
Ajenti
Release 2.2.6

Sep 01, 2023

Users

1 Feature Overview	3
Python Module Index	69
Index	71

Ajenti platform includes following products:

- **Ajenti Core**, a Python library, the platform itself including the HTTP server, socket engine and plugin container.
- **Ajenti Panel**, a startup script and a set of stock plugins such as file manager, network configurator and service manager.

CHAPTER 1

Feature Overview

1.1 HTTP Server

- HTTP 1.1 Support.
- Websockets with fallback to XHR polling.
- Fast event-loop based processing.
- Flexible routing.
- Session sandboxing.
- SSL with client certificate authentication.

1.2 Performance

- >1000 requests per second.
- 30 MB RAM footprint + 5 MB per session.

1.3 API

- Highly modular Python API. Everything is a module and can be removed or replaced.
- Builtin webserver API supports routing, file downloads, GZIP, websockets and more.
- Transparent SSL client authorization.
- Plugin architecture
- Dependency injection
- Server-side push and socket APIs.

1.4 Security

- Pluggable authentication and authorization.
- Stock authenticators: UNIX account, password, SSL client certificate and Mozilla Persona E-mail authentication.
- Unprivileged sessions isolated in separate processes.
- Fail2ban rule

1.5 Frontend

- Clean, modern and responsive UI. Single-page, no reloads.
- Live data updates and streaming with Socket.IO support.
- Full mobile and tablet support.
- LESS and CSS, CoffeeScript and JavaScript auto-build support.
- Numerous stock directives.
- AngularJS templating.

1.6 Platforms

- Debian 9 or later
- Ubuntu Bionic or later
- RHEL 8 or later
- Can be run on other Linux or BSD systems with minimal modifications.
- Supports Python 3.5+.

1.6.1 Installing

Caution: Supported operating systems:

- Debian 9 or later
- Ubuntu Bionic or later
- RHEL 8 or later

Other Linux-based systems *might* work, but you'll have to use manual installation method.

Automatic Installation

```
curl https://raw.githubusercontent.com/ajenti/ajenti/master/scripts/install.sh | sudo -E bash -s -
```

Automatic Installation in virtual environment

Caution: Please note that this install method is still under tests. Ajenti starts successfully on the previously mentioned supported operating systems, but all functionalities were not tested. Be kind to report any problem with this install method as issue here : <https://github.com/ajenti/ajenti/issues>

```
curl https://raw.githubusercontent.com/ajenti/ajenti/master/scripts/install-venv.sh |  
sudo bash -s -
```

Manual Installation

Native dependencies: Debian/Ubuntu

Enable Universe repository (Ubuntu only):

```
sudo add-apt-repository universe
```

```
sudo apt-get install build-essential python3-pip python3-dev python3-lxml libssl-dev  
python3-dbus python3-augeas python3-apt ntpdate
```

Native dependencies: RHEL

Enable EPEL repository:

```
sudo dnf install epel-release
```

```
sudo dnf install -y gcc python3-devel python3-pip python3-pillow python3-augeas  
python3-dbus chrony openssl-devel redhat-lsb-core
```

Install Ajenti 2

Upgrade PIP:

```
sudo pip3 install setuptools pip wheel -U
```

Minimal install:

```
sudo pip3 install ajenti-panel ajenti.plugin.core ajenti.plugin.dashboard ajenti.  
plugin.settings ajenti.plugin.plugins
```

With all plugins:

```
sudo pip3 install ajenti-panel ajenti.plugin.ace ajenti.plugin.augeas ajenti.plugin.  
auth-users ajenti.plugin.core ajenti.plugin.dashboard ajenti.plugin.datetime ajenti.  
plugin.filemanager ajenti.plugin.filesystem ajenti.plugin.network ajenti.plugin.  
notepad ajenti.plugin.packages ajenti.plugin.passwd ajenti.plugin.plugins ajenti.  
plugin.power ajenti.plugin.services ajenti.plugin.settings ajenti.plugin.terminal
```

Uninstall Ajenti 2

Ajenti is a collection of Python modules installed with pip, delivered with an init script (systemd or sysvinit). So it's necessary to remove the init script, then the Python librairies, and the configurations files.

Systemd

```
sudo systemctl stop ajenti.service
sudo systemctl disable ajenti.service
sudo systemctl daemon-reload
sudo rm -f /lib/systemd/system/ajenti.service
```

SysVinit

```
/etc/init.d/ajenti stop
rm -f /etc/init/ajenti.conf
```

Python3 modules

List all modules from Ajenti:

```
sudo pip3 list | grep aj
```

The result should be something like (eventually more or less plugins):

aj	2.1.43
ajenti-panel	2.1.43
ajenti.plugin.ace	0.30
ajenti.plugin.auth-users	0.31
ajenti.plugin.core	0.99
ajenti.plugin.dashboard	0.39
ajenti.plugin.filesystem	0.47
ajenti.plugin.passwd	0.24
ajenti.plugin.plugins	0.47
ajenti.plugin.session-list	0.4
ajenti.plugin.settings	0.30

Then simply remove all these modules:

```
sudo pip3 uninstall -y aj ajenti-panel ajenti.plugin.ace ajenti.plugin.auth-users
  ↳ ajenti.plugin.core ajenti.plugin.dashboard ajenti.plugin.filesystem ajenti.plugin.
  ↳ passwd ajenti.plugin.plugins ajenti.plugin.session-list ajenti.plugin.settings
```

Configuration files

If you don't need it for later, just delete the directory `/etc/ajenti/`:

```
sudo rm -rf /etc/ajenti/
```

1.6.2 Running Ajenti

Starting service

The automatic install script provides binary *ajenti-panel* and initscript/job/unit *ajenti*. You can ensure the service is running:

```
service ajenti restart
```

or:

```
/etc/init.d/ajenti restart
```

or:

```
systemctl restart ajenti
```

The panel will be available on **HTTPS** port **8000** by default. The default username is **root**, and the password is your system's root password.

Ajenti can also be run in a verbose debug mode:

```
ajenti-panel -v
```

Commandline options

- **-c, --config <file>** - Use given config file instead of default
- **-v** - Debug/verbose logging
- **--log <level>** - Fix log level : debug, info, warning or error
- **--dev** - Enables automatic resources build on each request
- **-d, --daemon** - Run in background (daemon mode)
- **--stock-plugins** - Run with provided plugins (default if option **--plugins** is not used)
- **--plugins <dir>** - Run with additional plugins
- **--autologin** - Will automatically log in the user under which the panel runs. **This is a security issue if your system is public.**

Debugging

If Ajenti does not start as intended, there are various ways to debug this, but it is good to know that the problem can have an origin in Python code or in Javascript code.

Debug Python problems

First of all, have a look at:

```
/var/log/ajenti/ajenti.log
```

It may contain some running errors which could be useful to understand the problem.

The traceback of a total crash would be stored in:

```
/var/log/ajenti/crash-DATE.log
```

If this log files do not provide enough informations, you can manually start Ajenti in debug mode as root:

```
systemctl stop ajenti  
/usr/local/bin/ajenti-panel -v
```

This will increase the verbosity of Ajenti in `/var/log/ajenti/ajenti.log`, but you can also directly follow the progress of Ajenti start with:

```
systemctl stop ajenti  
/usr/local/bin/ajenti-panel --dev
```

and then stop it as usual with Ctrl + C. Don't forget after this to restart the Ajenti process if necessary:

```
systemctl start ajenti
```

Debug Javascript problems

The best way to do it is to launch the developer tools in your browser, usually with F12, and to look if some errors are shown.

Submit the errors

The best way to help the development of Ajenti is then to submit the errors at <https://github.com/ajenti/ajenti/issues/new> with all informations (traceback, OS, Python version, ...).

1.6.3 Configuration files

All the configuration files are store in `/etc/ajenti` :

- **config.yml**: the main configuration file with all important parameters,
- **smtp.yml**: credentials to an email server relay, if you want to use some mail notifications or reset password functionality,
- **users.yml**: the default file which contains user account for the user authentication provider.

All configuration files use the `yaml` format

config.yml in details

Ajenti will use the following parameters :

auth block

```
auth:  
  allow_sudo: true  
  emails: {}  
  provider: os  
  users_file: /etc/ajenti/users.yml
```

Explanations:

- **allow_sudo**: **true** or **false** (allow users in the sudo group to elevate)
- **emails**: {} (not currently used)
- **provider**: authentication method to use, **os** (users from the os) or **users**
- **users_file**: if the users authentication provider is used, path to the users file (default `/etc/ajenti/users.yml`)

The parameter **user_config** was used to specified where the user configuration was stored, but is now deprecated, since it's bound to the **provider** (**os** or **users**) to avoid duplicates entries.

bind block

```
bind:
  host: 0.0.0.0
  mode: tcp
  port: 8000
```

Explanations:

- **host**: ip on which to listen (default **0.0.0.0**)
- **mode**: type of socket, **tcp** or **unix**
- **port**: port on which to listen, default **8000**

ssl block

```
ssl:
  enable: true
  certificate: /etc/ajenti/mycert.pem
  fqdn_certificate: /etc/letsencrypt/ajenti.pem
  force: false
  client_auth:
    enable: true
    force: true
    certificates:
      digest: ↴15:E8:5E:E5:D2:E8:75:0D:53:FF:22:A8:79:28:E5:BE:33:E0:37:07:FB:31:47:4D:61:69:AB:43:F8:5B:23:78
      name: C=NA, ST=NA, O=sajenti.mydomain.com, CN=root@ajenti.mydomain.com
      serial: 352674123960898230347891590646542168839110009016
      user: root
```

Explanations:

- **enable**: **true** or **false** to provide support for https. It's highly recommended to set it to **true**
- **certificate**: full path to default global certificate, used to generate client certificates, and fot the https protocol, if the parameter **fqdn_certificate** is not set. The PEM file should contains the certificate itself, and the private key.
- **fqdn_certificate**: full path certificate for your FQDN (e.g. `/etc/ajenti/mycert.pem`). The PEM file should contains the certificate itself, and the private key.
- **force**: spawn a small listener on port 80 to enable a redirect from `http://hostname` to `https://hostname:port`.

- **client_auth:**

- **enable:** **true** or **false** to enable client authentication via certificates
 - **force:** if **true**, only allows login with client certificate. If **false**, also permit authentication with password
 - **certificates:** this entry contains all client certificates for an automatic login. It will be filled through the settings in
- * **digest:** digest of the certificate
 - * **name:** name of the certificate
 - * **serial:** serial of the certificate
 - * **user:** username

email block

```
email:  
  enable: true  
  templates:  
    reset_email : /etc/ajenti/email/mytemplate_for_reset_password.html
```

Explanations:

- **enable:** **true** or **false**, if you want to enable the password reset function. But for this you need to set the smtp credentials in `/etc/ajenti/smtp.yml`
- **templates:** * **reset_email:** full path to template email for reset password functionality

The default template used to reset email password is located [here](#). The variables are automatically filled with jinja2.

Other global parameters

```
color: blue  
language: en  
logo: /srv/dev/ajenti/ajenti-panel/aj/static/images/Logo.png  
max_sessions: 10  
name: ajenti.mydomain.com  
restricted_user: nobody  
session_max_time: 1200
```

Explanations:

- **color:** secundary color of the CSS theme (possibles values are **default**, **bluegrey**, **red**, **deeporange**, **orange**, **green**, **teal**, **blue** and **purple**)
- **language:** language prefence for all users, default **en**
- **logo:** full path to your own logo, default is [the one from Ajenti](#)
- **max_sessions:** max number of simultaneously sessions, default is **99**. If the max is reached, the older inactive session will be deactivated
- **name:** your domain name

- **restricted_user**: user to use for the restricted functionalities, like for the login page. It's an important security parameter in order to limit the actions in restricted environments : all actions in restricted environments will be done with this user's privileges. Default is **nobody**.
- **session_max_time**: max validity time in seconds before automatic logout. Default is **3600** (one hour).
- **trusted_domains** (*Ajenti >= 2.2.1*) : comma separated list of trusted domains under which it's possible to reach your *Ajenti* server. When the HTTP headers are tested, a valid origin will be considered as one of the domains listed. It's necessary to specify the protocol and the port. It's mean that an entry should look like *http://my.domain.com:8000*. If set, the first entry of this list will be used as url for the password reset functionality, for more security.
- **trusted_proxies** (*Ajenti >= 2.2.1*) : comma separated list of trusted proxies. This is actually used in order to get the real ip of the client.

smtp.yml in details

This file contains all the credentials of an email server which can be used as email relay to send some notifications, like an email to reset a forgotten password.

```
smtp:
  password: MyVeryStrongStrongPassword
  port: starttls
  server: mail.mydomain.com
  user: mail@mydomain.com
```

Explanations:

- **port**: **starttls** (will use 587) or **ssl** (will use 465)
- **server**: server hostname, like *mail.mydomain.com*
- **user**: user to authenticate
- **password**: password of the mail user

users.yml in details

Ajenti gives the possibility to use two authentication methods : **os** or **users**. If **users** is used, all user informations are stored in **users_file**. It's automatically filled with the user plugin.

The default path for the **users_file** is */etc/ajenti/users.yml* with following structure:

```
users:
  arnaud:
    email: arnaud@mydomain.com
    fs_root: /home/arnaud
    password: 73637279707.....
    permissions:
      packages:install: false
      sidebar:view:/view/cron: false
    uid: 1002
```

Explanations:

- **password**: hash of the password
- **permissions**: list of permissions of the user
- **uid**: related os uid to run the worker on

- **fs_root**: root directory
- **email**: email to use for password reset.

1.6.4 Securing

Fail2ban

Failed login attempts are logged in `/var/log/ajenti/ajenti.log`. A basic filter for Fail2ban is available here : <https://raw.githubusercontent.com/ajenti/ajenti/master/scripts/ajenti.conf>

You can enable it by copying it in `/etc/fail2ban/filter.d/ajenti.conf` and with the following lines in `/etc/fail2ban/jail.d/ajenti` :

```
[ajenti]
enabled = true
port     = 8000
bantime = 120
maxretry = 3
findtime = 60
logpath = /var/log/ajenti/ajenti.log
filter   = ajenti
```

This is only an example : after 3 failed attempts (*maxretry*) the last 60 seconds (*findtime*), the found ip will be banned 2 minutes (*bantime*). You can naturally set other values related to your configuration.

1.6.5 Contributing to Ajenti

Translations

All translations are stored by [Crowdin](#), and any help is welcome. It's possible to translate directly all strings in the great interface of Crowdin and then we can include and compile it into the next release:

[Ajenti on Crowdin](#)

Testing

It's always good to have some users feedback, because we may oversee some problems. If you find an issue, please post it on [GitHub](#) with a complete description of the problem, and we will try to solve it and improve Ajenti.

Developping

There's two main axes to develop Ajenti :

- *Plugin development*: like e.g. a plugin to manage the fstab file,
- *Core development*: improve Ajenti on server side.

1.6.6 Plugin check_certificates

You can see with one look if your SSL certificates are still valid or not.

The screenshot shows the Ajenti interface for managing SSL certificates. On the left, there's a sidebar with sections for GENERAL (Dashboard, Plugins, Settings, Users), TOOLS (File Manager, Notepad, Session list, Terminal), and SOFTWARE (Docker, Services). The main area is titled "Ajenti Check certificates". It displays a table with columns: Domain, Port, Issuer, End, and Status. The table contains several rows of host information, each with a delete icon. A button labeled "Add host" is at the bottom left of the table area.

Domain	Port	Issuer	End	Status
[redacted]	8006		Can not handle SSL on this port !	✗
[green]	443	Let's Encrypt	2022-06-04 01:03:02	✓
[green]	443	Let's Encrypt	2022-05-08 22:45:40	✓
[green]	443	Let's Encrypt	2022-06-04 01:03:02	✓
[redacted]	8000		Host refuse the connection !	✗
[green]	443	Let's Encrypt	2022-05-25 03:49:23	✓

The list view let you see the hostname, the port, the issuer of the certificate, the end of the certificate, and the status of the connection.

