname of this website
path                     /var/www/ds9a.nl    Path on filesystem where content is hosted
redirect-to-https        false               If all http requests should be redirected to https
```



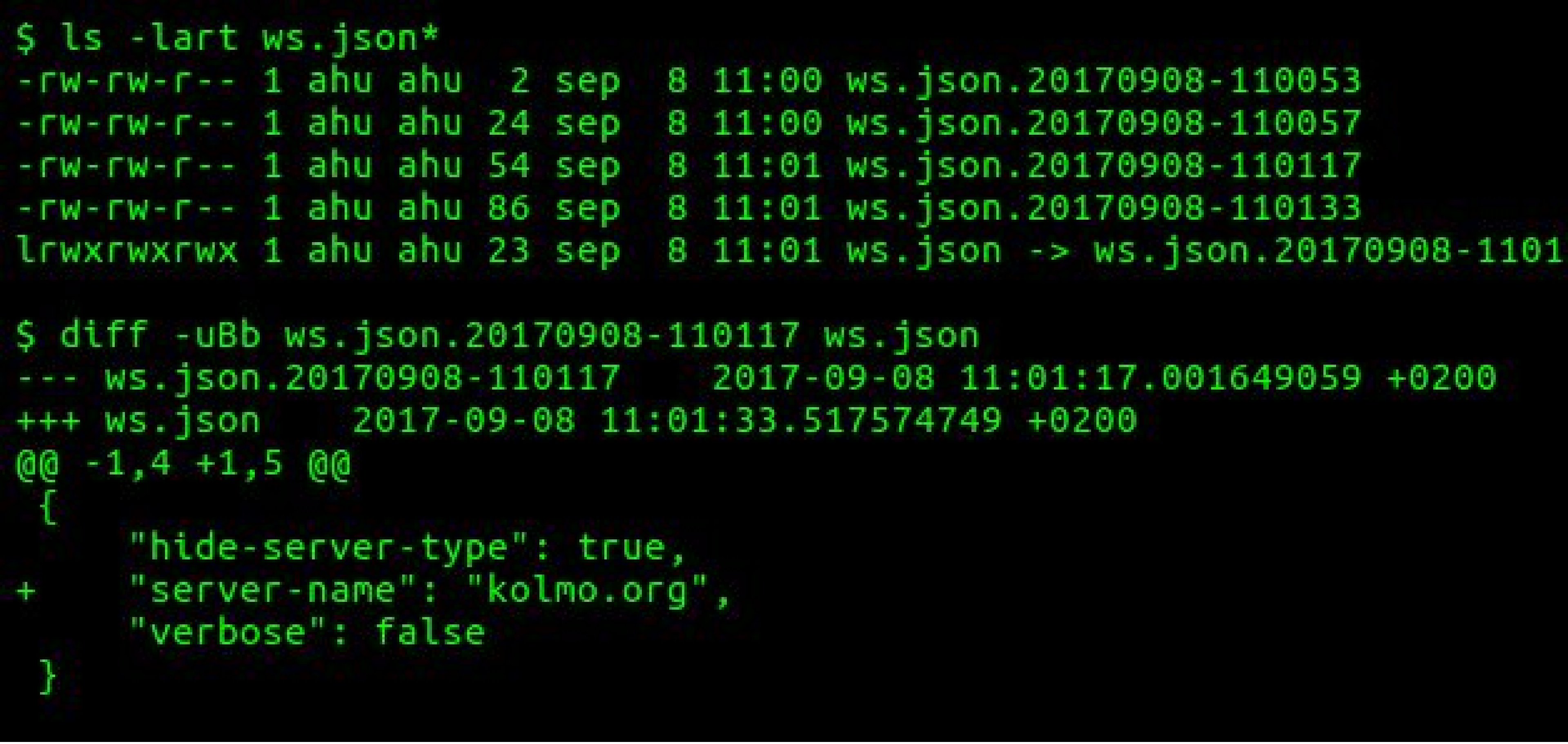
```
$ alias wsctl='kolctl --config ws.json --schema ws-schema.lua'

