Distill

Table of Contents

Distill

Installing Distill

You can either run Distill on the same server as Puppet or runt as a standalone server.

1. Running on Pupppet server
2. Standalone using Web Services

Running on Puppet server

Example 1. Install

```
# yum install -y distill
```

Create initial directory structure.

```
$ mkdir -p /etc/distill/input/{host,host_group}
$ mkdir -p /etc/distill/template/{shared,staged}/{default,operatingsystem,operatingsystemrelease}
```

Configure Puppet with an external ENC.

Example 2. /etc/puppet/puppet.conf

```
[master]
   node_terminus = exec
   external_nodes = /usr/bin/enc-lookup
```

Configure Puppet REST API permissions.

Example 3. /etc/puppet/auth.conf

```
path /fact
method find
auth no
allow *

path /certificate_status
method find
auth no
allow *
```

After this you should only need to restart the Puppet services.

Standalone server using Web Services

Configuring Puppet Server

Example 4. Install

```
# yum install -y distill
```

Configure Puppet with an external ENC.

Example 5. /etc/puppet/puppet.conf

```
[master]
    node_terminus = exec
    external_nodes = /usr/bin/enc-lookup
```

Configure Distill to query Distill server.

Example 6. /etc/distill/distill.conf

```
web-lookup = true
url = http://<server>/distill
```

Configure Puppet REST API permissions.

Example 7. /etc/puppet/auth.conf

```
path /fact
method find
auth no
allow *

path /certificate_status
method find
auth no
allow *
```

After this you should only need to restart the Puppet services.

Configuring Distill server

Example 8. Install

```
# yum install -y distill
```

Create initial directory structure.

```
$ mkdir -p /etc/distill/input/{host,host_group}
$ mkdir -p /etc/distill/template/{shared,staged}/{default,operatingsystem,operatingsystemrelease}
```

After this you should enable and start the services.

Example 9. Enable and start services

```
# chkconfig httpd on
# service httpd start
```

Example 10. Verify Web Service

```
# enc-lookup -w <host>
# export API_URI="http://<server>/distill/index.pl"
# curl "$API_URI/client/enc/yaml?host=<host>"
```

Puppet Client

Example 11. Verify Puppet Server

```
# puppetd --server <server> --pluginsync --waitforcert 60 --test --noop
```

Distill file structure

This is the default file structure for Distill.

```
/etc/distill/
    input/
        host/
        host_group/
        host_location/
    template/
        shared/
        staged/
```

Input for a specific host

Host input allows for per host input parameters like ip-address, network, owner etc.

Example 12. input/host/myhost.mydomain.json

```
{
    "ip-address": "192.168.0.2",
    "network": "192.168.0.0/24",
    "owner": "networking"
}
```

Create a group of hosts

Host group provides a way to group hosts together. A host can belong to one or more groups.

Example 13. input/host_group/mysql_server.json

```
{
    "name": "MySQL Server",
    "hosts": [
        "mysql1.mydomain.com",
        "mysql2.mydomain.com",
        "mysql3.mydomain.com"
    ]
}
```

Shared templates folder

Shared templates will be the same for all environments example: Dev., QA, Prod. This is useful when you have configuration that needs to be available in production immediately. Normally it's better to avoid shared configuration unless you really need it. The normal use case is host/host_group templates, templates that affect a larger number of machines it's usually advisable to stage the configuration.

Staged templates folder

Staged configuration will only be published to the Dev. environment and once tested it can be promoted to QA and Prod.

Distill Templates

Templates are always parsed in a predefined order substituting any previous key/value pairs.

For calling classes you append double colon **::** anything after this will be interpreted as an argument to this class.

Example 14. call Class without arguments

```
{
	"bashrc": null
}
```

This would work for a Puppet class defined as.

```
class bashrc {
```

Example 15. call Class with parameter

```
{
	"timezone::timezone": "Europe/Zurich"
}
```

This would work for a Puppet class defined as.

```
class timezone($timezone) {
```

For this example I will add the following substitution order for Distill.

Example 16. /etc/distill/distill.conf

```
sequence = network, region, country, city, datacenter, owner, host
```

So first we create a template for the network, so we can provide location information.

Example 17. template/shared/network/192.168.0.0_24.json

```
{
	"subnet": "255.255.255.0",
	"gateway": "192.168.0.1",
	"region": "Europe",
	"country": "Switzerland",
	"city": "Zurich",
	"datacenter": "Plex"
}
```

Then we use the location information provided by the network to set regional specific settings.