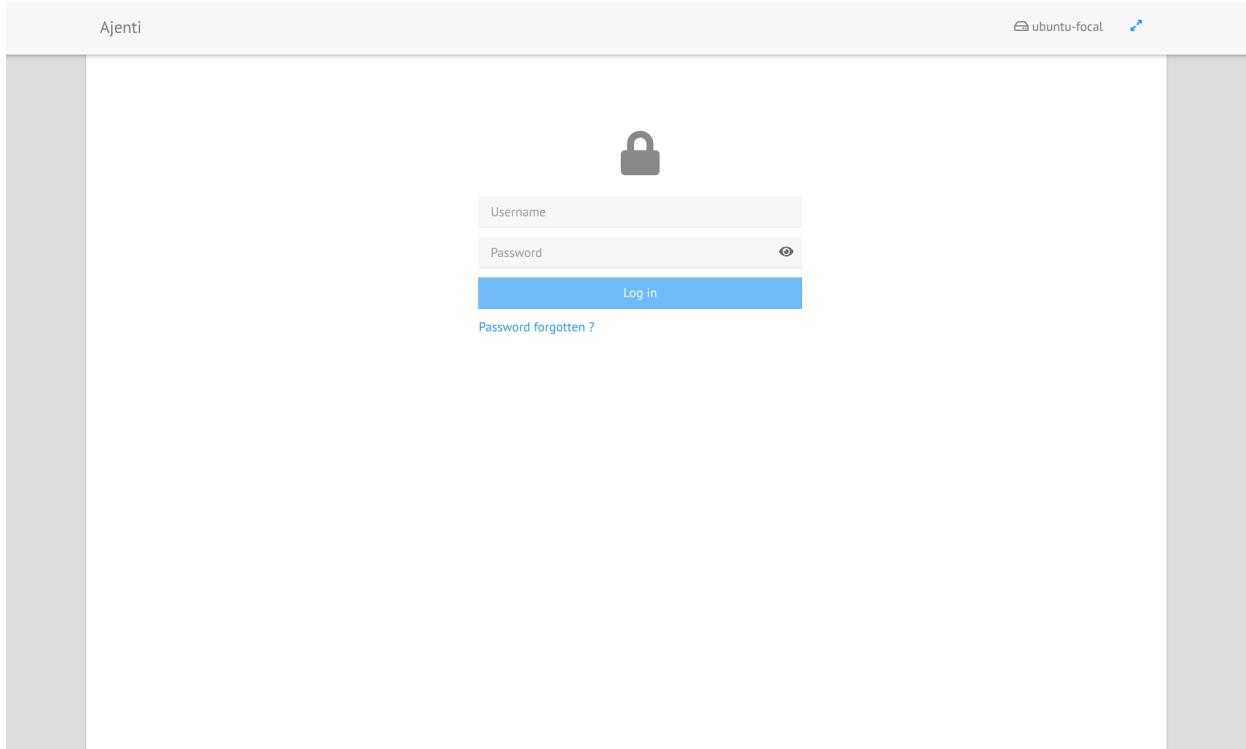
It's pretty easy to add or to remove an hostname. By default, a test will be done on port 443, the standard one for HTTPS. But you can naturally specify something else, like 8000 or 587.

If the port 587 is specified, Ajenti will try to open a STARTTLS connection, e.g. for email server.

This screenshot shows the Ajenti interface with a modal dialog box overlaid. The dialog is titled "New hostname" and has fields for "URL" (containing "mail.") and "Port (default is 443 if empty)" (containing "587"). At the bottom of the dialog are "ADD" and "CANCEL" buttons. In the background, the main interface shows a table of hosts with one row selected. The selected row for "gitlab.kientz.me" is shown in detail: Port 443, Issuer Let's Encrypt, and End date 2022-05-08 22:45:40. The status for this host is green with a checkmark.

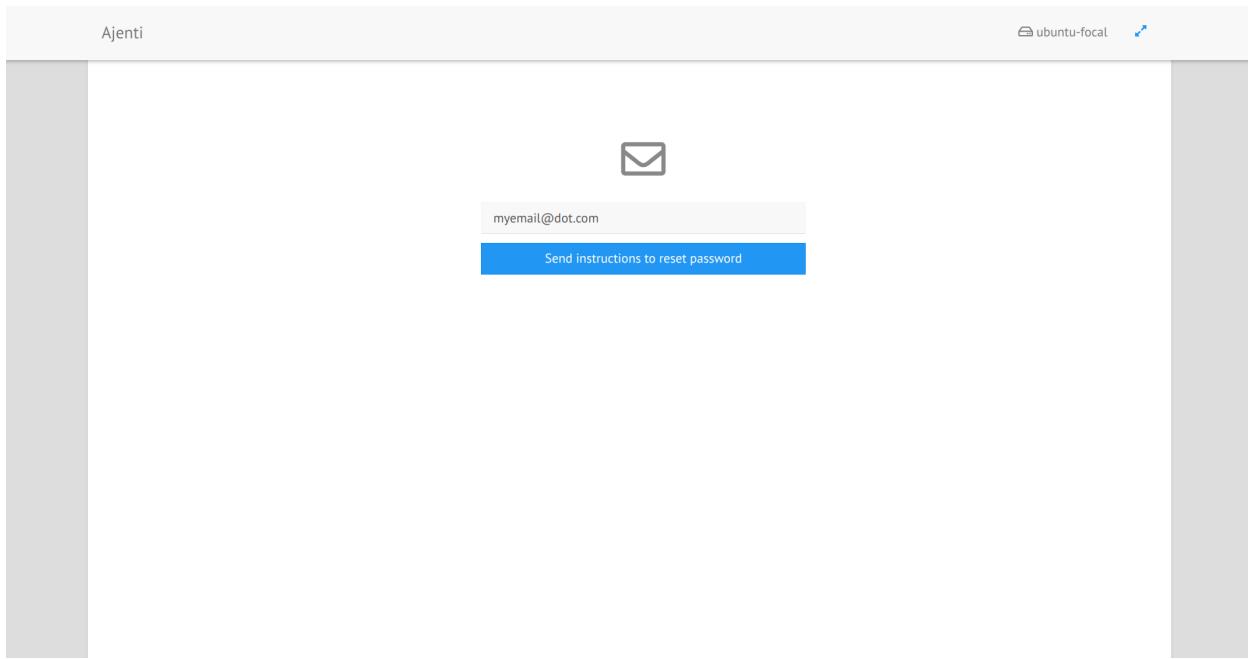
1.6.7 Plugin core

The main plugin of Ajenti is the core plugin.



This plugin manages:

- the authentication process,
- user environment setup,
- session management,
- the way the resources are delivered (CSS, JS, etc ...),
- the main template and the main style of Ajenti,
- the entries in the sidebar,
- error handling,
- password reset,
- configurations (Ajenti, user config).



It delivers a lot of tools, services, components for the other plugins too:

- hotkeys,
- tasks,
- pushes,
- dialogs,
- progress spinner,
- navbox,
- messagebox,
- smartprogerss,
- customization,
- translations with gettext,
- notifications,
- socketio.

1.6.8 Plugin cron

This plugin allows to handle all entries in a personal `cron` file.

The screenshot shows the Ajenti Cron interface. On the left, there's a sidebar with sections for GENERAL (Dashboard, Plugins, Settings, Users), TOOLS (Check certificates, File Manager, Notepad, Session list, Terminal), and SOFTWARE. The main area has tabs for Jobs, Special, and Environment variables. Under the Jobs tab, there's a table with columns: Minute, Hour, Day (month), Month, Day (week), and Command. Two cron jobs are listed:

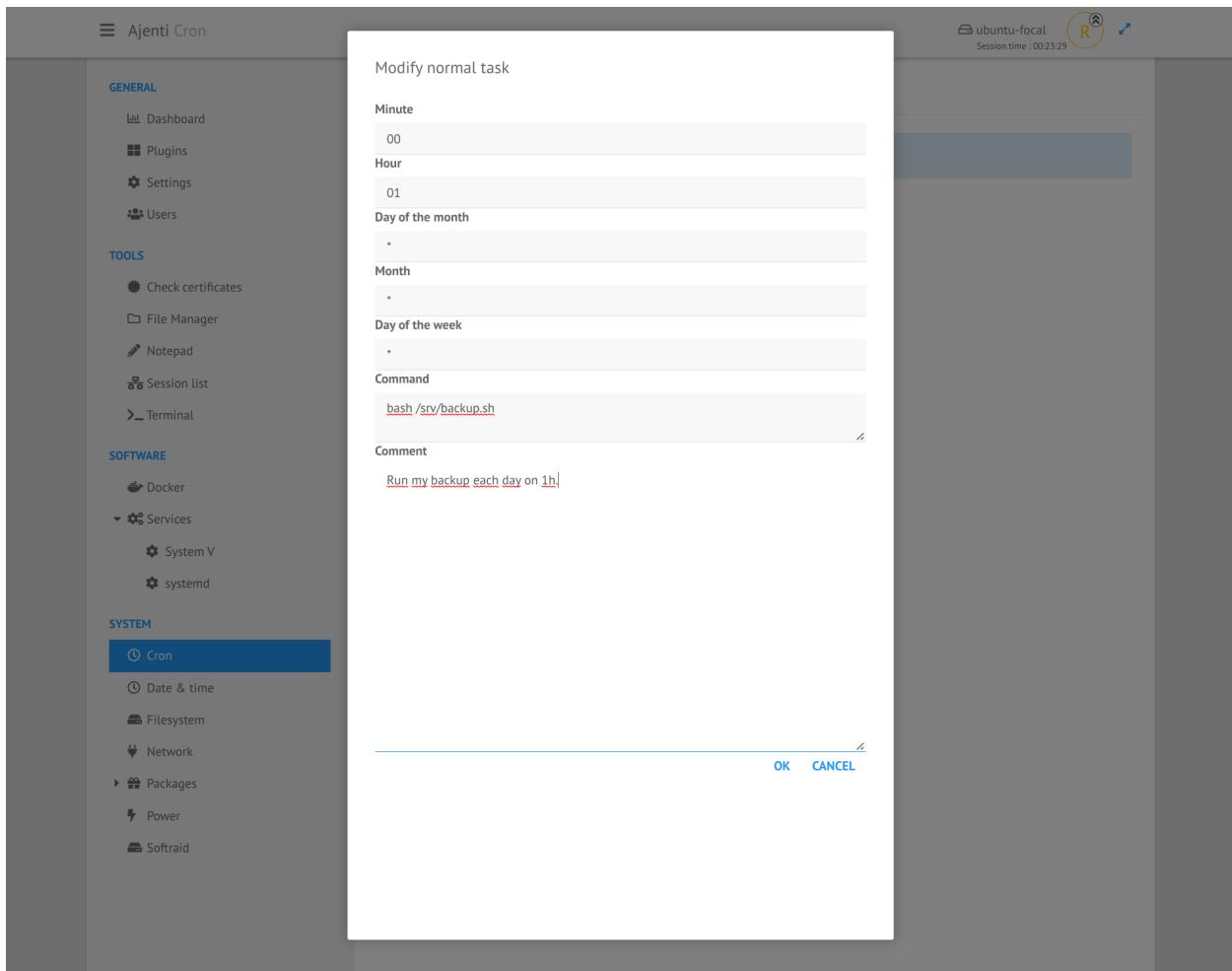
Minute	Hour	Day (month)	Month	Day (week)	Command
00	01	*	*	*	bash /srv/backup.sh
05	*	*	*	*	rsync /srv/dev /var/www

Each job row has a settings icon and a delete icon. A large 'Add' button is located at the bottom left of the table.

This is quite equivalent as running `crontab -l -u USER` to manage your own cronjobs.

With this plugin, you can:

- add jobs,
- remove jobs,
- edit jobs,
- edit special entries (@yearly, etc ...),
- set environment variables,
- add comments.



1.6.9 Plugin dashboard

This is the default landing page after successfully authenticate.

The screenshot shows the Ajenti Dashboard interface. On the left is a sidebar with a navigation menu:

- GENERAL**
 - Dashboard** (highlighted in blue)
 - Plugins
 - Settings
 - Users
- TOOLS**
 - Check certificates
 - File Manager
 - Notepad
 - Session list
 - Terminal
- SOFTWARE**
 - Services
 - System V
 - systemd
- SYSTEM**
 - Date & time
 - Network
 - Packages
 - APT
 - PIP
 - Power

The main content area displays several system monitoring widgets:

- Hostname: ubuntu-python
- Uptime: 00:40:47
- Active cores: 0/1
- CPU usage: 0% (1 min: 0, 5 min: 0, 15 min: 0.03)
- Total Memory usage: 3.9 GB (29.6 GB)
- Memory usage: 11% (Filesystem: 25%)

It's possible to display the widgets of your choice, and to order them as you want with a simple drag&drop.

You can also add other tabs, and rename them the way you want.

The list of actual available widgets:

- Check certificates,
- CPU Usage,
- Disk space (you can choose the mount point),
- Hostname,
- Load average,
- Memory usage,
- Power state,
- Script (run your own command),
- Service (status of a service in systemd or sysv init),
- Sessions (logged in users),
- Traffic,
- Uptime.

1.6.10 Plugin datetime

This plugin displays the current time zone used, and time and date set on the server.

The screenshot shows the Ajenti Date & Time plugin interface. On the left, there's a sidebar with sections for GENERAL (Dashboard, Plugins, Settings, Users), TOOLS (Check certificates, File Manager, Notepad, Session list, Terminal), and SOFTWARE (Docker, Services). The main area has tabs for 'Time zone' (set to Etc/UTC) and 'Date & time' (set to 29-March-2022, 13:30). It includes a 'Sync time from the Internet' button and a 'Set time' button. The top right shows session information for 'ubuntu-focal'.

It's possible to:

- change the time zone used,
- set the time on the server,
- synchronize time using NTP (package `ntpdate` is for this necessary).

1.6.11 Plugin docker

This plugin allows to show all running containers and images from a locally docker instance.

The screenshot shows the Ajenti Docker plugin interface. The sidebar includes sections for GENERAL (Dashboard, Plugins, Settings, Users), TOOLS (Check certificates, File Manager, Notepad, Session list, Terminal), and SOFTWARE (Docker, Services, Supervisor). The main area displays a table of running containers. The table has columns for Name (Hash), Memory usage, Cpu usage, Network used, and actions (stop/start, remove). The Docker tab is highlighted in blue. The top right shows session information for 'ubuntu-focal'.

Name (Hash)	Memory usage	Cpu usage	Network used		
mail 09039d50528f	118.6MiB / 3.852GiB	0.03%	1.63kB / 2.7kB		
nextcloud_app_1 ab5a5d165ba	41.54MiB / 3.852GiB	0.00%	1.38kB / 0B		
nextcloud_web_1 18ec48e0a723					
nextcloud_db_1 5c9f55cede02	91.85MiB / 3.852GiB	0.03%	936B / 0B		

The default tab shows all containers, with their names and id, and you can:

- start/stop a container,
- remove a container,

- see memory usage, cpu usage and network I/O

On the second tab, you will see the stored images with their sizes.

You can easily choose which one you want to delete.

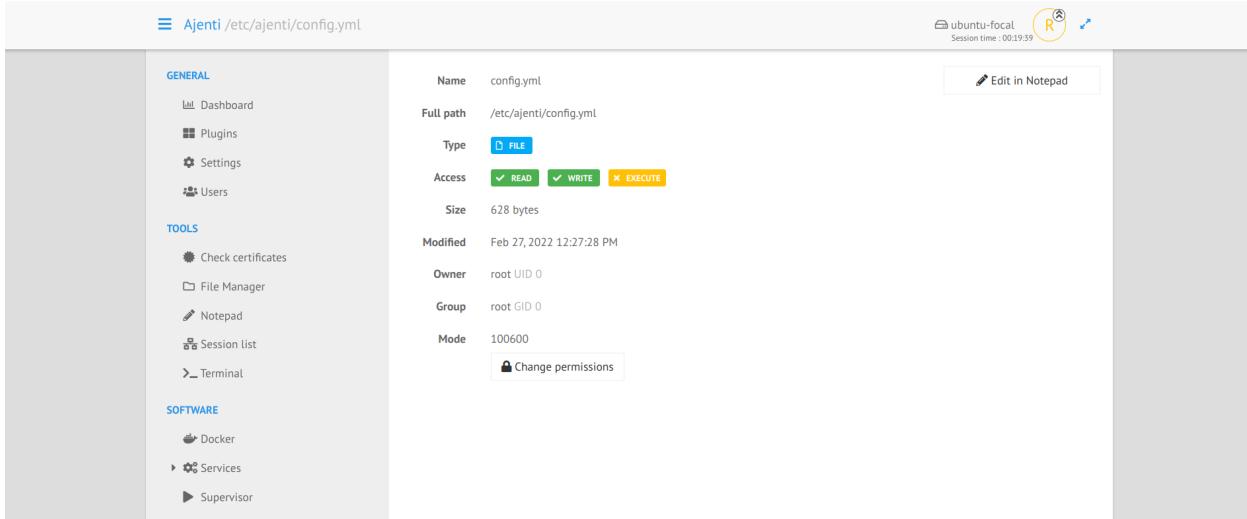
Repository	Created	Tag	Size
tviial/docker-mailserver	16 months ago	stable	628MB
tviial/docker-mailserver	22 months ago	<none>	579MB
nextcloud	2 years ago	fpm	712MB
nextcloud	2 years ago	latest	724MB
nginx	2 years ago	latest	127MB
mariadb	2 years ago	latest	356MB
tviial/docker-mailserver	2 years ago	<none>	541MB
jwilder/nginx-proxy	3 years ago	latest	148MB
apereo/cas	3 years ago	latest	642MB
jwilder/whoami	3 years ago	latest	10.1MB

1.6.12 Plugin filemanager

This plugin let you navigate on the server filesystem and perform all common operations on files and directories.

