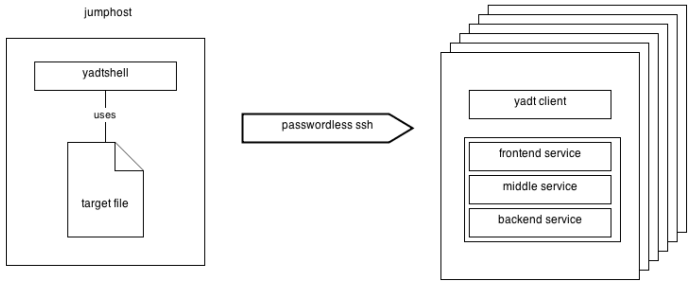
yadt 1.4

cheat sheet 0.2

http://www.yadt-project.org

concept



Using the yadtshell you can execute high level operations like updating a group of hosts . The only requirement is that the hosts are accessible via passwordless ssh and provide a yadt client.

Definition: Component URI

{artefact|host|service}://<hostname>[/<name>[/<version>]]

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

*Components are **always** host-specific.*

Brace Expansion

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

Range Expressions

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

Wildcards

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

yaml files

indented blocks have to start with **4 blanks**. Do not use tabs. Example where blanks are marked as "␣" :

```
-␣frontend:
␣␣␣␣needs_services:␣[middleservice1]
␣␣␣␣is_frontservice:␣true
```

yadt.services (file)

The yadt client uses a *yaml file* named /etc/yadt.services

```
- frontend:
    needs_services: [middleservice1]
    is_frontservice: true
- middleservice1:
    needs_services: [middleservice2]
- middleservice2:
    needs_services: [backendservice]
- backendservice:
```

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

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

```
- backendservice: ['service://hostname/servicename']
```

Please notice that 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.

target (file)

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

```
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:

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

this will change the way the hosts will be displayed.

view (file)

If you have a lot of hosts in a target you can use a *yaml-file* called view to configure the rendering of the status overview.

Place the view file together with the target file in the current working directory.

```
info-view: [matrix, color]
```

`matrix`	show status information in matrix
`color`	display status in color
`maxcols`	maximum number of columns
`3cols`	use three columns

Executing yadt commands

All involved hosts have to be accessible via ***passwordless ssh***.

1. Entering the yadtshell

Enter the yadtshell by calling

```
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

```
deactivate
```

2. Using yadtshell as a command

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

```
yadtshell [options] <command> [<component_uri> …]
```

`-v`	verbose
`-dryrun`	will print logging, but without execution
`-n`	same as dryrun

Status Information

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

```
status
```

this will also perform »**info**«, which displays a summary of all services for each host within the current target:

```
info [--full]
```

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

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

```
dump [uri-query0 [uri-query1 …]]
```

additional arguments for dump:

--attribute
--show-pending-updates
--show-current-artefacts

Example: *dump info of all services.*

```
dump service://
```

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 -m "message" [--force] <host_uri> [<host_uri> ...]
```

afterwards commands can only be executed by you, from the current target directory on the current host.

Example: *lock the host »hostname01«.*

```
lock -m "message" host://hostname01
```

Example: *hijacking a lock from somebody else*

```
lock -m "message" --force host://*
```

Attention: when using the *-m “message”* option, the message should reflect the reason why you are doing what you are doing and include your name as well:

```
lock -m "Need this host. [Michael]" host://hostname31
```

To release a lock use:

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

Example: *realase all of your locks on all target hosts.*

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

Services

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 -m "message" <service_uri> [<service_uri> ...]
```

Example: *ignore all nagios checks, since the nagios server is down.*

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

To *unignore* services on host use:

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

To start a service, regarding its dependencies, use:

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

Example: *start all services.*

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

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

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

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.

Artefacts

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

```
update <host_uri> [<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 <artefact_uri> [<artefact_uri> ...]
```

http://www.yadt-project.org