Example 18. template/shared/region/europe.json

```
{
	"timezone::timezone": "CET"
	"resolv::dns_servers": [
		"192.168.0.5",
		"192.168.0.6"
	],
	"sendmail::mail_server": "smtp.mydomain.com",
	"ntp::ntp_servers": [
		"ntp1.eu.mydomain.com",
		"ntp2.eu.mydomain.com",
		"ntp3.eu.mydomain.com"
	]
}
```

Then we unset sendmail for this host.

Example 19. template/shared/host/myhost.mydomain.json

```
{
    "u:sendmail::mail_server": null
}
```

The end result of this will be.

Example 20. JSON ENC

```
{
    "classes": {
        "timezone": {
            "timezone": "CET"
        },
        "resolv": {
            "dns_servers": [
                "192.168.0.5",
                "192.168.0.6"
            ]
        },
        "ntp": {
            "ntp_servers": [
                "ntp1.eu.mydomain.com",
                "ntp2.eu.mydomain.com",
                "ntp3.eu.mydomain.com"
            ]
        }
    },
    "parameters": {
        "subnet": "255.255.255.0",
        "gateway": "192.168.0.1",
        "region": "Europe",
        "country": "Switzerland",
        "city": "Zurich",
        "datacenter": "Plex"
    }
}
```

Distill operators

Distill currently support 4 different operations substitution, unset, merge and immutable.

Table 1. Distill operators

Operator	Action	Description
!:	Unset	Unset key (DEPRECATED use u:)
u:	Unset	Unset key, list item or hash key
m:	Merge	Merge array or hash
i:	Immutable	Prevent a key from being changed
e:	Expand	Expand value from another key (DEPRECATED use r: or c:)

Operator	Action	Description
r:	Reference	Reference value from another key
c:	Copy	Copy value from another key
iu:	Immutable + Unset	Combine immutable and unset
im:	Immutable + Merge	Combine immutable and merge
ie:	Immutable + Expand	Combine immutable and expand (DEPRECATED use ir: or ic:)
ir:	Immutable + Reference	Combine immutable and reference
ic:	Immutable + Copy	Combine immutable and copy

Distill Operators examples

Example 21. Unset a key

```
{
    "u:datacenter": null
}
```

Example 22. Unset a list item

```
{
    "u:ntp::ntp_servers": [
        "ntp3.eu.mydomain.com"
    ]
}
```

Example 23. Unset a hash key

```
{
    "u:users": {
        "jdoe": null
    }
}
```

Note

Unset doesn’t support unsetting keys inside nested hashes.

Example 24. Merge list’s

```
{
    "m:ntp::ntp_servers": [
        "ntp4.eu.mydomain.com"
    ]
}
```

Example 25. Merge hashes

```
{
      "m:users": {
            "mpersson": {
                  "uid": "500",
                  "gid": "500",
                  "name": "Michael Persson",
                  "home": "/home/mpersson",
                  "shell": "/bin/bash"
            }
      }
}
```

Example 26. Immutable

```
{
      "i:ntp::ntp_servers": [
                        "ntp1.eu.mydomain.com",
                        "ntp2.eu.mydomain.com",
                        "ntp3.eu.mydomain.com"
                  ]
}
```

Note

This means that no other template can replace or change these values

Example 27. Expand

```
{
   "e:postfix::mail_server": "sendmail::mail_server"
}
```

Distill REST API

Example 28. Get host Puppet ENC in JSON format

```
# export API_URI="http://distill-dev/distill/index.pl"
# curl "$API_URI/client/enc/json?host=<fqdn>"
```

Example 29. Get host Puppet ENC in YAML format

```
# curl "$API_URI/client/enc/yaml?host=<fqdn>"
```

Example 30. Get all host Puppet ENC’s in JSON format

```
# curl "$API_URI/client/enc/json/all"
```

Example 31. Get host Hash in JSON format

```
# curl "$API_URI/client/json?host=<fqdn>"
```

Example 32. Get host Hash in YAML format

```
# curl "$API_URI/client/yaml?host=<fqdn>"
```

Example 33. Get all host Hashes in JSON format

```
# curl "$API_URI/client/json/all"
```

Example 34. Get all hosts that has a specific parameter in JSON format

```
# curl "$API_URI/client/has/parameter/json?parameter=sendmail::mail_server"
```

Example 35. Get all hosts that has a specific parameter and value in JSON format

```
# curl "$API_URI/client/has/parameter/json?parameter=sendmail::mail_server=smtp.mydomain.com"
```

Example 36. Get all hosts that has a specific class

```
# curl "$API_URI/client/has/class/json?class=sendmail"
```