Currently, it's possible to:

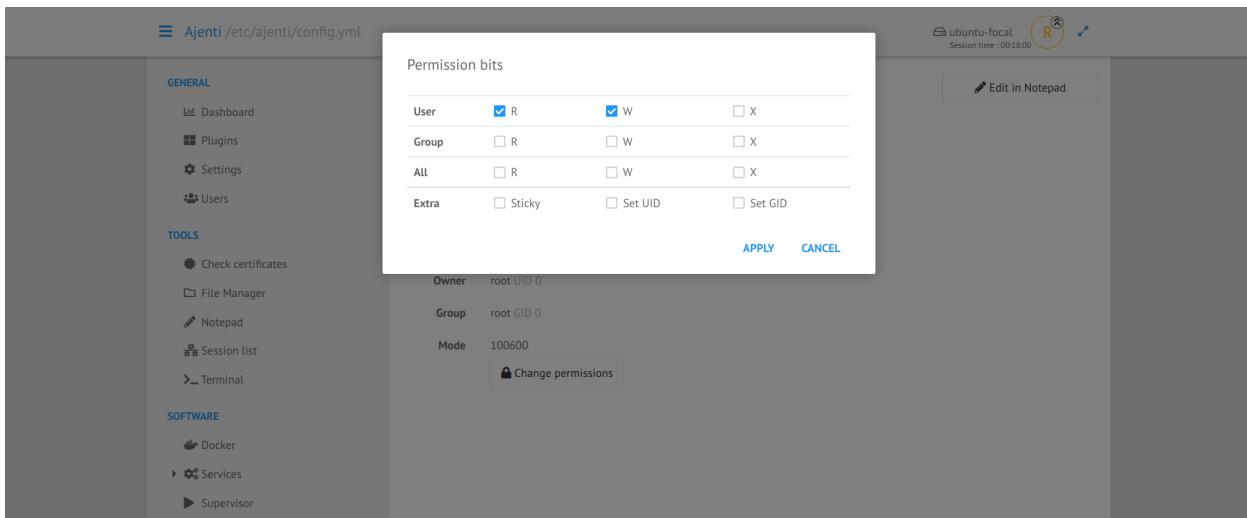
- create new files, new directories,
- upload a file through drag&drop,
- navigate in many tabs,
- cut, copy, delete files and directories (you must first select at least one object),
- display the properties of an object,
- easily navigate between directory with the breadcrumb.



In the properties view you will see all common informations (permissions, last change date, owner, etc ... the same as the command `stat`).

If the file is plain text, a button `Edit in Notepad` will appear and let you modify the file.

You can also change the permissions of the file:

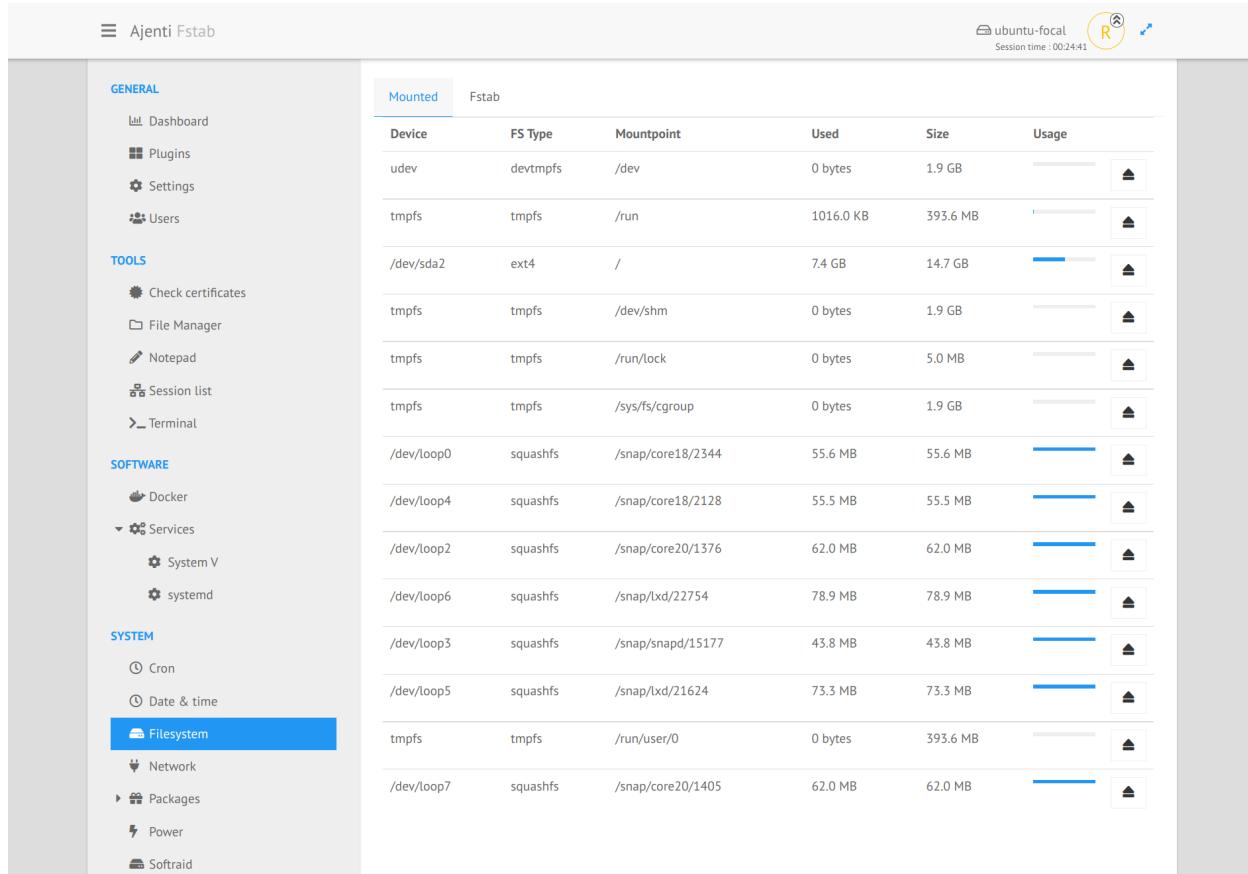


1.6.13 Plugin fstab

The first tab shows the output of the `mount` command with some util informations like:

- filesystem type,
- mountpoint,
- used space,
- total size.

The button on the right let you unmount the desired device, but you should use it with caution (don't try to unmount the root fs!).

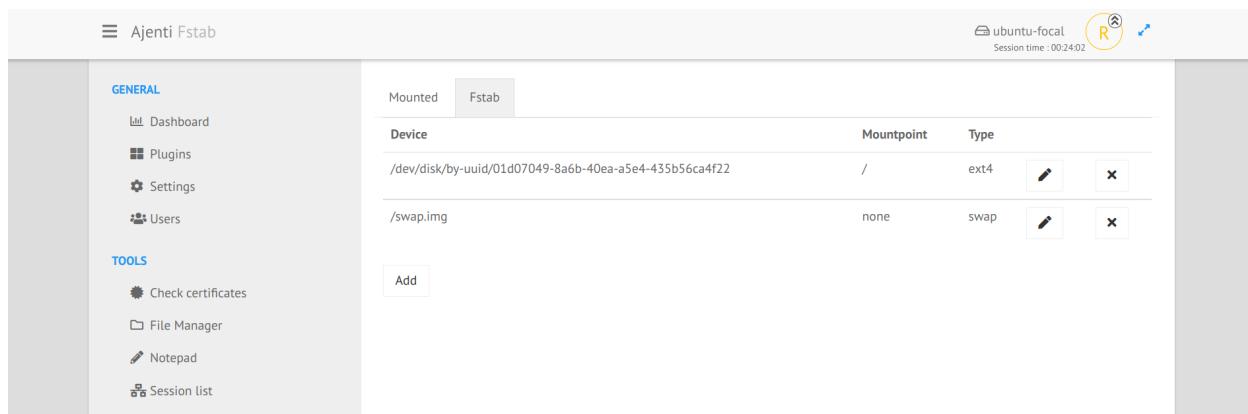


The screenshot shows the Ajenti Fstab interface. On the left is a sidebar with categories: GENERAL (Dashboard, Plugins, Settings, Users), TOOLS (Check certificates, File Manager, Notepad, Session list, Terminal), SOFTWARE (Docker, Services, System V, systemd), SYSTEM (Cron, Date & time, Filesystem, Network, Packages, Power, Softraid). The 'Filesystem' item under SYSTEM is highlighted with a blue background. The main area has two tabs: 'Mounted' (selected) and 'Fstab'. The 'Mounted' tab displays a table of file system entries:

Device	FS Type	Mountpoint	Used	Size	Usage
udev	devtmpfs	/dev	0 bytes	1.9 GB	<div style="width: 100px;"></div>
tmpfs	tmpfs	/run	1016.0 KB	393.6 MB	<div style="width: 100px;"></div>
/dev/sda2	ext4	/	7.4 GB	14.7 GB	<div style="width: 50px;"></div>
tmpfs	tmpfs	/dev/shm	0 bytes	1.9 GB	<div style="width: 100px;"></div>
tmpfs	tmpfs	/run/lock	0 bytes	5.0 MB	<div style="width: 100px;"></div>
tmpfs	tmpfs	/sys/fs/cgroup	0 bytes	1.9 GB	<div style="width: 100px;"></div>
/dev/loop0	squashfs	/snap/core18/2344	55.6 MB	55.6 MB	<div style="width: 100px;"></div>
/dev/loop4	squashfs	/snap/core18/2128	55.5 MB	55.5 MB	<div style="width: 100px;"></div>
/dev/loop2	squashfs	/snap/core20/1376	62.0 MB	62.0 MB	<div style="width: 100px;"></div>
/dev/loop6	squashfs	/snap/lxd/22754	78.9 MB	78.9 MB	<div style="width: 100px;"></div>
/dev/loop3	squashfs	/snap/snapd/15177	43.8 MB	43.8 MB	<div style="width: 100px;"></div>
/dev/loop5	squashfs	/snap/lxd/21624	73.3 MB	73.3 MB	<div style="width: 100px;"></div>
tmpfs	tmpfs	/run/user/0	0 bytes	393.6 MB	<div style="width: 100px;"></div>
/dev/loop7	squashfs	/snap/core20/1405	62.0 MB	62.0 MB	<div style="width: 100px;"></div>

The second tab lists all entries in `/etc/fstab` and let you add/modify or delete the entries.

But you should also be careful here with what you are doing.



The screenshot shows the Ajenti Fstab interface with the 'Fstab' tab selected. The sidebar is identical to the previous screenshot. The main area shows a table of entries:

Device	Mountpoint	Type
/dev/disk/by-uuid/01d07049-8a6b-40ea-a5e4-435b56ca4f22	/	ext4
/swap.img	none	swap

Below the table is a button labeled 'Add'.

1.6.14 Plugin network

This plugin contains the utilities to show the most important informations about your network interfaces.

Tab network

You will see all network interfaces, their IP and status. It's possible to bring an interface up or down and change some of their properties (not yet implemented for systems running with `netplan`). It's also possible to update the hostname name.

Tab DNS

This tab enable DNS management (add or delete DNS server).

Tab Hosts

Lists all entries in the file `/etc/hosts`, and modify or delete any single of them.

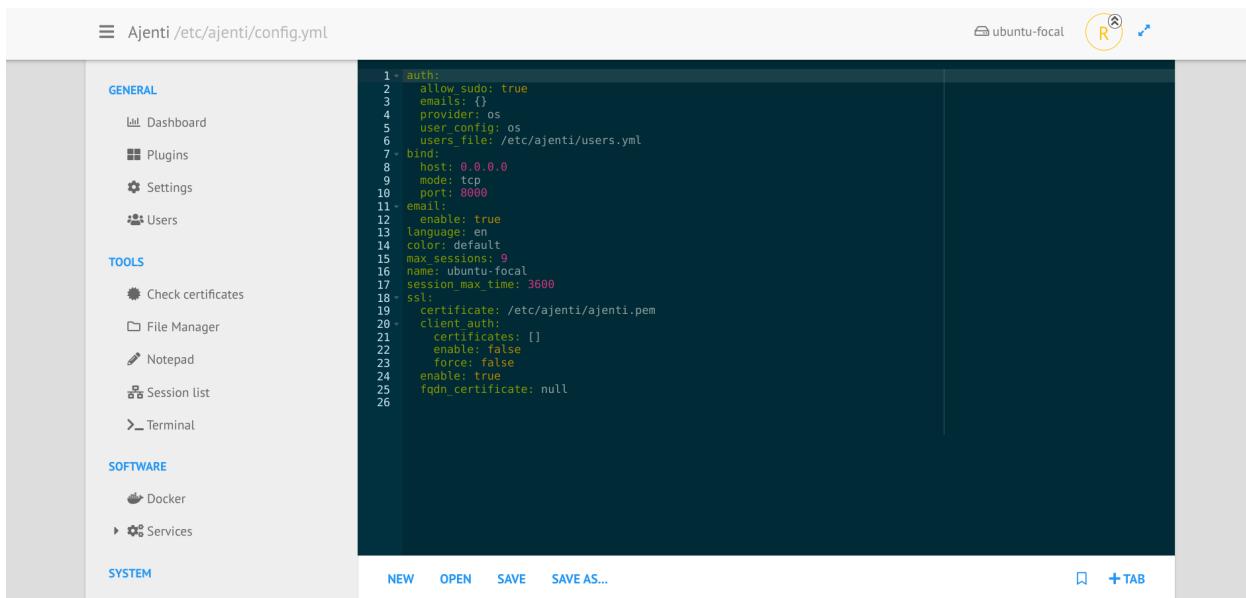
1.6.15 Plugin notepad

Based on the `ACE` editor, you can:

- edit all plain text files,
- create a new file,
- save an existing file in another location,
- manage all of these files with tabs.

Hotkey:

- `Ctrl + O` : open file
- `Ctrl + N` : new file
- `Ctrl + S` : save file



The screenshot shows the Ajenti web interface. On the left is a sidebar with navigation links: GENERAL (Dashboard, Plugins, Settings, Users), TOOLS (Check certificates, File Manager, Notepad, Session list, Terminal), SOFTWARE (Docker, Services), and SYSTEM. At the bottom of the sidebar are buttons for NEW, OPEN, SAVE, and SAVE AS... On the right is a code editor window displaying the contents of /etc/ajenti/config.yml. The code is as follows:

```
1 - auth:
2   allow sudo: true
3   emails: {}
4   provider: os
5   user_config: os
6   users_file: /etc/ajenti/users.yml
7 - bind:
8   host: 0.0.0.0
9   mode: tcp
10  port: 8000
11 - email:
12  enable: true
13  language: en
14  color: default
15  max_sessions: 9
16  name: ubuntu-focal
17  session_max_time: 3600
18 - ssl:
19  certificate: /etc/ajenti/ajenti.pem
20 - client_auth:
21  certificates: []
22  enable: false
23  force: false
24  enable: true
25  fqdn_certificate: null
26
```

1.6.16 Plugin packages

In order to manage the packages installed on your server, the plugin packages provides a quick search to filter the packages matching the search query.

Actually, the supported package engines are APT and PIP.

It's necessary to enter at least 3 chars in the search to automatically get a packages list, and then perform usual operations:

- see if a package is installed,
- see the version,
- see if a newer version is available,
- install/update a package,
- remove a package.

Package	Version	Last Update	Status
ajenti	1.2.23.13	Mar 14, 2018	Installed
ajenti-dev-multitool	1.2.0	Mar 22, 2021	Installed
ajenti-panel	2.1.44	Feb 17, 2022	Installed
ajenti.plugin.ace	0.31	Feb 17, 2022	Installed
ajenti.plugin.augeas	0.19	Feb 17, 2022	Installed
ajenti.plugin.auth-users	0.32	Feb 17, 2022	Installed
ajenti.plugin.check-certificates	0.8	Feb 17, 2022	Installed
ajenti.plugin.core	0.100	Feb 17, 2022	Installed
ajenti.plugin.cpu-temp	0.0.4	Mar 29, 2020	Installed
ajenti.plugin.cpu-temp-widget	0.1	Oct 25, 2018	Installed
ajenti.plugin.cron	0.4	Feb 17, 2022	Installed
ajenti.plugin.dashboard	0.40	Feb 17, 2022	Installed

1.6.17 Plugin plugins

Ajenti is pretty flexible and allow anyone to write its own plugin (backend Python and frontend AngularJS).

In order to manage all plugins and their versions, the plugin `plugins` lists all available plugins, shows if they are installed, or if an update if published.

The main plugin `core` can not be uninstalled, because Ajenti can not run without it, but you can check whenever a new version is available.

Updating or removing a plugin is this way pretty easy.

