yadt 1.8

cheat sheet 0.3

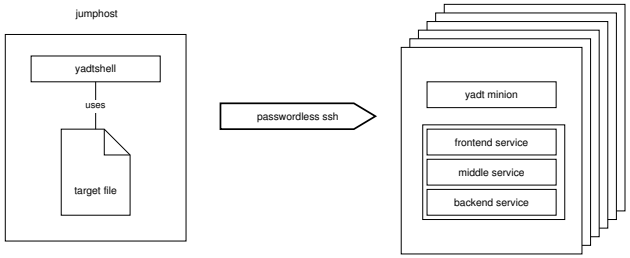
http://www.yadt-project.org/

Concept

Using yadtshell you can orchestrate high level operations like updating a group of hosts, as well as stopping and starting relevant and dependent services in the correct order.

The most important command is `update`.
After running `update` successfully, all hosts are up-to-date and all services are up and running.

Note: In case of a problem (i.e. a command terminates with exit code != 0), yadtshell stops immediately. Subsequent calls of the same cammend continue where the previous call stopped.



Note: It is required that the hosts are accessible via passwordless ssh and provide a yadt minion (the counterpart to the yadtshell).

Definition: Component URI

compontent uris

```
host://<hostname>
service://<hostname>/<name>
artefact://<hostname>/<name>/<version>
```

Examples

```
host://hostname
service://hostname/tomcat6
artefact://hostname/web-application/0:1.23
```

Note: Components are *always* host-specific.

Brace Expression

```
artefact://{hostname01|hostname03}/myapp
```

Range Expression

```
host://hostname0[1..3]
```

Wildcards

```
service://hostname/*
```

yadt.conf.d (directory)

The yadt minion gets configured via `*.yaml` files in `/etc/yadt.conf.d`; they get merged in alphanumeric order.

Note: Indented blocks have to start with *4 blanks*. Do *not* use tabs.

/etc/yadt.conf.d/my-services.yaml

```
services:
    frontend:
        needs_services: [middleservice1]
        is_frontservice: true
    middleservice1:
        needs_services: [middleservice2]
```

The service name must be equal to the corresponding name of the service script (as found in `/etc/init.d`).

is_frontservice is a marker for the status overview. The status (shown in percentage) of the target will be calculated by determining how many frontservices are running.

needs_services the services that have to be running before starting this service (reverse for stopping)

The service definition may also contain a complete component URI as string, which describes a service on another host, e.g.

need_services

```
needs_services: ['service://hostname/service']
```

Note: This notation only allows the *hostname*, not the full qualified domain name. Yadtshell extracts the hostname from the fqdn as the string until the first dot.

Please see the host configuration section of the yadtshell wiki for more information.

target (file)

yadtshell uses a yaml file named `target` in the current working directory to define a yadt target (set of hosts), e.g.

./target

```
hosts:
- hostname1.spam.eggs
- hostname2.spam.eggs
- hostname*.spammy.eggs
- hostname0[1..3].foo.bar
```

It is possible to group your hosts within a target:

./target

```
hosts:
- hostname1.spam.eggs  hostname2.spam.eggs
- hostname3.foo.bar  hostname4.foo.bar
```

This will change the way the hosts will be displayed.

Using the yadtshell

start yadtshell

```
> init-yadtshell
```

- activates autocompletion for component uris,
- allows to omit `yadtshell` when executing a yadtshell commands.

To restores your shell environment you can use `CTRL+D` or

stop yadtshell

```
> deactivate
```

Yadtshell Commands

Use the `yadtshell` command if you prefer to execute yadtshell commands without entering the yadtshell itself:

general usage

```
> yadtshell [options] <command> [<uri> ...]
```

-v verbose

–dryrun no actions executed (just logging)

-n same as dryrun

Status Information

To retrieve the status of all services and artefacts versions from the current target use:

fetch status

```
> status
```

This will also perform `info`, which displays a summary of all services for each host within the current target:

show status

```
> info [––full]
```

–full shows complete information (artefacts of hosts, etc.)

To display low-level data of components (in yaml format) use

dump json data

```
> dump [uri–query0 [uri–query1 ...]]
```

additional arguments for `dump`:

–attribute

–show-pending-updates

–show-current-artefacts

dump raw data of all services

```
> dump service://
```

Note: The output of info and dump is generated using cached data.

Hosts

To prevent others from executing commands on a host it is possible to lock the host:

lock host

```
> lock –m "message" [––force] <host_uri> [...]
```

afterwards commands can only be executed

- by you,
- from the current target directory
- on the current host.

lock hostname01

```
> lock –m "I need this" host://hostname01
```

hijacking a locked host

```
> lock –m "hijacking" ––force host://*
```

Note: The message should reflect the reason why you are doing what you are doing and include your name as well

with message

```
> lock –m "updating host [Michael]" host://
    hostname31
```

To release a lock use:

unlock host

```
> unlock <host_uri> [<host_uri] ...]
```

release all your locks on all hosts

```
> unlock host://*
```

Services

To start a service, regarding its dependencies, use:

start service

```
> start <service_uri> [<service_uri> ...]
```

start all services

```
> start service://*
```

To stop a service and all services depending on the service:

stop service

```
> stop <service_uri> [<service_uri> ...]
```

Note: When stopping a service all services depending on this service will be stopped as well. But starting the service will *not* start the services depending on the service again.

If a service is currently out of order you can ignore the state of a service (e.g. assume all operations on that service are successful):

ignore service

```
> ignore –m "message" <service_uri> [...]
```

nagios server down, so ignore

```
> ignore –m "nagios server is down" service
    ://*/nagios
```

To unignore services on host use:

unignore service

```
> unignore <service_uri> [...]
```

Artefacts

To install updates (if there are any) and stop/start the defined services use:

update target

```
> update [<host_uri> ...] [–p <number>]
```

If you only want to update artefacts without restarting services, use `updateartefact`. Take care when using this command: it is ignoring all service dependencies.

updateartefact

```
> updateartefact <artefact_uri> [...]
```

yadtshell wiki at github
http://www.yadt-project.org/