Example 37. Get all hosts

```
# curl "$API_URI/client/cached/json/all"
```

Example 38. Get all hosts changed since a specific time ago

```
# curl "$API_URI/client/cached/changed/json?changed_since=5-days-ago"
```

Table 2. Accepted values

Value
x-sec-ago
x-min-ago
x-hour-ago
x-hours-ago
x-day-ago
x-days-ago
x-week-ago
x-weeks-ago

Distill environments

In order to support multiple environments that you might be using in Puppet. Distill will parse the Fact environment for the host and use this to allow overrides in the configuration file like this:

Example 39. /etc/distill/distill.conf

```
[main]
sequence = default, region, country, datacenter, owner, host_group, host
user = apache
group = apache
# Allow override environment from host Fact
override-environment = true

[lookup]
web-lookup = true
url = http://localhost/distill

[facter]
facts = macosx_productversion_major

# Override basedir and sequence for environment macosx
[macosx]
basedir = /etc/distill/macosx
sequence = default, region, country, datacenter, macosx_productversion_major, host_group, host
```

Distill Schema

Background

Distill allows you to generate configuration using hierarchical substitution and use this as an input for Puppet through the Puppet YAML ENC.

However this doesn't guarantee that the configuration provided is formated correctly. Therefore it is recommended to introduce a validation step for the configuration. This is where Distill Schema is usefull it can validate Distill configuration using JSON schemas, this insures the data structure is correct and you can also validate entries using regular expression.

The JSON schema files are stored in the /etc/distill/schema directory and is split up per module. This makes it easy to review and change the schemas for each module.

When Distill Schema is run, it will merge all the module schemas to one schema and use this to validate the output from Distill.

Structure

Site

For each site there is one site schema that has the basic structure and all parameters.

Example 40. schema/site.json

```
{
  "type": "object",
  "additionalProperties": false,
  "properties": {
      "classes": {
          "type": "object",
          "additionalProperties": true,
          "properties": {
          }
      },
      "parameters": {
          "type": "object",
          "additionalProperties": true,
          "properties": {
          }
      }
  }
}
```

Module

For each module there is one schema file.

Example 41. schema/module/bashrc.json

```
{
   "type": "object",
   "additionalProperties": false,
   "properties": {
      "bashrc": {
         "type": "null"
      }
   }
}
```

type

Type defines which type of object we're validating this is usually "string", "number", "null", "object", "array".

string

A string is a normal text string.

number

A number is a normal decimal number.

null

null is an empty entry.

object

An object is a hash i.e. key/value pairs.

array

An array is a list of entries.

additionalProperties

So for the site schema we're first defining an object (hash) that must contain 2 entries. The reason for this is that we set additionalProperties to false which means it won't allow any additional entries.

properties

Properties defines a number of sub-entries in a hash.

Module schemas

If you look underneath the "classes" object, this is where class JSON schemas will be merged. Since "additionalProperties" is set to true it will allow additional entries. Schemas from modules will be merged here.

Example from bashrc module that doesn't take any parameters.

```
{
   "type": "object",
   "additionalProperties": false,
   "properties": {
      "bashrc": {
         "type": "null"
      }
   }
}
```

First we're defining a has hash that doesn't allow for any additional entries.

So this would validate true for the following JSON ENC input.

```
{
   "bashrc": null
}
```

Schema with parameters

In the following example we allow the passing of additional parameters to ssh configuration files.

The class definition in Puppet looks as following.

```
class ssh($ssh_opts = undef, $sshd_opts = undef) {
```

The JSON ENC looks like the following, it's a list of options.

```
{
   "ssh": {
      "sshd_opts": [
         "PermitRootLogin no",
         "PasswordAuthentication no"
      ]
   }
}
```

To verify this we have to use a new statement **items** since we're passing an array.

```
{
   "type": "object",
   "additionalProperties": false,
   "properties": {
      "ssh": {
         "type": [ "object", "null" ],
         "additionalProperties": false,
         "properties": {
            "ssh_opts": {
               "type": "array",
               "items": {
                  "type" : "string"
               }
            },
            "sshd_opts": {
               "type": "array",
               "items": {
                  "type" : "string"
               }
            }
         }
      }
   }
}
```

items

Validate items in a list.