The screenshot shows the Ajenti web interface. On the left is a sidebar with a navigation menu:

- GENERAL
 - Dashboard
 - Plugins**
 - Settings
 - Users
- TOOLS
 - Check certificates
 - File Manager
 - Notepad
 - Session list
 - Terminal
- SOFTWARE
 - Services
 - System V
 - systemd
- SYSTEM
 - Date & time
 - Network
 - Packages
 - APT
 - PIP
 - Power

The main content area is titled "Core" and displays a message: "Ajenti core 2.1.34, no upgrades available." Below this is a section titled "Installed plugins" which lists the following:

Plugin Name	Description	Version	Action
Ace editor	ace	0.26	Uninstall
Augeas API	augeas	0.16	Uninstall
Check certificates	check_certificates	0.3	Uninstall
Core	core	0.94	Uninstall
Custom users authentication	auth_users	0.29	Uninstall
Dashboard	dashboard	0.37	Uninstall
Date & time	datetime	0.37	Uninstall
File Manager	filemanager	0.25	Uninstall
Filesystem API	filesystem	0.44	Uninstall
Network	network	0.22	Uninstall
Notepad	notepad	0.25	Uninstall
Packages	packages	0.31	Uninstall
Plugins	plugins	0.45	Uninstall
Power management	power	0.20	Uninstall
Services	services	0.28	Uninstall
Session list	session_list	0.1	Uninstall

1.6.18 Plugin power

Basically handle all around power management on your server.

Uptime appears, and you can also reboot or shutdown the server if needed.

Ajenti Power management

GENERAL

- Dashboard
- Plugins
- Settings
- Users

TOOLS

- Check certificates
- File Manager
- Notepad
- Session list
- Terminal

SOFTWARE

- Docker
- Services
 - System V
 - systemd

SYSTEM

- Cron
- Date & time
- Filesystem
- Network
- Packages
 - APT
 - PIP
- Power
- Softraid

System uptime: 00:36:57

Operations

- Power off
- Reboot
- Suspend
- Hibernate

Batteries

No batteries detected

Adapters

No adapters detected

1.6.19 Plugin services

The plugin services shows the status of services in `systemd` or in `system V init`.

The screenshot shows the Ajenti Services interface. On the left is a sidebar with categories: GENERAL (Dashboard, Plugins, Settings, Users), TOOLS (Check certificates, File Manager, Notepad, Session list, Terminal), and SOFTWARE (Docker, Services, Supervisor, System V, **systemd**, Supervisor). The main area is titled "Filter services" and lists various services with icons, names, and status (green play, yellow pause, red error). Services listed include accounts-daemon, acpid, ajenti, apparmor, apport, apport-autoreport, apt-daily, apt-daily-upgrade, atd, avahi-daemon, bittorrent, blk-availability, bootlog, bootlogs, bootmisc, and checks. Most services are marked as static.

For the `systemd` unit services, you can:

- start/stop/restart the service,
- enable/disable the service, if not static.

For the `System V` init services, you can:

- start/stop/restart the service,
- kill a running service.

1.6.20 Plugin session_list

this plugin displays the logged users, their ip and the timeout.

The screenshot shows the Ajenti List all sessions interface. The sidebar includes GENERAL (Dashboard, Plugins, Settings, Users), TOOLS (Check certificates, File Manager, Notepad, **Session list**, Terminal), and SOFTWARE (Docker, Services). The main area displays a table with columns: User (2), IP, Logged in, and Timeout. Two sessions are listed: one for root (IP redacted, Logged in 29 March 2022 16:15:18, Timeout 29 March 2022 16:35:18) and another for a user (IP redacted, Logged in 29 March 2022 16:15:01, Timeout 29 March 2022 16:35:01).

User (2)	IP	Logged in	Timeout
root	[redacted]	29 March 2022 16:15:18	29 March 2022 16:35:18
[user]	[redacted]	29 March 2022 16:15:01	29 March 2022 16:35:01

1.6.21 Plugin settings

This page gives access to the settings stored in `/etc/ajenti/config.yml` and `/etc/ajenti/smtp.yml`.

For a full description of the configuration files, please see [Configuration files](#).

After changing the settings, it's necessary to restart the panel.

Tab General

The screenshot shows the Ajenti Settings interface with the "General" tab selected. On the left, there is a sidebar with categories: GENERAL (Dashboard, Plugins, **Settings**, Users), TOOLS (Check certificates, File Manager, Notepad, Session list, Terminal), SOFTWARE (Docker, Services), and SYSTEM (Cron). The main area has tabs for General, Security, and Smtp relay. Under General, there are fields for Machine name (ubuntu-focal) and Color tag (a color palette), Language (en), and Binding (TCP, UNIX, 0.0.0.0, 8000). The top right corner shows the host name (ubuntu-focal), a yellow circular icon with an 'R' and a gear, and a blue circular icon with a lightning bolt.

This tab contains the binding settings, language, hostname set in Ajenti et color style.

Tab Security

The screenshot shows the Ajenti Settings interface with the 'Security' tab selected. The left sidebar contains categories like General, Tools, Software, and System, with 'Settings' currently highlighted. The main panel displays security configuration options:

- Authentication provider:** Set to "OS users".
- Sudo:** A checkbox labeled "Allow sudo elevation for logged in users" is checked.
- Max session time:** Set to 3600 seconds.
- SSL:** A checkbox labeled "Enable SSL" is checked.
- SSL certificate file:** Set to "/etc/ajenti/ajenti.pem".
- SSL FQDN certificate file:** An empty field.
- SSL client authentication:** Two checkboxes are present: "Enable client authentication" (unchecked) and "Deny other means of authentication" (unchecked).

You can choose:

- the authentication provider (OS or USERS),
- allow sudo elevation or not,
- set the timeout of a session,
- configure SSL and certificates,
- configure SSL and certificates for client authentication.

Tab Smtpt relay

The screenshot shows the Ajenti Settings interface with the 'Smtp relay' tab selected. The left sidebar has a blue header bar with the title 'Ajenti Settings'. Below it are several sections: GENERAL (Dashboard, Plugins, **Settings**, Users); TOOLS (Check certificates, File Manager, Notepad, Session list, Terminal); SOFTWARE (Docker, Services); and SYSTEM (Cron, Date & time, Filesystem). The main panel has tabs for General, Security, and Smtp relay. Under the Smtp relay tab, there is a section for 'Email relay' with a checked checkbox for 'Enable email relay for notification'. Below that are fields for 'SMTP host' (disabled), 'SMTP user' (disabled), 'Port' (set to 'TLS (port 587)'), 'SMTP password' (disabled), and a 'Template email reset' field.

This tab provides the credentials saved in `/etc/ajenti/smtp.yml`.

1.6.22 Plugin terminal

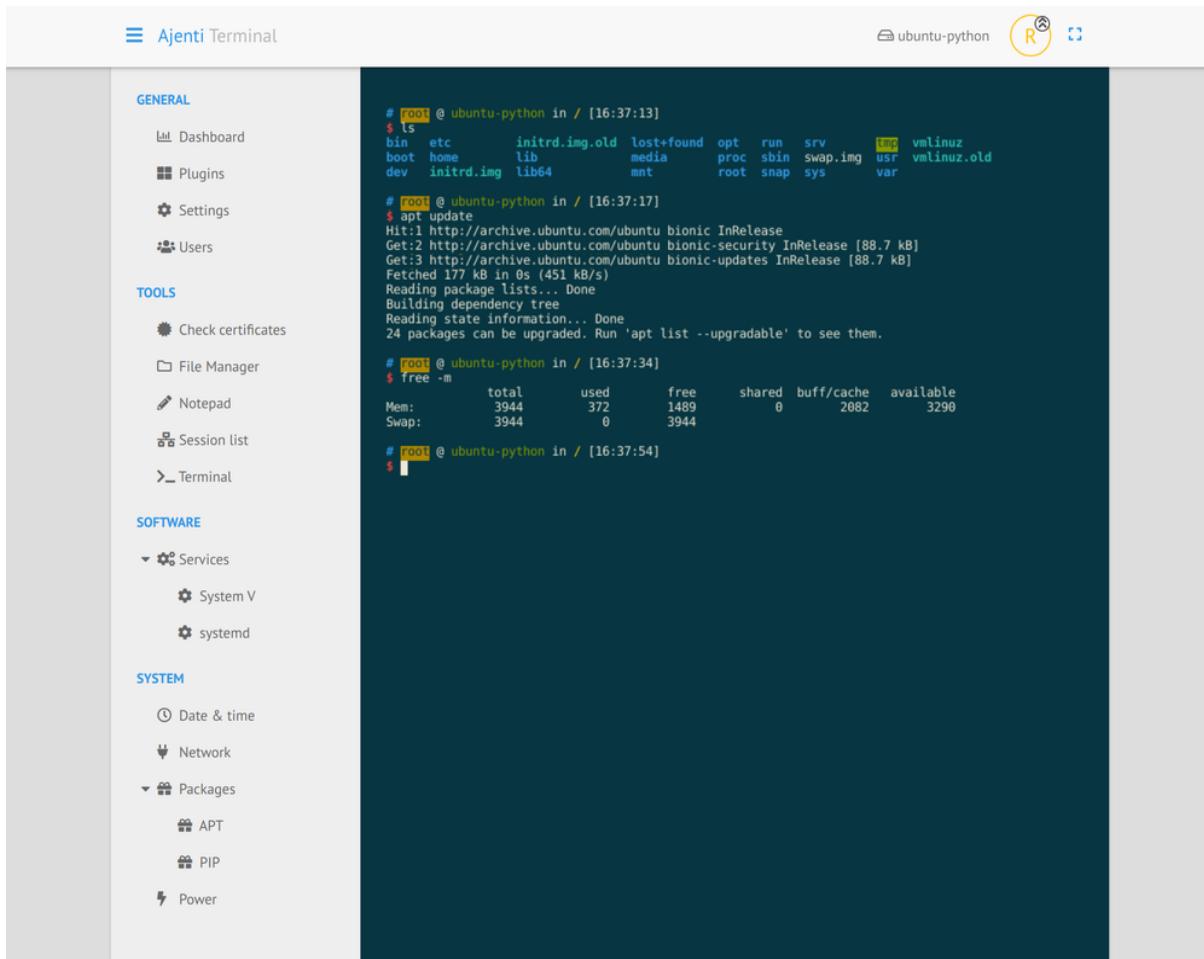
It would be really cool to have an terminal access on the server. That's exactly what this plugin does!

You have the possibility to launch a command (and naturally see the result) or to open a whole terminal on the server. You will get the same environment as your user on the system.

Type exit or Ctrl + D to come back to the terminal list.

Hotkeys

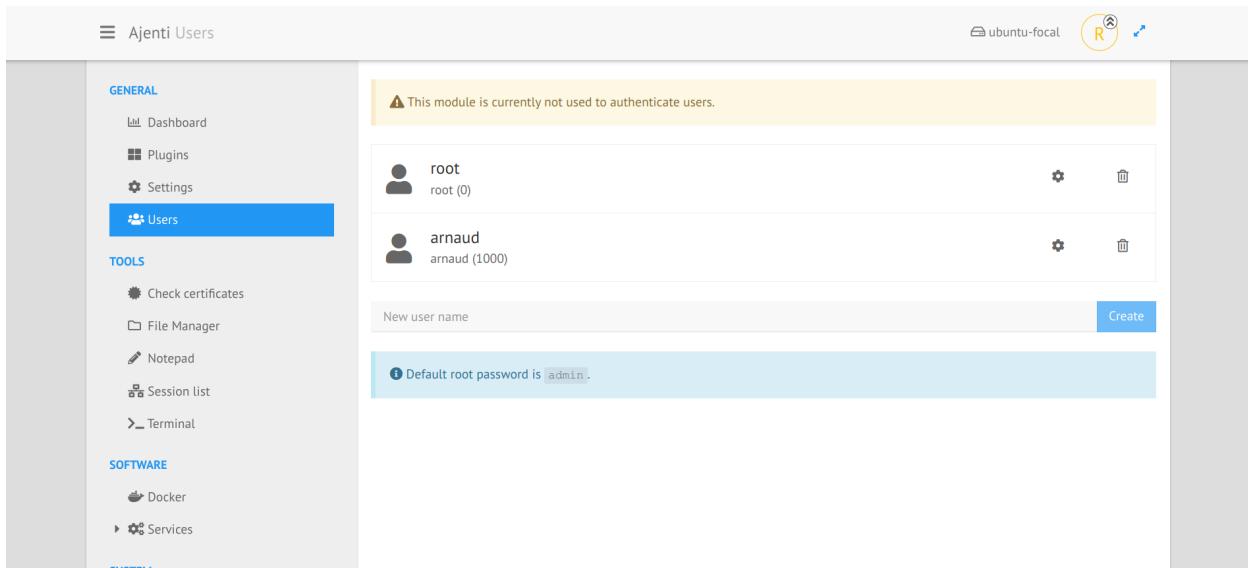
- Ctrl + C : copy
- Ctrl + V : paste
- Ctrl + D : exit



1.6.23 Plugin users

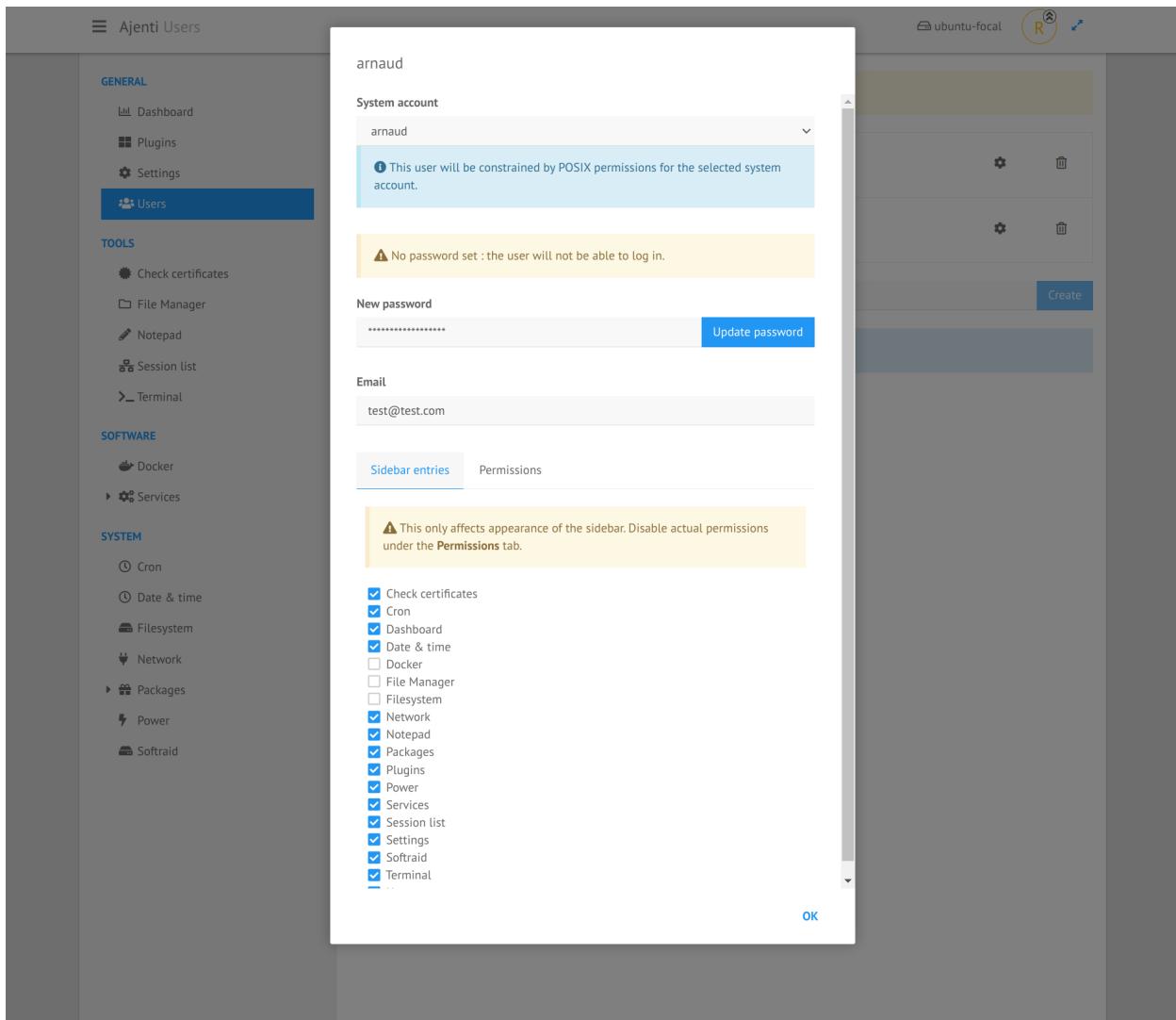
The default authentication provider used in Ajenti is the OS provider which allows all users of the system to log in.