$ echo {} > ws.json

$ wsctl minimal-config
{}

$ wsctl full-config
{
  "carbon-server": "",
  "client-timeout": 5000,
  "hide-server-type": false,
  "hide-server-version": false,
  "kolmo-server": "127.0.0.1:1234",
  "listeners": {},
  "loggers": {
      "messages": {
          "log-errors": true,                "log-file": "",
          "log-hits": false,                 "log-warning": true,
          "syslog": true,                    "syslog-facily": "daemon"
      }
  },
  "max-connections": 200,
  "server-name": "",
  "sites": {},
  "verbose": true
}
```



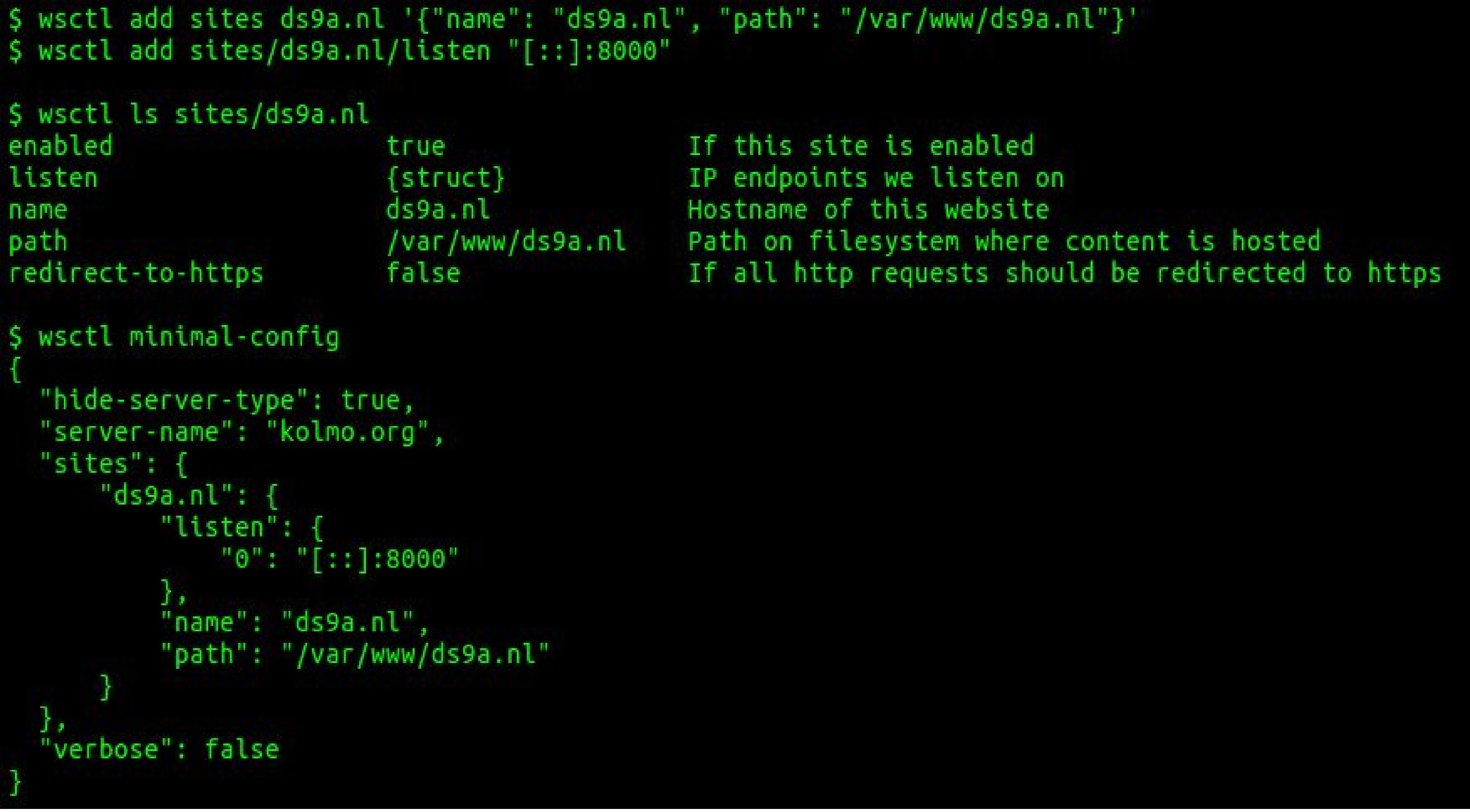
```
$ wsctl minimal-config
{}