Schema with nested hashes

A more complicated example is when you have nested data types like a hash in a hash.

```
{
   "ssh::auth_keys": {
      "keys": {
         "infrastructure": {
            "user": "root",
            "type": "ssh-dss",
            "key": "AAAAB3NzaC1kc3..."
         }
      }
   }
}
```

The class definition in Puppet looks as following.

```
class ssh::auth_keys($keys, $overwrite = false) {
```

To do this we basically replicate the same structure as in the input inside the Schema.

```
{
   "type": "object",
   "additionalProperties": false,
   "properties": {
      "ssh::auth_keys": {
         "type": "object",
         "additionalProperties": false,
         "properties": {
            "keys": {
               "type": "object",
               "patternProperties": {
                  ".*": {
                     "type": "object",
                     "additionalProperties": false,
                     "properties": {
                        "user": {
                           "type": "string",
                           "required": true,
                           "pattern": "^[a-z0-9\\-_]+$"
                        },
                        "type": {
                           "type": "string",
                           "required": true,
                           "enum": [ "ssh-dss" ]
                        },
                        "key": {
                           "type": "string",
                           "required": true
                        }
                     }
                  }
               }
            },
            "overwrite": {
               "type": "boolean"
            }
         }
      }
   }
}
```

You'll notice I'm using a new statement **patternProperties** this allows matching sub entries using a regex.

patternProperties

patternProperties defines a number of sub-entries in a hash using regular experssion.

required

Specifies if an entry is required or not.

pattern

Allows for regex validation of the entry.

Links

IETF JSON Schema

IETF JSON Schema [http://tools.ietf.org/html/draft-zyp-json-schema-03]

Example Schemas

modules/ssh/schema/ssh_schema.json

```
{
  "type": "object",
  "additionalProperties": false,
  "properties": {
    "ssh": {
      "type": [ "object", "null" ],
      "additionalProperties": false,
      "properties": {
        "ssh_opts": {
          "type": "array",
          "items": {
            "type" : "string"
          }
        },
        "sshd_opts": {
          "type": "array",
          "items": {
            "type" : "string"
          }
        }
      }
    },
    "ssh::auth_keys": {
      "type": "object",
      "additionalProperties": false,
      "properties": {
        "keys": {
          "type": "object",
          "patternProperties": {
            ".*": {
              "type": "object",
              "additionalProperties": false,
              "properties": {
                "user": {
                  "type": "string",
                  "required": true,
                  "pattern": "^[a-z0-9\\-_]+$"
                },
                "type": {
                  "type": "string",
                  "required": true,
                  "enum": [ "ssh-dss" ]
                },
                "key": {
                  "type": "string",
                  "required": true
                }
              }
            }
          }
        },
        "overwrite": {
          "type": "boolean"
        }
      }
    }
  }
}
```

Command Reference

distill

NAME

distill - Host template engine for Puppet

SYNOPSIS

distill -h

distill [-c config] [-b basedir] hostname

distill [-c config] [-b basedir] -u

DESCRIPTION

Host template engine for Puppet, will generate a JSON file for each host that contains all host configuration.

OPTIONS

-h, --help	Display help.
-d, --debug	Debug.
-s, --silent	Silent mode, don't print YAML ENC.
--print-config	Print configuration.
-ph, --puppet-hosts	Print all Puppet hosts.
-ah, --all-hosts	Print all_hosts.
-ch, --changed-hosts=*TIME*	Print changed_hosts since x-sec-ago, x-min-ago, x-hours-ago, x-days-ago and x-week-ago.
--diff	Can be used in combination with -ch and it will print a diff of the differences for each host.
-hp, --has-parameter=*PARAM*	Print hosts that have parameter x. yum_server or yum_server=yum.
-hc, --has-class=*CLASS*	Print hosts that have class x.
-c, --config=*CONFIG*	Configuration file, defaults to /etc/distill/distill.conf.
-l, --logfile=*LOGFILE*	Log file, defaults to /var/puppetmaster/distill.log. Must have same permissions as puppetmasterd, since distill is called by puppetmasterd.

-b, --basedir=*BASEDIR*	Base directory, defaults to /etc/distill.
-o, --outputdir=*OUTPUTDIR*	Output directory, defaults to /var/lib/distill.
-p, --puppet-server=*SERVER*	Puppet server, defaults to localhost.
-u, --update	Update configuration for all hosts.
-e, --environment=*ENVIRONMENT*	Distill environment, defaults to production.
-oe, --override-environment	Override Distill environment based on Puppet Facts.
--user	User that will be used to run the application, defaults to puppetmaster.
--group	Group that will be used to run the application, defaults to puppetmaster.
hostname	Add configuration for machine with specified hostname.

EXIT STATUS

0	Success
1	Failure

FILES

/etc/distill/distill.conf	Default configuration file.
/var/puppetmaster/distill.log	Default log file.
/etc/distill	Default base directory for template and validate JSON files.
/var/lib/distill	Default output directory for client JSON files.