The plugin `auth_users` provides an alternative way to authenticate users, and to create custom users. All users data are stored in plain text, in `/etc/ajenti/users.yml` (but this is configurable).



The default view presents a list of current users and let you:

- add a new user,
- manage the properties of an existing user,
- delete an existing user.



The property modal window displays some utilities per account:

- system account: all user accounts must be bound to a system account in order to set the privileges. An user bound to `root` wil have all privileges, but an user bound to a system user account like `arnaud` will only have the privileges of the system user `arnaud`.
- password change: only a hash is stored, not the password itself,
- set the email: for notifications or password reset function,
- select the sidebar entries and permissions of the user.

Don't forget to **SAVE** the changes when updating an user.

1.6.24 What's Ajenti and how it works

Ajenti Project itself consists of **Ajenti Core** itself and a set of stock plugins forming the **Ajenti Panel**.

Ajenti Core

Ajenti Core is a web interface development framework which includes a web server, IoC container, a simplistic web framework and set of core components aiding in client-server communications.

Ajenti Panel

Ajenti Panel consists of plugins developed for the Ajenti Core and a startup script, together providing a server administration panel experience. The Panel's plugins include: file manager, terminal, notepad, etc.

Modus operandi

During bootstrap, Ajenti Core will locate and load Python modules containing Ajenti plugins (identified by a `plugin.yml` file). It will then register the implementation classes found in them in the root IoC container. Some interfaces to be implemented include `aj.api.http.HttpPlugin`, `aj.plugins.core.api.sidebar SidebarItemProvider`.

Ajenti Core runs a HTTP server on a specified port, managing a pool of isolated session workers and forwarding requests to these workers, delivering them to the relevant `aj.api.http.HttpPlugin` instances. It also supports Socket.IO connections, forwarding them to the relevant `aj.api.http.SocketEndpoint` instances.

Ajenti contains a mechanism for session authentication through PAM login and sudo elevation. Standard `core` plugin provides HTTP API for that.

Authenticated sessions are moved to isolated worker processes running under the corresponding account.

Ajenti frontend is an AngularJS application composed from Angular modules provided by each plugin. Every plugin can contribute its own JS/CSS code to the combined resource package delivered to the client.

The `core` plugin provides a `ng:view` container for `ngRoute` navigation. So, the plugins that have UI are expected to provide additional `ngRoute` routes, templates and controllers.

1.6.25 Getting Started

Required knowledge

- Python 3
- JavaScript (ES5, ES6 or CoffeeScript)
- basic AngularJS knowledge (modules & controllers)
- basic HTML skills

Setting up development environment

1. Install Ajenti

We recommend to use the automatic installer - see the [installation guide](#)

2. Install build tools

Build tools require NodeJS - you can use the NodeSource repositories for quick setup:

```
# Using Ubuntu
curl -sL https://deb.nodesource.com/setup_18.x | sudo -E bash -
sudo apt-get install -y nodejs

# Using Debian, as root
curl -sL https://deb.nodesource.com/setup_18.x | bash -
apt-get install -y nodejs

# Using RHEL, as root
curl -sL https://rpm.nodesource.com/setup_18.x | bash -
```

Now, install the build tools:

```
npm -g install bower babel-cli babel-preset-es2015 babel-plugin-external-helpers less_
↳coffee-script angular-gettext-cli angular-gettext-tools

# Ubuntu or Debian:
apt-get install gettext

# RHEL
dnf install gettext
```

3. Install ajenti-dev-multitool

```
pip3 install ajenti-dev-multitool
```

Your first plugin

Create a new plugin in the current directory:

```
ajenti-dev-multitool --new-plugin "Some plugin name"
```

Build resources:

```
cd some_plugin_name
ajenti-dev-multitool --build
```

And start it:

```
sudo ajenti-dev-multitool --run-dev
```

This will start Ajenti with the stock plugins plus the current one, and will rebuild plugin resources every time you reload Ajenti in browser.

Navigate to <http://localhost:8000/>. You should see new plugin in the sidebar.

What's inside

Each plugin package consists of Python modules, which contain `jadi.component` classes (*components*). Packages also may contain *static files, templates and JS and CSS code*, e.g.:

```

* some_plugin_name
  - plugin.yml # plugin description
  - __init__.py
  - other_python_module.py
* resources
  * vendor
    - jquery-ui # Bower components
* js
  - module.js # Angular.js code
  - ecmascript6-code.es
  - coffeescript-code.coffee
* css
  - styles.css
  - styles.less
* partials
  -- index.html

```

Where to go from here

Example plugins

Download plugins from here: <https://github.com/ajenti/demo-plugins> or clone this entire repository.

Prep work:

```

ajenti-dev-multitool --bower install
ajenti-dev-multitool --rebuild

```

Run:

```

ajenti-dev-multitool --run-dev

```

Hint: Changes in ES6, CoffeeScript and LESS files will be recompiled automatically when you refresh the page; Python code will not. Additional debug information will be available in the console output and browser console. Reloading the page with Ctrl-F5 (Cache-Control: no-cache) will unconditionally rebuild all resources

1.6.26 Ajenti Dev Multitool

```

sudo pip install ajenti-dev-multitool

```

ajenti-dev-multitool is a mini-utility to help you with common plugin development tasks.

ajenti-dev-multitool typically operates on all plugins found in current directory and below.

- --run will launch the globally installed Ajenti with plugins from the current directory. --run-dev will additionally enable developer mode.
- --bower "<bower-command-with-args>" will run a Bower command for each plugin having its own bower.json file. Example: ajenti-dev-multitool --bower "install".
- --build updates the resource bundles. --rebuild will discard any previously built resources.

- `--setuppy "<setup.py-command-with-args>"` runs a setuptools command on the plugin package. A setup.py file is generated automatically. Example: `ajenti-dev-multitool --setuppy 'sdist upload --sign --identity "John Doe"`

1.6.27 User Interface

Basics

Ajenti frontend is a AngularJS-based single-page rich web application.

Your plugins can extend it by adding new Angular components (services, controllers, directives) and routes (ngRoute).

Client-server communication is facilitated by AJAX requests to backend API (\$http) and a Socket.IO connection (socket and push Angular services).

Client styling is based on a customized Twitter Bootstrap build.

Example

Basic UI example can be browsed and downloaded at https://github.com/ajenti/demo-plugins/tree/master/demo_2_ui

The basic UI plugin includes:

- an AngularJS module containing a route and a controller:
- an AngularJS view template (HTML)

1.6.28 Plugin resources

Basics

Plugin resource files are contained under resources directory nested in the plugin directory.

We encourage following structure:

```
* plugin
  * resources
    * css
      - styles.less
    * js
      - module.coffee
      - routing.coffee
    * controllers
      - some.controller.coffee
    * services
      - some.service.coffee
    * img
      - image.png
    * partials
      - view.html
```

CSS, JS and HTML resources must be listed in the `plugin.yml` file in order to be served to client. Example:

```

name: test
...
resources:
  - 'resources/vendor/jquery/dist/jquery.min.js'    # Bower component
  - 'resources/css/animations.less'                 # Styles
  - 'resources/js/core/filters.coffee'              # JS
  - 'resources/partial/index.html'                  # HTML
  - 'ng:moduleName'                                # Special syntax for publishing
  ↪an AngularJS module.

```

Please note that the last item instructs Ajenti core to load the specified AngularJS module (`test`) from the plugin.

Bower components

Example can be browsed and downloaded at https://github.com/ajenti/demo-plugins/tree/master/demo_3_bower

Plugins can depend on Bower components. To use this feature, create a `bower.json` file in your plugin directory:

```
{
  "name": "plugin",
  "private": true,
  "dependencies": {
    "jquery": "~2.1.3"
  }
}
```

Components are installed into `<plugin>/resources/vendor` directory. To install/update the components, run `ajenti-dev-multitool --bower install`. You can also run `make bower` in the root of a complete Ajenti code tree to install Bower components in all plugins.

You can run other Bower commands with e.g. `ajenti-dev-multitool --bower "list --force --verbose"`.

Resource access

AngularJS templates are pre-loaded on the client. A template residing in `plugins/test/resources/dir/template.html` can be accessed with the following URL: `/test:resources/dir/template.html`.

Other resource files are available through HTTP at `/resources/<plugin_id>/resources/<path>`.

Resource compilation

When running in dev mode (`--dev`), Ajenti will invoke `ajenti-dev-multitool --build` on page reload. Force-reloading the page (Ctrl/Cmd-F5) will rebuild all resources in all plugins using `ajenti-dev-multitool --rebuild`

`ajenti-dev-multitool` will automatically compile CoffeeScript and LESS code, concatenate CSS and JS specified in `plugin.yml` and place built CSS and JS files in `plugin/resources/build`. Please note that `ajenti-dev-multitool` will only process files in the current directory and below.

1.6.29 Handling HTTP Requests

Example

Basic HTTP API example can be browsed and downloaded at https://github.com/ajenti/demo-plugins/tree/master/demo_4_http

Plugins can provide their own HTTP endpoints by extending the `aj.api.http.HttpPlugin` abstract class.

Example:

```
import time
from jadi import component

from aj.api.http import get, HttpPlugin

from aj.api.endpoint import endpoint, EndpointError, EndpointReturn

@Component(HttpPlugin)
class Handler(HttpPlugin):
    def __init__(self, context):
        self.context = context

    @get(r'/api/demo4/calculate/(\?P<operation>\w+)/(\?P<a>\d+)/(\?P<b>\d+)')
    @endpoint(api=True)
    def handle_api_calculate(self, http_context, operation=None, a=None, b=None):
        start_time = time.time()

        try:
            if operation == 'add':
                result = int(a) + int(b)
            elif operation == 'divide':
                result = int(a) / int(b)
            else:
                raise EndpointReturn(404)
        except ZeroDivisionError:
            raise EndpointError('Division by zero')

        return {
            'value': result,
            'time': time.time() - start_time
        }
```

`@endpoint(api=True)` mode provides automatic JSON encoding of the responses and error handling.

If you need lower-level access to the HTTP response, use `@endpoint(page=True)`:

```
@get(r'/api/test')
@endpoint(page=True)
def handle_api_calculate(self, http_context):
    http_context.add_header('Content-Type', '...')
    content = "Hello!"
    #return http_context.respond_not_found()
    #return http_context.respond_forbidden()
    #return http_context.file('/some/path')
    http_context.respond_ok()
    return content
```

See `aj.http.HttpContext` for the available `http_context` methods.

1.6.30 Dashboard Widgets

Example

Basic example of a dynamic and configurable widget can be browsed and downloaded at https://github.com/ajenti/demo-plugins/tree/master/demo_5_widget

Plugins can provide dashboard widgets by extending the `aj.plugins.dashboard.api.Widget` abstract class:

```
@component(Widget)
class RandomWidget(Widget):
    id = 'random'

    # display name
    name = 'Random'

    # template of the widget
    template = '/demo_5_widget:resources/partial/widget.html'

    # template of the configuration dialog
    config_template = '/demo_5_widget:resources/partial/widget.config.html'

    def __init__(self, context):
        Widget.__init__(self, context)

    def get_value(self, config):
        # generate value based on widget's config
        if 'bytes' not in config:
            return 'Not configured'
        return os.urandom(int(config['bytes'])).encode('hex')
```

There are some CSS classes available for the standard widget looks:

```
<div ng:controller="Demo5WidgetController">
    <div class="widget-header">
        Random
    </div>
    <div class="widget-value">
        {{value || 'Unknown'}}
    </div>
</div>
```

The templates should reference appropriate controllers:

```
angular.module('ajenti.demo5').controller 'Demo5WidgetController', ($scope) ->
    # $scope.widget is our widget descriptor here
    $scope.$on 'widget-update', ($event, id, data) ->
        if id != $scope.widget.id
            return
        $scope.value = data

angular.module('ajenti.demo5').controller 'Demo5WidgetConfigController', ($scope) ->
    # $scope.configuredWidget is our widget descriptor here
    # some defaults
    $scope.configuredWidget.config.bytes ?= 4
```

Initially, dashboard will create your widget with an empty ({}) config and show the configuration dialog you provided.

Dashboard will issue periodic requests to your `aj.plugins.dashboard.api.Widget` implementations. Your widget classes should not retain any state. If user creates multiple widgets of same type, a single instance will be created to service their requests.

1.6.31 Getting Started with Core Development

Attention: This article is only useful to the developers interested in developing the Ajenti core itself. For plugin/extension development, see [Getting started with plugin development](#)

Required knowledge

- Python 3.x
- async programming with gevent
- HTML
- CoffeeScript (with AngularJS)
- LESS

Prerequisites

The following is the absolutely minimal set of software required to build and run Ajenti:

- git
- bower, babel, babel-preset-es2015 and lessc (from NPM)

Debian/Ubuntu extras:

- python3-dbus (ubuntu)

Automatic Installation

The following script will perform a complete automatic installation under Debian or Ubuntu, using virtual environment with `Python`. The virtual environment is then located in `/opt/ajenti` and the cloned git repository in `/opt/ajenti/ajenti`. This install script will install a lot of dependencies, this may take several minutes.

```
curl https://raw.githubusercontent.com/ajenti/ajenti/master/scripts/install-dev.sh |  
  sudo bash -s -
```

After a successful installation, do the following to activate the dev mode:

- Activate the virtual environment : `source /opt/ajenti/bin/activate`
- Navigate in the git repository : `cd /opt/ajenti/ajenti`
- Launch a rundev recipe : `make rundev` (quit with Ctrl+ C)
- Call `https://localhost:8000` in your browser (you will get some warnings because of the self-signed certificate, it's perfectly normal.)

Manual installation

First, it's necessary to complete the install for plugin developpement mentioned here : [Dev getting started](#)

Download the source:

```
git clone git://github.com/ajenti/ajenti.git
```

Install the dependencies:

```
# Debian/Ubuntu
sudo apt-get install build-essential python3-pip python3-dev python3-lxml libffi-dev
  ↵libssl-dev libjpeg-dev libpng-dev uuid-dev python3-dbus gettext

# RHEL
sudo dnf install gcc python3-devel python3-pip libxslt-devel libxml2-devel libffi-
  ↵-devel openssl-devel libjpeg-turbo-devel libpng-devel dbus-python gettext

cd ajenti
sudo pip3 install -r ajenti-core/requirements.txt
sudo pip3 install ajenti-dev-multitool

sudo npm install -g coffee-script less bower
```

Download and install Bower dependencies:

```
make bower
```

Ensure that resource compilation is set up correctly and works (optional):

```
make build
```

Launch Ajenti in dev mode:

```
make rundev
```

Navigate to <http://localhost:8000/>.

Hint: Changes in CoffeeScript and LESS files will be recompiled automatically when you refresh the page; Python code will not. Additional debug information will be available in the console output and browser console. Reloading the page with Ctrl-F5 (Cache-Control: no-cache) will unconditionally rebuild all resources

1.6.32 API: jadi

`jadi.get_fqdn (cls)`

Returns a fully-qualified name for the given class

`jadi.interface (cls)`

Marks the decorated class as an abstract interface.