$ wsctl set verbose=true

$ wsctl minimal-config
{}

$ wsctl set verbose=false
$ wsctl minimal-config
{
    "verbose": false
}
```



```
$ wsctl set hide-server-type=true
$ wsctl set server-name="kolmo.org"
$ wsctl minimal-config
{
    "hide-server-type": true,
    "server-name": "kolmo.org",
    "verbose": false
}

$ cat ws.json
{
    "hide-server-type": true,
    "server-name": "kolmo.org",
    "verbose": false
}
```

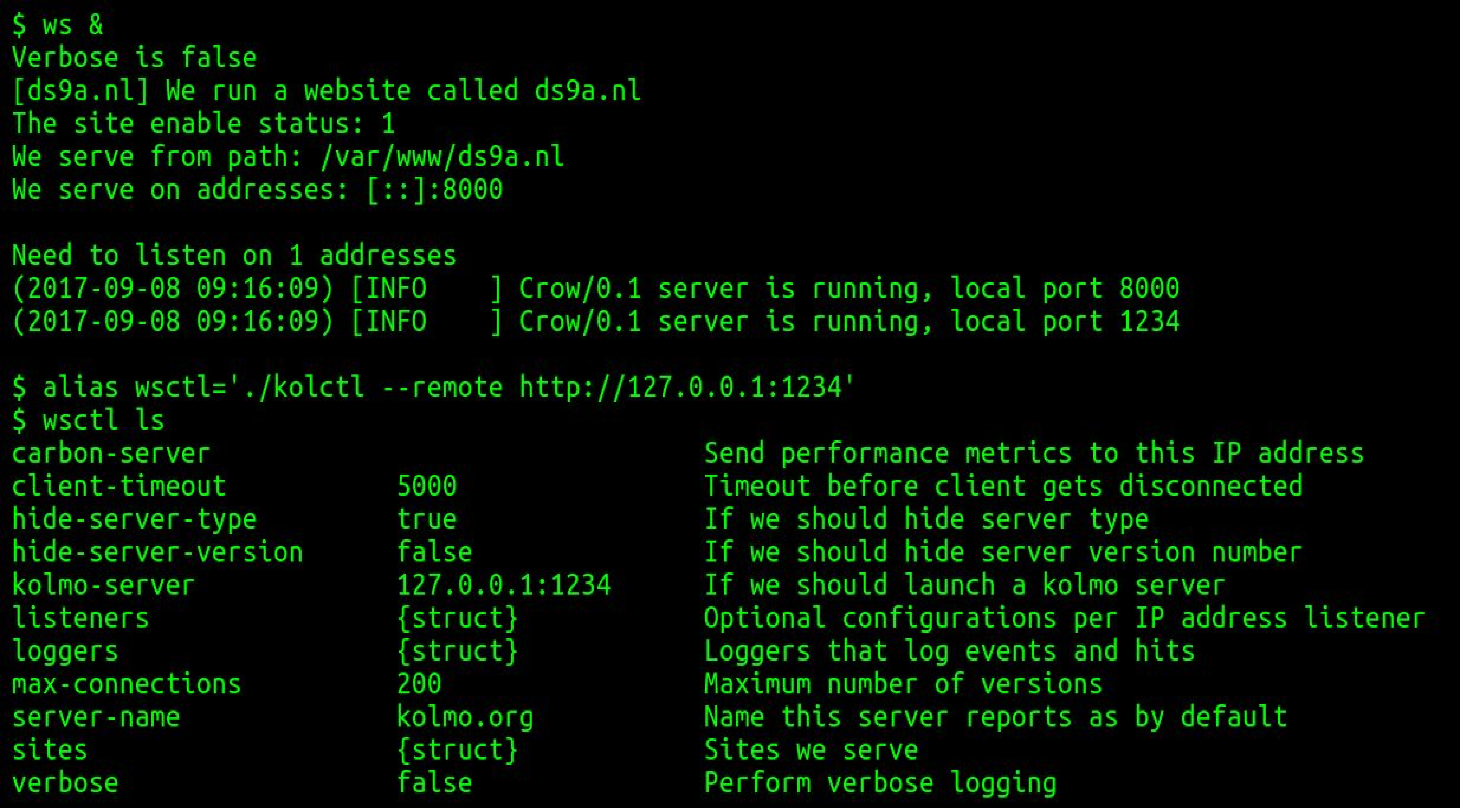