AUTHOR

Michael Persson

COPYING

enc-lookup

NAME

enc-lookup - Host template lookup for Puppet

SYNOPSIS

enc-lookup -h

enc-lookup [-c config] [-b basedir] hostname

enc-lookup [-c config] [-b basedir] -u

DESCRIPTION

Host template lookup for Puppet, will display JSON file for each host as a Puppet ENC.

OPTIONS

-h, --help	Display help.
-c, --config=*CONFIG*	Configuration file, defaults to /etc/distill/distill.conf.
-l, --logfile=*LOGFILE*	Log file, defaults to /var/puppetmaster/distill.log. Must have same permissions as puppetmasterd, since distill is called by puppetmasterd.
-b, --basedir=*BASEDIR*	Base directory, defaults to /etc/distill.
-w, --web-lookup	Lookup using Web service, defaults to False.
-u, --url	URL for Web service, defaults to http://localhost/distill.
-e, --environment=*ENVIRONMENT*	Distill environment, defaults to production.
-oe, --override-environment	Override Distill environment based on Puppet Facts.
--user	User that will be used to run the application, defaults to puppetmaster.
--group	Group that will be used to run the application, defaults to puppetmaster.
hostname	Add configuration for machine with specified hostname.

EXIT STATUS

0 Success

1 Failure

FILES

/etc/distill/distill.conf	Default configuration file.
/var/puppetmaster/distill.log	Default log file.
/etc/distill	Default base directory for template and validate JSON files.
/var/lib/distill	Default output directory for client JSON files.

AUTHOR

Michael Persson

COPYING

distill.conf

NAME

distill.conf - Configuration file for Distill, template engine for Puppet

SYNOPSIS

/etc/distill.conf

DESCRIPTION

Configuration file for Distill. Host template engine for Puppet, will generate a JSON file for each host that contains all host configuration.

OPTIONS

[main]	Main section.
basedir=*BASEDIR*	Directory where configuration, templates and validation is stored. **Default:** basedir = /etc/distill
outputdir=*OUTPUTDIR*	Directory where output is stored. **Default:** outputdir = /var/lib/distill
logfile=*LOGFILE*	Log file, must have same permissions as puppetmasterd or apache depending if it runs directly from Puppet or as a Web Service. **Default:** logfile = /var/log/distill/distill.log
sequence=*SEQUENCE*	Determines the sequence substitution is performed. **Default:** sequence = region, country, city, datacenter, owner, environment, host
user=*USER*	User that will be used to run the application. **Default:** user = puppetmaster
group=*GROUP*	Group that will be used to run the application. **Default:** group = puppetmaster
environment=*ENVIRONMENT*	Distill environment. **Default:** environment = production
override-environment=*TRUE/FALSE*	Override Distill environment based on Puppet Facts. **Default:** override-environment = false
[lookup]	Lookup section.
web-lookup=*TRUE/FALSE*	Lookup using Web service. **Default:** web-lookup = false
url=*URL*	URL for Web service. **Default:** url = http://localhost/distill

[facter] Facter section.

facts=*FACTS* Facts that are exposed to Distill. **Default:** facts = operatingsystem, operatingsystemrelease

use-host-group=*TRUE/FALSE* Use host groups derived from a Fact. **Default:** use-host-group = false

host-group=*HOST GROUP FACT* Fact to use for host groups, must be as a comma separated list. **Default:** host-group = *host_group*

convert-to-array=*FACT* Fact to convert to an array, must be as a comma separated list

[regex] Regex section. Can be used to validate input using regexp. **Example:**

AUTHOR

Michael Persson

COPYING

distill_schema

NAME

distill_schema - Validate Distill configuration

SYNOPSIS

distill_schema -h

distill_schema --print-schema

distill_schema [-a] [-d] [-b BASEDIR] [-s SERVER] [-p SERVER] [--host HOST]

DESCRIPTION

Validate Distill configuration using JSON schemas.

OPTIONS

-h, --help	Display help.
-b, --basedir=*BASEDIR*	Distill base directory, defaults to /etc/distill.
--host=*HOST*	Host to validate.
-a, --all-hosts	All hosts
-s, --server=*SERVER*	Distill server, defaults to distill.
-p, --puppet-server=*SERVER*	Puppet server (Only required for the -a option), defaults to puppet.
--print-schema	Print schema.
-d, --debug	Debug.
-h, --help	Help.

EXIT STATUS

0 Success

1 Failure

FILES

/etc/distill/schema/site_schema.json	Puppet default site schema.
/etc/distill/schema/module/<module>.json	Puppet module schemas.

AUTHOR

Michael Persson

COPYING