Injects following classmethods:

`.all (context)`

Returns a list of instances of each component in the context implementing this @interface

Parameters `context (Context)` – context to look in

Returns list(cls)

```
.any (context)
    Returns the first suitable instance implementing this @interface or raises
    NoImplementationError if none is available.
    Parameters context (Context) – context to look in
    Returns cls

.classes ()
    Returns a list of classes implementing this @interface
    Returns list(class)

jadi.component (iface)
    Marks the decorated class as a component implementing the given iface
    Parameters iface (interface ()) – the interface to implement

jadi.service (cls)
    Marks the decorated class as a singleton service.

Injects following classmethods:

.get (context)
    Returns a singleton instance of the class for given context
    Parameters context (Context) – context to look in
    Returns cls

class jadi.Context (parent=None)
    An IoC container for interface () s, service () s and component () s
    Parameters parent (Context) – a parent context

    get_component (cls)
    get_components (cls, ignore_exceptions=False)
    get_service (cls)

exception jadi.NoImplementationError (cls)
```

1.6.33 API: aj

```
aj.config = <module 'aj.config' from '/home/docs/checkouts/readthedocs.org/user_builds/ajer
    Configuration dict

aj.platform = 'debian'
    Current platform

aj.platform_string = 'Ubuntu 18.04.5 LTS'
    Human-friendly platform name

aj.platform_unmapped = 'ubuntu'
    Current platform without "Ubuntu is Debian"-like mapping

aj.version = '2.2.6'
    Ajenti version

aj.server = None
    Web server

aj.debug = False
    Debug mode

aj.init()
```

```
aj.exit()
aj.restart()
```

1.6.34 API: aj.api.http

```
class aj.api.http.BaseHttpHandler
    Base class for everything that can process HTTP requests

    handle(http_context)
        Should create a HTTP response in the given http_context and return the plain output

        Parameters http_context (aj.http.HttpContext) – HTTP context

class aj.api.http.HttpMasterMiddleware(context)

    handle(http_context)
        Should create a HTTP response in the given http_context and return the plain output

        Parameters http_context (aj.http.HttpContext) – HTTP context
```

```
class aj.api.http.HttpMiddleware(context)
```

```
    handle(http_context)
        Should create a HTTP response in the given http_context and return the plain output

        Parameters http_context (aj.http.HttpContext) – HTTP context
```

```
class aj.api.http.HttpPlugin(context)
    A base interface for HTTP request handling:
```

```
@component
class HelloHttp(HttpPlugin):
    @get('/hello/(?P<name>.+)')
    def get_page(self, http_context, name=None):
        context.add_header('Content-Type', 'text/plain')
        context.respond_ok()
        return f'Hello, {name}!'
```

```
handle(http_context)
    Finds and executes the handler for given request context (handlers were methods decorated with url()
    and will be decorated with e.g. @get and @post in the future)
```

Parameters http_context (aj.http.HttpContext) – HTTP context

Returns reponse data

```
class aj.api.http.SocketEndpoint(context)
    Base interface for Socket.IO endpoints.
```

```
destroy()
    Destroys endpoint, killing the running greenlets
```

```
on_connect(message)
    Called on a successful client connection
```

```
on_disconnect(message)
    Called on a client disconnect
```

on_message (*message, *args*)

Called when a socket message arrives to this endpoint

plugin = None

arbitrary plugin ID for socket message routing

send (*data, plugin=None*)

Sends a message to the client.the

Parameters

- **data** – message object

- **plugin** (*str*) – routing ID (this endpoint's ID if not specified)

spawn (*target, *args, **kwargs*)

Spawns a greenlet in this endpoint, which will be auto-killed when the client disconnects

Parameters target – target function

`aj.api.http.copy(pattern)`

Exposes the decorated method of your *HttpPlugin* via HTTP

Parameters pattern (*str*) – URL regex (^ and \$ are implicit)

Return type

function

Named capture groups will be fed to function as **kwargs

`aj.api.http.delete(pattern)`

Exposes the decorated method of your *HttpPlugin* via HTTP

Parameters pattern (*str*) – URL regex (^ and \$ are implicit)

Return type

function

Named capture groups will be fed to function as **kwargs

`aj.api.http.get(pattern)`

Exposes the decorated method of your *HttpPlugin* via HTTP

Parameters pattern (*str*) – URL regex (^ and \$ are implicit)

Return type

function

Named capture groups will be fed to function as **kwargs

`aj.api.http.head(pattern)`

Exposes the decorated method of your *HttpPlugin* via HTTP

Parameters pattern (*str*) – URL regex (^ and \$ are implicit)

Return type

function

Named capture groups will be fed to function as **kwargs

`aj.api.http.lock(pattern)`

Exposes the decorated method of your *HttpPlugin* via HTTP

Parameters pattern (*str*) – URL regex (^ and \$ are implicit)

Return type

function

Named capture groups will be fed to function as **kwargs

`aj.api.http.mkcol(pattern)`

Exposes the decorated method of your *HttpPlugin* via HTTP

Parameters `pattern` (*str*) – URL regex (^ and \$ are implicit)

Return type

function

Named capture groups will be fed to function as **kwargs

`aj.api.http.move(pattern)`

Exposes the decorated method of your *HttpPlugin* via HTTP

Parameters `pattern` (*str*) – URL regex (^ and \$ are implicit)

Return type

function

Named capture groups will be fed to function as **kwargs

`aj.api.http.options(pattern)`

Exposes the decorated method of your *HttpPlugin* via HTTP

Parameters `pattern` (*str*) – URL regex (^ and \$ are implicit)

Return type

function

Named capture groups will be fed to function as **kwargs

`aj.api.http.patch(pattern)`

Exposes the decorated method of your *HttpPlugin* via HTTP

Parameters `pattern` (*str*) – URL regex (^ and \$ are implicit)

Return type

function

Named capture groups will be fed to function as **kwargs

`aj.api.http.post(pattern)`

Exposes the decorated method of your *HttpPlugin* via HTTP

Parameters `pattern` (*str*) – URL regex (^ and \$ are implicit)

Return type

function

Named capture groups will be fed to function as **kwargs

`aj.api.http.propfind(pattern)`

Exposes the decorated method of your *HttpPlugin* via HTTP

Parameters `pattern` (*str*) – URL regex (^ and \$ are implicit)

Return type

function

Named capture groups will be fed to function as **kwargs

`aj.api.http.proppatch(pattern)`

Exposes the decorated method of your `HttpPlugin` via HTTP

Parameters `pattern` (`str`) – URL regex (^ and \$ are implicit)

Return type

function

Named capture groups will be fed to function as **kwargs

`aj.api.http.put(pattern)`

Exposes the decorated method of your `HttpPlugin` via HTTP

Parameters `pattern` (`str`) – URL regex (^ and \$ are implicit)

Return type

function

Named capture groups will be fed to function as **kwargs

`aj.api.http.requests_decorator_generator(method)`

Factorization to generate request decorators like @get or @post.

Parameters `method` (`basestring`) – Request method decorator to generate, like get or post

Returns

Return type

`aj.api.http.unlock(pattern)`

Exposes the decorated method of your `HttpPlugin` via HTTP

Parameters `pattern` (`str`) – URL regex (^ and \$ are implicit)

Return type

function

Named capture groups will be fed to function as **kwargs

`aj.api.http.url(pattern)`

Exposes the decorated method of your `HttpPlugin` via HTTP. Will be deprecated in favor of new decorators
(@get, @post, ...)

Parameters `pattern` (`str`) – URL regex (^ and \$ are implicit)

Return type

function

Named capture groups will be fed to function as **kwargs

1.6.35 API: aj.api.endpoint

`exception aj.api.endpoint.EndpointError(inner, message=None)`

To be raised by endpoints when a foreseen error occurs. This exception doesn't cause a client-side crash dialog.

Parameters

- `inner` – inner exception
- `message` – message

```
exception aj.api.endpoint.EndpointReturn(code, data=None)
```

Raising EndpointReturn will return a custom HTTP code in the API endpoints.

Parameters

- **code** – HTTP code
- **data** – response data

```
aj.api.endpoint.endpoint(page=False, api=False, auth=True)
```

It's recommended to decorate all HTTP handling methods with @endpoint.

@endpoint (auth=True) will require authenticated session before giving control to the handler.

@endpoint (api=True) will wrap responses and exceptions into JSON, and will also provide special handling of EndpointsError

Parameters

- **auth** (bool) – requires authentication for this endpoint
- **page** (bool) – enables page mode
- **api** (bool) – enables API mode

1.6.36 API: aj.config

```
class aj.config.UserConfigService(context)

    classmethod get(context)
        get_provider()
```

1.6.37 API: aj.core

```
aj.core.restart()
```

```
aj.core.run(config=None, plugin_providers=None, product_name='ajenti', dev_mode=False, debug_mode=False, autologin=False)
```

A global entry point for Ajenti.

Parameters

- **config** (aj.config.BaseConfig) – config file implementation instance to use
- **plugin_providers** (list([aj.plugins.PluginProvider](#))) – list of plugin providers to load plugins from
- **product_name** (str) – a product name to use
- **dev_mode** (bool) – enables dev mode (automatic resource recompilation)
- **debug_mode** (bool) – enables debug mode (verbose and extra logging)
- **autologin** (bool) – disables authentication and logs everyone in as the user running the panel. This is EXTREMELY INSECURE.

1.6.38 API: aj.entry

```
aj.entry.handle_crash(exc)
aj.entry.start(daemonize=False, log_level=20, dev_mode=False, **kwargs)
```

A wrapper for `run()` that optionally runs it in a forked daemon process.

Parameters `kwargs` – rest of arguments is forwarded to `run()`

1.6.39 API: aj.http

```
class aj.http.CGIFieldStorage(fp=None,      headers=None,      outerboundary=b'',      envi-
ron=environ({'READTHEDOCS_VIRTUALENV_PATH':
'/home/docs/checkouts/readthedocs.org/user_builds/ajenti2/envs/latest',
'READTHEDOCS_CANONICAL_URL':
'http://docs.ajenti.org/en/latest/',
'RTD PYTHON VERSION_35':
'3.5.10',          'HOSTNAME':         'build-21790423-project-27581-ajenti2',
'RTD PYTHON VERSION_27':
'2.7.18',          'RTD PYTHON VERSION_36':   '3.6.12',
'RTD PYTHON VERSION_37': '3.7.9',  'APPDIR':     '/app',
'RTD PYTHON VERSION_38': '3.8.6',  'READTHE-
DOCS_GIT_CLONE_URL': 'https://github.com/ajenti/ajenti',
'HOME':             '/home/docs',    'RTD_SETUPTOOLS_VERSION':
'45.1.0',          'NO_COLOR':       '1', 'READTHEDOCS': 'True',
'RTD_PIPY_VERSION_35': 'pypy3.5-7.0.0', 'READTHE-
DOCS_PROJECT': 'ajenti2',  'READTHEDOCS_OUTPUT':
'/home/docs/checkouts/readthedocs.org/user_builds/ajenti2/checkouts/latest/_readthedocs-
PATH': '/home/docs/checkouts/readthedocs.org/user_builds/ajenti2/envs/latest/bin:/home',
'READTHEDOCS_VERSION_TYPE': 'branch',
'RTD_PIP_VERSION': '20.0.1',    'LANG':      'C.UTF-8',
'READTHEDOCS_LANGUAGE': 'en',  'DEBIAN_FRONTEND':
'noninteractive',    'READTHEDOCS_GIT_COMMIT_HASH':
'0dae5b9f3f29b1204d14beebaa47b3aad2e80b0f',
'READTHEDOCS_VERSION_NAME': 'lat-
est',              'RTD_VIRTUALENV_VERSION':
'16.7.9',          'RTD_CONDA_VERSION': '4.6.14',
'READTHEDOCS_VERSION': 'latest',    'PWD':
'/home/docs/checkouts/readthedocs.org/user_builds/ajenti2/checkouts/latest/docs/source',
'PYENV_ROOT':        '/home/docs/.pyenv',  'READTHE-
DOCS_GIT_IDENTIFIER': 'master',  'DOCUTILSCONFIG':
'/home/docs/checkouts/readthedocs.org/user_builds/ajenti2/checkouts/latest/docs/source/d
keep_blank_values=0, strict_parsing=0, limit=None,
encoding='utf-8', errors='replace', max_num_fields=None)
```

`make_file(binary=None)`

Always open a tempfile as binary

```
class aj.http.HttpContext(env, start_response=None)
```

Instance of `HttpContext` is passed to all HTTP handler methods

`env`

WSGI environment dict

`path`

Path segment of the URL

method
Request method

headers
List of HTTP response headers

body
Request body

response_ready
Indicates whether a HTTP response has already been submitted in this context

query
HTTP query parameters

add_header (key, value)
Adds a given HTTP header to the response

Parameters

- **key** (*str*) – header name
- **value** (*str*) – header value

classmethod deserialize (data)

dump_env ()

fallthrough (handler)
Executes a handler in this context

Returns handler-supplied output

file (path, stream=False, inline=False, name=None)
Returns a GZip compressed response with content of file located in path and correct headers

get_cleaned_env ()

gzip (content, compression=6)
Returns a GZip compressed response with given content and correct headers

Parameters **compression** (*int*) – compression level from 0 to 9

Return type str

json_body ()

redirect (location)
Returns a HTTP 302 Found redirect response with given location

remove_header (key)
Removed a given HTTP header from the response

Parameters **key** (*str*) – header name

respond (status)
Creates a response with given HTTP status line

respond_forbidden ()
Returns a HTTP 403 Forbidden response

```
respond_not_found()
    Returns a HTTP 404 Not Found response

respond_ok()
    Creates a HTTP 200 OK response

respond_server_error()
    Returns a HTTP 500 Server Error response

respond_unauthenticated()
    Returns a HTTP 401 Unauthenticated response

run_response()
    Finalizes the response and runs WSGI's start_response().

serialize()

class aj.http.HttpMiddlewareAggregator(stack)
    Stacks multiple HTTP handlers together in a middleware fashion.

        Parameters stack (list(aj.api.http.BaseHttpHandler)) – handler list

handle(http_context)
    Should create a HTTP response in the given http_context and return the plain output

        Parameters http_context (aj.http.HttpContext) – HTTP context

class aj.http.HttpRoot(handler)
    A root WSGI middleware object that creates the HttpContext and dispatches it to an HTTP handler.

        Parameters handler (aj.api.http.BaseHttpHandler) – next middleware handler

dispatch(env, start_response)
    Dispatches the WSGI request
```

1.6.40 API: aj.plugins

```
class aj.plugins.PluginProvider
    A base class for plugin locator

    provide()
        Should return a list of found plugin paths

        Returns list(str)

class aj.plugins.DirectoryPluginProvider(path)
    A plugin provider that looks up plugins in a given directory.

        Parameters path – directory to look for plugins in

    provide()
        Should return a list of found plugin paths

        Returns list(str)

class aj.plugins.PythonPathPluginProvider
    A plugin provider that looks up plugins on $PYTHONPATH

    provide()
        Should return a list of found plugin paths

        Returns list(str)

exception aj.plugins.PluginLoadError
```

```
exception aj.plugins.PluginCrashed(exception)

    describe()

class aj.plugins.Dependency

    exception Unsatisfied

        describe()
        reason()
        build_exception()
        check()
        value
        yaml_loader
            alias of yaml.loader.SafeLoader
        yaml_tag = '!Dependency'

class aj.plugins.ModuleDependency(module_name=None)

    exception Unsatisfied

        reason()
        description = 'Python module'
        is_satisfied()
        yaml_tag = '!ModuleDependency'

class aj.plugins.PluginDependency(plugin_name=None)

    exception Unsatisfied

        reason()
        description = 'Plugin'
        is_satisfied()
        yaml_tag = '!PluginDependency'

class aj.plugins.OptionalPluginDependency(plugin_name=None)

    exception Unsatisfied

        reason()
        description = 'Plugin'
        is_satisfied()
        yaml_tag = '!OptionalPluginDependency'
```

```
class aj.plugins.BinaryDependency (binary_name=None)

    exception Unsatisfied

        reason()

    description = 'Application binary'

    is_satisfied()

    yaml_tag = '!BinaryDependency'

class aj.plugins.FileDependency (file_name=None)

    exception Unsatisfied

        reason()

    description = 'File'

    is_satisfied()

    yaml_tag = '!FileDependency'

class aj.plugins.PluginManager (context)
    Handles plugin loading and unloading

    classmethod get (context)

        get_content_path (name, path)

        get_crash (name)

        get_loaded_plugins_list ()

        load_all_from (providers)
            Loads all plugins provided by given providers.

            Parameters providers (list(PluginProvider)) -
```

1.6.41 Angular: ajenti.core

This Angular module contains core components of Ajenti frontend.

Services

```
class config()

    config.data
        Config file content object

    config.load()
        Gets complete configuration data of the backend

        Returns promise

    config.save()
        Updates and saves configuration data
```

Returns promise

```
config.getUserConfig()
    Gets per-user configuration data of the backend
```

Returns promise → per-user Ajenti config object

```
config.setUserConfig(config)
    Updates and saves per-user configuration data
```

Arguments

- **config** (*object*) – updated configuration data from `getUserConfig()`

Returns promise

```
class core()
```

```
core.pageReload()
    Reloads the current URL
```

```
core.restart()
    Restarts the Ajenti process
```

```
class hotkeys()
    Captures shortcut key events
```

```
hotkeys.ENTER, ESC
    Respective key codes
```

```
hotkeys.on(scope, handler, mode='keydown')
    Registers a hotkey handler in the provided scope
```

Arguments

- **scope** (*\$scope*) – \$scope to install handler into
- **handler** (*function(keyCode, rawEvent)*) – handler function. If the function returns a truthy value, event is cancelled and other handlers aren't notified.
- **mode** (*string*) – one of keydown, keypress or keyup.

```
class identity()
    Provides info on the authentication status and user/machine identity
```

```
identity.user
    Name of the logged in user
```

```
identity.effective
    Effective UID of the server process
```

```
identity.machine.name
    User-provided name of the machine
```

```
identity.isSuperuser
    Whether current user is a superuser or not
```

```
identity.auth(username, password, mode)
    Attempts to authenticate current session as username:password with a mode of normal or sudo
```

```
identity.login()
    Redirects user to a login dialog
```

```
identity.logout()
    Deauthenticates current session
```

```
identity.elevate()  
    Redirects user to a sudo elevation dialog
```

```
class messagebox()  
    Provides interface to modal messagebox engine  
  
    messagebox.show(options)  
        Opens a new messagebox.
```

Arguments

- **options (object)** –
- **options.title (string)** –
- **options.text (string)** –
- **options.positive (string)** – positive action button text. Clicking it will resolve the returned promise.
- **options.negative (string)** – negative action button text. Clicking it will reject the returned promise.
- **options.template (string)** – (optional) custom body template
- **options.scrollable (boolean)** – whether message body is scrollable
- **options.progress (boolean)** – whether to display an indeterminate progress indicator in the message

Returns a Promise-like object with an additional `close()` method.

```
class notify()
```

```
    notify.info(title, text)  
    notify.success(title, text)  
    notify.warning(title, text)  
    notify.error(title, text)  
        Shows an appropriately styled notification  
    notify.custom(style, title, text, url)  
        Shows a clickable notification leading to url.
```

```
class pageTitle()  
    Alters page <title> and global heading.
```

```
pageTitle.set(text)  
    Sets title text  
pageTitle.set(expression, scope)  
    Sets an title expression to be watched. Example:
```

```
$scope.getTitle = (page) -> someService.getPageTitle(page)  
$scope.page = ...  
  
pageTitle.set("getTitle(page)", $scope)
```

```
class push()  
    Processes incoming push messages (see aj.plugins.core.api.push). This service has no public methods.
```

This service broadcasts events that can be received as:

```
$scope.$on 'push:pluginname', (message) ->
    processMessage(message) ...
```

class tasks()

An interface to the tasks engine (see [aj.plugins.core.api.tasks](#)).

tasks.tasks

A list of task descriptors for the currently running tasks. Updated automatically.

tasks.start (cls, args, kwargs)

Starts a server-side task.