```
$ ls -lart ws.json*
-rw-rw-r-- 1 ahu ahu  2 sep  8 11:00 ws.json.20170908-110053
-rw-rw-r-- 1 ahu ahu 24 sep  8 11:00 ws.json.20170908-110057
-rw-rw-r-- 1 ahu ahu 54 sep  8 11:01 ws.json.20170908-110117
-rw-rw-r-- 1 ahu ahu 86 sep  8 11:01 ws.json.20170908-110133
lrwxrwxrwx 1 ahu ahu 23 sep  8 11:01 ws.json -> ws.json.20170908-1101

$ diff -uBb ws.json.20170908-110117 ws.json
--- ws.json.20170908-110117    2017-09-08 11:01:17.001649059 +0200
+++ ws.json    2017-09-08 11:01:33.517574749 +0200
@@ -1,4 +1,5 @@
 {
     "hide-server-type": true,
+    "server-name": "kolmo.org",
     "verbose": false
 }
```



```
$ wsctl add sites ds9a.nl '{"name": "ds9a.nl", "path": "/var/www/ds9a.nl"}'
$ wsctl add sites/ds9a.nl/listen "[::]:8000"

$ wsctl ls sites/ds9a.nl
enabled                  true                 If this site is enabled
listen                   {struct}             IP endpoints we listen on
name                     ds9a.nl              Hostname of this website
path                     /var/www/ds9a.nl     Path on filesystem where content is hosted
redirect-to-https        false                If all http requests should be redirected to https

$ wsctl minimal-config
{
  "hide-server-type": true,
  "server-name": "kolmo.org",
  "sites": {
      "ds9a.nl": {
          "listen": {
              "0": "[::]:8000"
          },
          "name": "ds9a.nl",
          "path": "/var/www/ds9a.nl"
      }
  },
  "verbose": false
}
```

Runtime & Constraints



```
$ ws &
Verbose is false
[ds9a.nl] We run a website called ds9a.nl
The site enable status: 1
We serve from path: /var/www/ds9a.nl
We serve on addresses: [::]:8000

Need to listen on 1 addresses
(2017-09-08 09:16:09) [INFO    ] Crow/0.1 server is running, local port 8000
(2017-09-08 09:16:09) [INFO    ] Crow/0.1 server is running, local port 1234

$ alias wsctl='./kolctl --remote http://127.0.0.1:1234'
$ wsctl ls
carbon-server                                Send performance metrics to this IP address
client-timeout           5000                Timeout before client gets disconnected
hide-server-type         true                If we should hide server type
hide-server-version      false               If we should hide server version number
kolmo-server             127.0.0.1:1234      If we should launch a kolmo server
listeners                {struct}            Optional configurations per IP address listener
loggers                  {struct}            Loggers that log events and hits
max-connections          200                 Maximum number of versions
server-name              kolmo.org           Name this server reports as by default
sites                    {struct}            Sites we serve
verbose                  false               Perform verbose logging
```

```
$ wsctl delta-config
{}

$ wsctl set verbose=true
{"result":"ok"}

$ wsctl delta-config
{
    "verbose": true
}

$ wsctl set max-connections=300
{"reason":"Attempting to change a variable at runtime
 that does not support runtime changes","result":"failure"}

$ wsctl set verbose=MORE
{"reason":"Attempt to set bool to something not true
  or false","result":"failure"}
```

```
$ wsctl set client-timeout=0
{"reason":"Timeout must be at least one millisecond","result":"failure"}

$ wsctl set carbon-server=192.168.123.321:2
{"reason":"Unable to convert presentation address '192.168.123.321:2'","result":"failure"}