Arguments

- **cls** (*string*) – full task class name (aj.plugins.pluginname....)
- **args** (*array*) – task arguments
- **kwargs** (*object*) – task keyword arguments

Returns a promise, resolved once the task actually starts

Directives

autofocus()

Automatically focuses the input. Example:

```
<input type="text" autofocus ng:model="..." />
```

checkbox()

Renders a checkbox. Example:

```
<span checkbox ng:model="..." text="Enable something"></span>
```

dialog()

A modal dialog

Example:

```
<dialog ng:show="showDialog">
    <div class="modal-header">
        <h4>
            Heading
        </h4>
    </div>
    <div class="modal-body scrollable">
        ...
    </div>
    <div class="modal-footer">
        <a ng:click="..." class="btn btn-default btn-flat">
            Do something
        </a>
    </div>
</dialog>
```

Arguments

- **ngShow** (*expression*) –

- **dialogClass** (*string*) –

floating-toolbar()

A toolbar pinned to the bottom edge. Example:

```
<div class="floating-toolbar-padder"></div>

<floating-toolbar>
  <a ng:click="..." class="btn btn-default btn-flat">
    Do something useful
  </a>
</floating-toolbar>

<!-- accented toolbar for selection actions -->

<floating-toolbar class="accented" ng:show="haveSelectedItems">
  Some action buttons here
</floating-toolbar>
```

ng-enter()

Action handler for Enter key in inputs. Example:

```
<input type="text" ng:enter="commitStuff()" ng:model="..." />
```

progress-spinner()

root-access()

Blocks its inner content if the current user is not a superuser.

smart-progress()

An improved version of ui-bootstrap's progressbar

Arguments

- **animate** (*boolean*) –
- **value** (*float*) –
- **max** (*float*) –
- **text** (*string*) –
- **maxText** (*string*) –

Filters

bytesFilter (*value, precision*)

Arguments

- **value** (*int*) – number of bytes
- **precision** (*int*) – number of fractional digits in the output

Returns string, e.g.: 123.45 KB

ordinalFilter (*value*)

Arguments

- **value** (*int*) –

Returns string, e.g.: 121st

pageFilter(list, page, pageSize)

Provides a page-based view on an array

Arguments

- **list** (array) – input data
- **page** (int) – 1-based page index
- **pageSize** (int) – page size

Returns array

1.6.42 Angular: ajenti.ace

ACE code editor integration

Directives**ace-editor()****Arguments**

- **ngModel** (binding) –
- **aceOptions** (object) – (optional) options for ace.setOptions()

1.6.43 Angular: ajenti.augeas**Services****class augeas()****augeas.get(endpoint)**

Reads an Augeas tree from server side.

Returns promise → AugeasConfig

augeas.set(endpoint, config)

Overwrites an Augeas tree on the server side.

Returns promise

class AugeasNode()**AugeasNode.name****AugeasNode.value****AugeasNode.parent****AugeasNode.children****AugeasNode fullPath()****class AugeasConfig()**

This is a JS doppelganger of normal Augeas API. In particular, it doesn't support advanced XPath syntax, and operates with regular expressions instead.

```
AugeasConfig.get (path)
    Returns AugeasNode

AugeasConfig.set (path, value)
AugeasConfig.model (path)
    Returns a getter/setter function suitable for use as a ngModel
AugeasConfig.insert (path, value, index)
AugeasConfig.remove (path)
AugeasConfig.match (path)
    Returns Array(string)
AugeasConfig.matchNodes (path)
    Returns Array(AugeasNode)
```

1.6.44 Angular: ajenti.filesystem

Services

```
class filesystem()
```

```
filesystem.read (path)
    Returns promise → content of path
filesystem.write (path, content)
    Returns promise
filesystem.list (path)
    Returns promise → array
filesystem.stat (path)
    Returns promise → object
filesystem.chmod (path, mode)
```

Arguments

- **mode** (*int*) – numeric POSIX file mode

```
    Returns promise
```

```
filesystem.createFile (path, mode)
```

Arguments

- **mode** (*int*) – numeric POSIX file mode

```
    Returns promise
```

```
filesystem.createDirectory (path, mode)
```

Arguments

- **mode** (*int*) – numeric POSIX file mode

```
    Returns promise
```

`filesystem.downloadBlob(content, mime, name)`

Launches a browser-side file download

Arguments

- **content** (*string*) – Raw file content
- **mime** (*string*) – MIME type used
- **name** (*string*) – Default file name for saving

Returns promise

Directives

`file-dialog()`

File open/save dialog. Example:

```
<file-dialog
    mode="open"
    ng:show="openDialogVisible"
    on-select="open(item.path)"
    on-cancel="openDialogVisible = false">
</file-dialog>

<file-dialog
    mode="save"
    ng:show="saveDialogVisible"
    on-select="saveAs(path)"
    on-cancel="saveDialogVisible = false"
    name="saveAsName">
</file-dialog>
```

Arguments

- **ngShow** (*expression*) –
- **onSelect** (*expression (item)*) – called after opening or saving a file. *item* is an object with a *path* property.
- **onCancel** (*expression*) – (optional) handler for the cancel button
- **mode** (*string*) – one of open, save
- **name** (*binding*) – (optional) name for the saved file
- **path** (*binding*) – (optional) current

`path-selector()`

An input with a file selection dialog:

```
<path-selector ng:model="filePath"></path-selector>
```

1.6.45 Angular: ajenti.passwd

Services

`class passwd()`

```
passwd.list()  
    Returns promise → array of the users registered in the system
```

1.6.46 Angular: ajenti.services

Services

```
class services()
```

```
    services.getManagers()  
        Returns promise → array of the available service managers  
    services.getServices(managerId)  
        Returns promise → array of the available services in the ServiceManager  
    services.getService(managerId, serviceId)  
        Returns promise → object, gets a single service from the manager  
    services.runOperation(managerId, serviceId, operation)
```

Arguments

- **operation** (*string*) – typically start, stop, restart, reload; depends on the service manager

Returns promise

1.6.47 Angular: ajenti.terminal

Services

```
class terminals()
```

```
    terminals.list()  
        Returns promise → array of opened terminal descriptors  
    terminals.kill(terminalId)  
        Kills a running terminal process
```

Returns promise

```
    terminals.create(options)  
        Creates a new terminal
```

Arguments

- **options.command** (*string*) –
- **options.autoclose** (*boolean*) –

Returns promise → new terminal ID

```
    terminals.full(terminalId)  
        Returns promise → full content of the requested terminal
```

1.6.48 Plugin: aj.plugins.core.api.push

```
class aj.plugins.core.api.push.Push(context)
    A service providing push messages to the client.

    classmethod get(context)

    push(plugin, msg)
        Sends a push message to the client.

    Parameters
        • plugin – routing ID
        • msg – message

    register()
```

1.6.49 Plugin: aj.plugins.core.api.sidebar

```
class aj.plugins.core.api.sidebar.Sidebar(context)

    build()
        Returns a complete tree of sidebar items.

    Returns dict

    classmethod get(context)

class aj.plugins.core.api.sidebar.SidebarItemProvider(context)
    Interface for providing sidebar items.

    provide()
        Should return a list of sidebar items, each in the following format:
```

```
{
    'id': 'optional-id',
    'attach': 'category:general', # id of the attachment point or None for ↴top level
    'name': 'Dashboard',
    'icon': 'bar-chart',
    'url': '/view/dashboard',
    'children': [
        ...
    ]
}
```

Returns list(dict)

1.6.50 Plugin: aj.plugins.core.api.tasks

```
class aj.plugins.core.api.tasks.Task(context, *args, **kwargs)
    Tasks are one-off child processes with progress reporting. This is a base abstract class.

    abort()

    name = None
        Display name
```

push (plugin, message)

An interface to `aj.plugins.core.api.push.Push` usable from inside the task's process

report_progress (message=None, done=None, total=None)

Updates the task's process info.

Parameters

- **message** – text message
- **done** – number of processed items
- **total** – total number of items

run ()

Override this with your task's logic.

send_log_event (method, message, *args, **kwargs)**start ()**

Starts the task's process

```
class aj.plugins.core.api.tasks.TasksService (context)
```

abort (_id)**format_tasks ()****classmethod get (context)****notify (message=None)****remove (_id)****send_update ()****start (task)**

1.6.51 Plugin: aj.plugins.augeas.api

```
class aj.plugins.augeas.api.Augeas (modules=[], loadpath=None)
```

A smarter and faster wrapper around augeas.Augeas.augeas

For faster startup, no modules and lenses are preloaded:

```
aug = Augeas (modules=[ {  
    'name': 'Interfaces', # module name  
    'lens': 'Interfaces.lns', # lens name  
    'incl': [ # included files list  
        self.path,  
        self.path + '.d/*',  
    ]  
} ])
```

Don't forget to call `load()` afterwards.

dump (path)

Dumps contents under `path` to stdout.

get (path)**match (path)**

```

raise_error()
    Extracts error information from Augeas tree and raises AugeasError

save()

set(path, value)

setd(path, value, default=None)
    Sets path to value, or removes path if value == default

class aj.plugins.augeas.api.AugeasEndpoint(context)
    Implement this to provide Augeas trees to the frontend.

        get_augeas()
            Should return a ready-to-use Augeas

        get_root_path()
            Should return an Augeas path of the root node to be provided to the frontend.

        id = None

exception aj.plugins.augeas.api.AugeasError(aug)

```

1.6.52 Plugin: aj.plugins.auth-users.api

1.6.53 Plugin: aj.plugins.dashboard.api

```

class aj.plugins.dashboard.api.Widget(context)
    Base interface for dashboard widgets.

        config_template = None
            Configuration dialog template URL

        get_value(config)
            Override this to return the widget value for the given config dict.

        id = None

        name = None
            Display name

        template = None
            Angular view template URL

```

1.6.54 Plugin: aj.plugins.check_certificates.api

`aj.plugins.check_certificates.api.CertLimitSSL(hostname, port)`
Get the SSL certificate from a host.

Parameters

- **hostname** (*string*) – Hostname
- **port** (*integer*) – Port

Returns OpenSSL cert

Return type OpenSSL cert

`aj.plugins.check_certificates.api.CertLimitSTARTTLS(hostname)`
Get the SSL certificate from host with STARTTLS connection on port 587.

Parameters **hostname** (*string*) – Hostname

Returns OpenSSL cert

Return type OpenSSL cert

`aj.plugins.check_certificates.api.checkOnDom(hostname, port='443')`

Store all details from a certificate into a dict and add a status attribute to display a bootstrap class for the time remaining before renew : - renew the next 7 days : danger - renew the next 14 days: warning - renew the next 28 days : info - or success

Parameters

- **hostname** (*string*) – Hostname
- **port** (*string*) – Port to test, default 443

Returns Dict with all details from the certificate

Return type dict

1.6.55 Plugin: aj.plugins.datetime.api

`class aj.plugins.datetime.api.TZManager(context)`

Abstract interface class for time zone management. All subclasses are stored in the directory managers.

`get_offset()`
`get_tz()`
`list_tz()`
`set_tz(name)`

1.6.56 Plugin: aj.plugins.network.api

`class aj.plugins.network.api.NetworkManager(context)`

Abstract interface to manage all network managers.

`down(iface)`
`get_config()`
`get_hostname()`
`get_state(iface)`
`restart()`
`set_config(config)`
`set_hostname(value)`
`up(iface)`

1.6.57 Plugin: aj.plugins.packages.api

`class aj.plugins.packages.api.Package(manager)`

Basic container class for package informations.

`class aj.plugins.packages.api.PackageManager(context)`

Abstract interface for all managers. Managers are defined in the directory managers.

```

get_apply_cmd(selection)
get_package(_id)
id = None
list(query=None)
name = None
update_command = None
update_lists(progress_callback)

```

1.6.58 Plugin: aj.plugins.power.api

```

class aj.plugins.power.api.PowerManager(context)
    Utility class for power management.

    classmethod get(context)
        get_adapters()
            Read the list of adapters from /proc/acpi/ac_adapter.

            Returns List of adapters, one per dict
            Return type list of dict

        get_batteriesReturns All batteries informations
            Return type list of dict

        hibernatepoweroff()
            Basically power off the system.

        reboot()
            Basically reboot the system.

        suspend()
            Basically suspend the system.

```

1.6.59 Plugin: aj.plugins.services.api

```

class aj.plugins.services.api.Service(manager)
    Basic class to store service informations.

class aj.plugins.services.api.ServiceManager
    Abstract interface for all managers.

    get_service(_id)
        id = None
        kill(_id)
        list()
        name = None

```

```
restart (_id)
start (_id)
stop (_id)

exception aj.plugins.services.api.ServiceOperationError (inner)
Exception class for services.
```

Python Module Index

a

aj, 44
aj.api.endpoint, 48
aj.api.http, 45
aj.config, 49
aj.core, 49
aj.entry, 50
aj.http, 50
aj.plugins, 52
aj.plugins.augeas.api, 64
aj.plugins.check_certificates.api, 65
aj.plugins.core.api.push, 63
aj.plugins.core.api.sidebar, 63
aj.plugins.core.api.tasks, 63
aj.plugins.dashboard.api, 65
aj.plugins.datetime.api, 66
aj.plugins.network.api, 66
aj.plugins.packages.api, 66
aj.plugins.power.api, 67
aj.plugins.services.api, 67

j

jadi, 43

Index

A

abort () (*aj.plugins.core.api.tasks.Task method*), 63
abort () (*aj.plugins.core.api.tasks.TasksService method*), 64
ace-editor () (*built-in function*), 59
add_header () (*aj.http.HttpContext method*), 51
aj (*module*), 44
aj.api.endpoint (*module*), 48
aj.api.http (*module*), 45
aj.config (*module*), 49
aj.core (*module*), 49
aj.entry (*module*), 50
aj.http (*module*), 50
aj.plugins (*module*), 52
aj.plugins.augeas.api (*module*), 64
aj.plugins.check_certificates.api (*module*), 65
aj.plugins.core.api.push (*module*), 63
aj.plugins.core.api.sidebar (*module*), 63
aj.plugins.core.api.tasks (*module*), 63
aj.plugins.dashboard.api (*module*), 65
aj.plugins.datetime.api (*module*), 66
aj.plugins.network.api (*module*), 66
aj.plugins.packages.api (*module*), 66
aj.plugins.power.api (*module*), 67
aj.plugins.services.api (*module*), 67
all () (*jadi. method*), 43
any () (*jadi. method*), 44
Augeas (*class in aj.plugins.augeas.api*), 64
augeas () (*class*), 59
augeas.get () (*augeas method*), 59
augeas.set () (*augeas method*), 59
AugeasConfig () (*class*), 59
AugeasConfig.get () (*AugeasConfig method*), 59
AugeasConfig.insert () (*AugeasConfig method*), 60
AugeasConfig.match () (*AugeasConfig method*), 60
AugeasConfig.matchNodes () (*AugeasConfig*

method), 60

AugeasConfig.model () (*AugeasConfig method*), 60
AugeasConfig.remove () (*AugeasConfig method*), 60
AugeasConfig.set () (*AugeasConfig method*), 60
AugeasEndpoint (*class in aj.plugins.augeas.api*), 65
AugeasError, 65
AugeasNode () (*class*), 59
AugeasNode.children (*global variable or constant*), 59
AugeasNode fullPath () (*AugeasNode method*), 59
AugeasNode.name (*global variable or constant*), 59
AugeasNode.parent (*global variable or constant*), 59
AugeasNode.value (*global variable or constant*), 59
autofocus () (*built-in function*), 57

B

BaseHttpHandler (*class in aj.api.http*), 45
BinaryDependency (*class in aj.plugins*), 53
BinaryDependency.Unsatisfied, 54
body (*aj.http.HttpContext attribute*), 51
build () (*aj.plugins.core.api.sidebar Sidebar method*), 63
build_exception () (*aj.plugins.Dependency method*), 53
bytesFilter () (*built-in function*), 58

C

CertLimitSSL () (*in module aj.plugins.check_certificates.api*), 65
CertLimitSTARTTLS () (*in module aj.plugins.check_certificates.api*), 65
CGIFieldStorage (*class in aj.http*), 50
check () (*aj.plugins.Dependency method*), 53
checkbox () (*built-in function*), 57
checkOnDom () (*in module aj.plugins.check_certificates.api*), 66

classes () (*jadi. method*), 44
component () (*in module jadi*), 44
config (*in module aj*), 44
config() (*class*), 54
config.data (*global variable or constant*), 54
config.getUserConfig () (*config method*), 55
config.load () (*config method*), 54
config.save () (*config method*), 54
config.setUserConfig () (*config method*), 55
config_template (*aj.plugins.dashboard.api.Widget attribute*), 65
Context (*class in jadi*), 44
copy () (*in module aj.api.http*), 46
core () (*class*), 55
core.pageReload () (*core method*), 55
core.restart () (*core method*), 55

D

debug (*in module aj*), 44
delete () (*in module aj.api.http*), 46
Dependency (*class in aj.plugins*), 53
Dependency.Unsatisfied, 53
describe () (*aj.plugins.Dependency.Unsatisfied method*), 53
describe () (*aj.plugins.PluginCrashed method*), 53
description (*aj.plugins.BinaryDependency attribute*), 54
description (*aj.plugins.FileDependency attribute*), 54
description (*aj.plugins.ModuleDependency attribute*), 53
description (*aj.plugins.OptionalPluginDependency attribute*), 53
description (*aj.plugins.PluginDependency attribute*), 53
deserialize () (*aj.http(HttpContext class method)*, 51
destroy () (*aj.api.http.SocketEndpoint method*), 45
dialog () (*built-in function*), 57
DirectoryPluginProvider (*class in aj.plugins*), 52
dispatch () (*aj.http.HttpRoot method*), 52
down () (*aj.plugins.network.api.NetworkManager method*), 66
dump () (*aj.plugins.augeas.api.Augeas method*), 64
dump_env () (*aj.http(HttpContext method*), 51

E

endpoint () (*in module aj.api.endpoint*), 49
EndpointError, 48
EndpointReturn, 48
env (*aj.http(HttpContext attribute*), 50
exit () (*in module aj*), 44

F

fallthrough () (*aj.http.HttpContext method*), 51
file () (*aj.http.HttpContext method*), 51
file-dialog () (*built-in function*), 61
FileDependency (*class in aj.plugins*), 54
FileDependency.Unsatisfied, 54
filesystem () (*class*), 60
filesystem.chmod () (*filesystem method*), 60
filesystem.createDirectory () (*filesystem method*), 60
filesystem.createFile () (*filesystem method*), 60
filesystem.downloadBlob () (*filesystem method*), 60
filesystem.list () (*filesystem method*), 60
filesystem.read () (*filesystem method*), 60
filesystem.stat () (*filesystem method*), 60
filesystem.write () (*filesystem method*), 60
floating-toolbar () (*built-in function*), 58
format_tasks () (*aj.plugins.core.api.TasksService method*), 64

G

get () (*aj.config.UserConfigService class method*), 49
get () (*aj.plugins.augeas.api.Augeas method*), 64
get () (*aj.plugins.core.api.push.Push class method*), 63
get () (*aj.plugins.core.api.sidebar Sidebar class method*), 63
get () (*aj.plugins.core.api.tasks TasksService class method*), 64
get () (*aj.plugins.PluginManager class method*), 54
get () (*aj.plugins.power.api.PowerManager class method*), 67
get () (*in module aj.api.http*), 46
get () (*jadi. method*), 44
get_adapters () (*aj.plugins.power.api.PowerManager method*), 67
get_apply_cmd () (*aj.plugins.packages.api.PackageManager method*), 66
get_augeas () (*aj.plugins.augeas.api.AugeasEndpoint method*), 65
get_batteries () (*aj.plugins.power.api.PowerManager method*), 67
get_cleaned_env () (*aj.http(HttpContext method*), 51
get_component () (*jadi.Context method*), 44
get_components () (*jadi.Context method*), 44
get_config () (*aj.plugins.network.api.NetworkManager method*), 66
get_content_path () (*aj.plugins.PluginManager method*), 54
get_crash () (*aj.plugins.PluginManager method*), 54
get_fqdn () (*in module jadi*), 43

```

get_hostname () (aj.plugins.network.api.NetworkManager)
    method), 66
get_loaded_plugins_list ()
    (aj.plugins.PluginManager method), 54
get_offset () (aj.plugins.datetime.api.TZManager)
    method), 66
get_package () (aj.plugins.packages.api.PackageManager)
    method), 67
get_provider () (aj.config.UserConfigService)
    method), 49
get_root_path () (aj.plugins.augeas.api.AugeasEndpoint)
    method), 65
get_service () (aj.plugins.services.api.ServiceManager)
    method), 67
get_service () (jadi.Context method), 44
get_state () (aj.plugins.network.api.NetworkManager)
    method), 66
get_tz () (aj.plugins.datetime.api.TZManager)
    method), 66
get_value () (aj.plugins.dashboard.api.Widget)
    method), 65
gzip () (aj.http.HttpContext method), 51

```

H

```

handle () (aj.api.http.BaseHttpHandler method), 45
handle () (aj.api.http.HttpMasterMiddleware method),
    45
handle () (aj.api.http.HttpMiddleware method), 45
handle () (aj.api.http.HttpPlugin method), 45
handle () (aj.http.HttpMiddlewareAggregator method),
    52
handle_crash () (in module aj.entry), 50
head () (in module aj.api.http), 46
headers (aj.http.HttpContext attribute), 51
hibernate () (aj.plugins.power.api.PowerManager)
    method), 67
hotkeys () (class), 55
hotkeys.ENTER, ESC (global variable or constant),
    55
hotkeys.on () (hotkeys method), 55
HttpContext (class in aj.http), 50
HttpMasterMiddleware (class in aj.api.http), 45
HttpMiddleware (class in aj.api.http), 45
HttpMiddlewareAggregator (class in aj.http), 52
HttpPlugin (class in aj.api.http), 45
HttpRoot (class in aj.http), 52

```

I

```

id (aj.plugins.augeas.api.AugeasEndpoint attribute), 65
id (aj.plugins.dashboard.api.Widget attribute), 65
id (aj.plugins.packages.api.PackageManager attribute),
    67
id (aj.plugins.services.api.ServiceManager attribute), 67
identity () (class), 55

```

```

identity.auth () (identity method), 55
identity.effective (global variable or constant),
    55
identity.elevate () (identity method), 55
identity.isSuperuser (global variable or constant), 55
identity.login () (identity method), 55
identity.logout () (identity method), 55
identity.machine.name (global variable or constant), 55
identity.user (global variable or constant), 55
init () (in module aj), 44
interface () (in module jadi), 43
is_satisfied () (aj.plugins.BinaryDependency
    method), 54
is_satisfied () (aj.plugins.FileDependency
    method), 54
is_satisfied () (aj.plugins.ModuleDependency
    method), 53
is_satisfied () (aj.plugins.OptionalPluginDependency
    method), 53
is_satisfied () (aj.plugins.PluginDependency
    method), 53

```

J

```

jadi (module), 43
json_body () (aj.http.HttpContext method), 51

```

K

```

kill () (aj.plugins.services.api.ServiceManager
    method), 67

```

L

```

list () (aj.plugins.packages.api.PackageManager
    method), 67
list () (aj.plugins.services.api.ServiceManager
    method), 67
list_tz () (aj.plugins.datetime.api.TZManager
    method), 66
load_all_from () (aj.plugins.PluginManager
    method), 54
lock () (in module aj.api.http), 46

```

M

```

make_file () (aj.http.CGIFieldStorage method), 50
match () (aj.plugins.augeas.api.Augeas method), 64
messagebox () (class), 56
messagebox.show () (messagebox method), 56
method (aj.http(HttpContext attribute), 50
mkcol () (in module aj.api.http), 47
ModuleDependency (class in aj.plugins), 53
ModuleDependency.Unsatisfied, 53
move () (in module aj.api.http), 47

```

N

name (*aj.plugins.core.api.tasks.Task* attribute), 63
name (*aj.plugins.dashboard.api.Widget* attribute), 65
name (*aj.plugins.packages.api.PackageManager* attribute), 67
name (*aj.plugins.services.api.ServiceManager* attribute), 67
NetworkManager (*class in aj.plugins.network.api*), 66
ng-enter () (*built-in function*), 58
NoImplementationError, 44
notify() (*aj.plugins.core.api.tasks.TasksService method*), 64
notify() (*class*), 56
notify.custom() (*notify method*), 56
notify.error() (*notify method*), 56
notify.info() (*notify method*), 56
notify.success() (*notify method*), 56
notify.warning() (*notify method*), 56

O

on_connect() (*aj.api.http.SocketEndpoint method*), 45
on_disconnect() (*aj.api.http.SocketEndpoint method*), 45
on_message() (*aj.api.http.SocketEndpoint method*), 45
OptionalPluginDependency (*class in aj.plugins*), 53
OptionalPluginDependency.Unsatisfied, 53
options() (*in module aj.api.http*), 47
ordinalFilter() (*built-in function*), 58

P

Package (*class in aj.plugins.packages.api*), 66
PackageManager (*class in aj.plugins.packages.api*), 66
pageFilter() (*built-in function*), 58
pageTitle() (*class*), 56
pageTitle.set() (*pageTitle method*), 56
passwd() (*class*), 61
passwd.list() (*passwd method*), 61
patch() (*in module aj.api.http*), 47
path (*aj.http.HttpContext* attribute), 50
path-selector() (*built-in function*), 61
platform (*in module aj*), 44
platform_string (*in module aj*), 44
platform_unmapped (*in module aj*), 44
plugin (*aj.api.http.SocketEndpoint* attribute), 46
PluginCrashed, 52
PluginDependency (*class in aj.plugins*), 53
PluginDependency.Unsatisfied, 53
PluginLoadError, 52

PluginManager (*class in aj.plugins*), 54
PluginProvider (*class in aj.plugins*), 52
post() (*in module aj.api.http*), 47
PowerManager (*class in aj.plugins.power.api*), 67
poweroff() (*aj.plugins.power.api.PowerManager method*), 67
progress-spinner() (*built-in function*), 58
propfind() (*in module aj.api.http*), 47
proppatch() (*in module aj.api.http*), 48
provide() (*aj.plugins.core.api.sidebar SidebarItemProvider method*), 63
provide() (*aj.plugins.DirectoryPluginProvider method*), 52
provide() (*aj.plugins.PluginProvider method*), 52
provide() (*aj.plugins.PythonPathPluginProvider method*), 52
Push (*class in aj.plugins.core.api.push*), 63
push() (*aj.plugins.core.api.push.Push method*), 63
push() (*aj.plugins.core.api.tasks.Task method*), 63
push() (*class*), 56
put() (*in module aj.api.http*), 48
PythonPathPluginProvider (*class in aj.plugins*), 52

Q

query (*aj.http.HttpContext* attribute), 51

R

raise_error() (*aj.plugins.augeas.api.Augeas method*), 64
reason() (*aj.plugins.BinaryDependency.Unsatisfied method*), 54
reason() (*aj.plugins.Dependency.Unsatisfied method*), 53
reason() (*aj.plugins.FileDependency.Unsatisfied method*), 54
reason() (*aj.plugins.ModuleDependency.Unsatisfied method*), 53
reason() (*aj.plugins.OptionalPluginDependency.Unsatisfied method*), 53
reason() (*aj.plugins.PluginDependency.Unsatisfied method*), 53
reboot() (*aj.plugins.power.api.PowerManager method*), 67
redirect() (*aj.http.HttpContext* method), 51
register() (*aj.plugins.core.api.push.Push method*), 63
remove() (*aj.plugins.core.api.tasks.TasksService method*), 64
remove_header() (*aj.http.HttpContext* method), 51
report_progress() (*aj.plugins.core.api.tasks.Task method*), 64
requests_decorator_generator() (*in module aj.api.http*), 48

respond() (*aj.http.HttpContext method*), 51
 respond_forbidden() (*aj.http.HttpContext method*), 51
 respond_not_found() (*aj.http(HttpContext method)*, 51
 respond_ok() (*aj.http(HttpContext method)*, 52
 respond_server_error() (*aj.http(HttpContext method)*, 52
 respond_unauthenticated() (*aj.http(HttpContext method)*, 52
 response_ready (*aj.http(HttpContext attribute)*), 51
 restart() (*aj.plugins.network.api.NetworkManager method*), 66
 restart() (*aj.plugins.services.api.ServiceManager method*), 67
 restart() (*in module aj*), 45
 restart() (*in module aj.core*), 49
 root-access () (*built-in function*), 58
 run() (*aj.plugins.core.api.tasks.Task method*), 64
 run() (*in module aj.core*), 49
 run_response() (*aj.http(HttpContext method)*, 52

S

save() (*aj.plugins.augeas.api.Augeas method*), 65
 send() (*aj.api.http.SocketEndpoint method*), 46
 send_log_event() (*aj.plugins.core.api.tasks.Task method*), 64
 send_update() (*aj.plugins.core.api.tasks.TasksService method*), 64
 serialize() (*aj.http(HttpContext method)*, 52
 server (*in module aj*), 44
 Service (*class in aj.plugins.services.api*), 67
 service() (*in module jadi*), 44
 ServiceManager (*class in aj.plugins.services.api*), 67
 ServiceOperationError, 68
 services() (*class*), 62
 services.getManagers() (*services method*), 62
 services.getService() (*services method*), 62
 services.getServices() (*services method*), 62
 services.runOperation() (*services method*), 62
 set() (*aj.plugins.augeas.api.Augeas method*), 65
 set_config() (*aj.plugins.network.api.NetworkManager method*), 66
 set_hostname() (*aj.plugins.network.api.NetworkManager method*), 66
 set_tz() (*aj.plugins.datetime.api.TZManager method*), 66
 setd() (*aj.plugins.augeas.api.Augeas method*), 65
 Sidebar (*class in aj.plugins.core.api.sidebar*), 63
 SidebarItemProvider (*class in aj.plugins.core.api.sidebar*), 63
 smart_progress() (*built-in function*), 58
 SocketEndpoint (*class in aj.api.http*), 45
 spawn() (*aj.api.http.SocketEndpoint method*), 46

start() (*aj.plugins.core.api.tasks.Task method*), 64
 start() (*aj.plugins.core.api.tasks.TasksService method*), 64
 start() (*aj.plugins.services.api.ServiceManager method*), 68
 start() (*in module aj.entry*), 50
 stop() (*aj.plugins.services.api.ServiceManager method*), 68
 suspend() (*aj.plugins.power.api.PowerManager method*), 67

T

Task (*class in aj.plugins.core.api.tasks*), 63
 tasks() (*class*), 57
 tasks.start() (*tasks method*), 57
 tasks.tasks (*global variable or constant*), 57
 TasksService (*class in aj.plugins.core.api.tasks*), 64
 template (*aj.plugins.dashboard.api.Widget attribute*), 65

terminals() (*class*), 62
 terminals.create() (*terminals method*), 62
 terminals.full() (*terminals method*), 62
 terminals.kill() (*terminals method*), 62
 terminals.list() (*terminals method*), 62
 TZManager (*class in aj.plugins.datetime.api*), 66

U

unlock() (*in module aj.api.http*), 48
 up() (*aj.plugins.network.api.NetworkManager method*), 66
 update_command (*aj.plugins.packages.api.PackageManager attribute*), 67
 update_lists() (*aj.plugins.packages.api.PackageManager method*), 67
 url() (*in module aj.api.http*), 48
 UserConfigService (*class in aj.config*), 49

V

value (*aj.plugins.Dependency attribute*), 53
 version (*in module aj*), 44

W

Widget (*class in aj.plugins.dashboard.api*), 65

Y

yaml_loader (*aj.plugins.Dependency attribute*), 53
 yaml_tag (*aj.plugins.BinaryDependency attribute*), 54
 yaml_tag (*aj.plugins.Dependency attribute*), 53
 yaml_tag (*aj.plugins.FileDependency attribute*), 54
 yaml_tag (*aj.plugins.ModuleDependency attribute*), 53
 yaml_tag (*aj.plugins.OptionalPluginDependency attribute*), 53
 yaml_tag (*aj.plugins.PluginDependency attribute*), 53