$ less ws-schema.lua

main:registerVariable("client-timeout", "integer",
                         {
                             default="5000",
                             runtime="true",
                             unit="milliseconds",
                             description="Timeout before client gets disconnected",
                             check=
                              'if(x < 1) then error("Timeout must be at least one millisecond") end
                         })

main:registerVariable("carbon-server", "ipendpoint",
      { default="",
        runtime="true",
        description="Send performance metrics to this IP address"
      })
```

Dealing with new versions

- New versions of software bring new configuration settings
- And worse, changed defaults of old configuration settings
  - And often, this change is an improvement
- Can also simply be incompatible
- Kolmo procedure: **kolctl full-config > full.json** of “version 1.0” software
- Next up: **kolctl --schema version2-schema.lua --config full.json minimal-config**
- Suddenly, your ‘old’ defaults stick out as explicit configuration items in new version
- If your old configuration conflicts with new schema, will tell you where

Some usecases

- Webserver is misbehaving “all of a sudden”
  - Run kolctl delta-config on running process: gives all runtime changes versus startup configuration
  - Discover someone helpfully changed the certificate file to the wrong path, at runtime

- Webserver is misbehaving “all of a sudden”
  - “delta-config” shows nothing interesting
  - “ls -l ws.json*” however shows a new configuration was created 5 minutes ago
  - “diff -uBb ws.json.20170908-1245 ws.json” shows client-timeout was set to 1 millisecond
    - By operator who had ignored the helpful “units” output in “kolctl ls”

```
int main(int argc, char** argv)
{
  KolmoConf kc;
  kc.initSchemaFromFile("ws-schema.lua");
  kc.initConfigFromJSON("ws.json");
  kc.declareRuntime();
  kc.initConfigFromCmdline(argc, argv);

  kc.d_main.tieBool("verbose", &g_verbose);

  if(g_verbose) {
    cerr<<"Must be verbose"<<endl;
    cerr<<"Server name is "<<kc.d_main.getString("server-name")<<endl;
  }
  else {
    cerr<<"Verbose is false"<<endl;
  }
```

Relation to Automation

For “Kolmo” enabled services

1. Instead of hacks to manipulate configuration file, execute ‘kolctl’ commands on server

2. Alternatively, deploy the ‘minimal-config’ you got from your Kolmo-enabled service when it was “Just Right” (but mind the ‘migration’ slide)

FAQ

- Q: Yet another configuration file parser!
  A: It is nothing of the sort. World does not need more configuration file parsers. World needs configuration files with schemas, versioning, APIs for online and offline use

- Q: My playbooks already do this
  A: Likely. But I want to make it a lot easier for you so you don't have to depend on third party modules

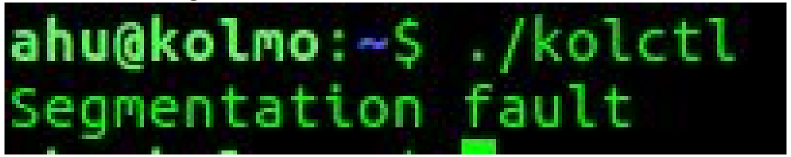
- Q: Windows did this already!
  A: True - and they let it die:
  https://en.wikipedia.org/wiki/Windows_Management_Instrumentation

Status

- Working prototype, running code!
    - https://kolmo.org/ & https://github.com/ahupowerdns/kolmo

- Tremendous fun to work with, concept works for me as a programmer
    - (I'm not ever writing or even USING a configuration file parser ever again)

- Quality of code: this segmentation fault is your hint



```
ahu@kolmo:~$ ./kolctl
Segmentation fault
```

- Only available for C++ 2014

- GOAL: Get everyone hot for this concept

Now what?

- Please join in!
- If you are a developer: share your thoughts on the API
- As an automation person: how does this fit in your life? Has it been done?
- As software users: get annoyed that all your favourite servers do not come with a configuration schema file!
  - You’ll miss ‘wctl ls’ from today onwards!

Kolmo

Making automation & configuration easier

bert hubert

https://kolmo.org/

@PowerDNS_Bert

POWERDNS
AN OX COMPANY