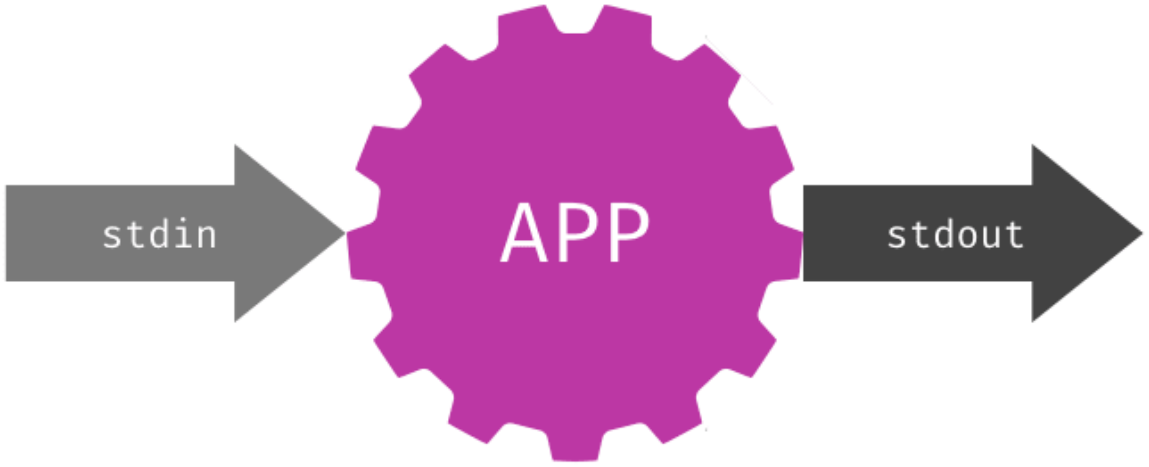
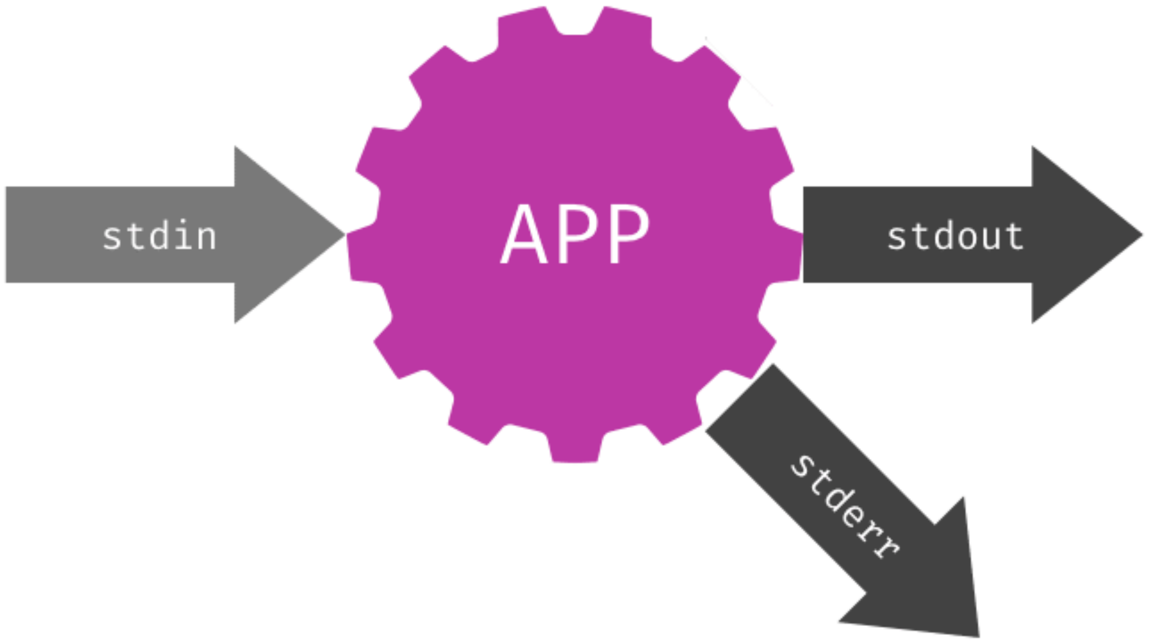
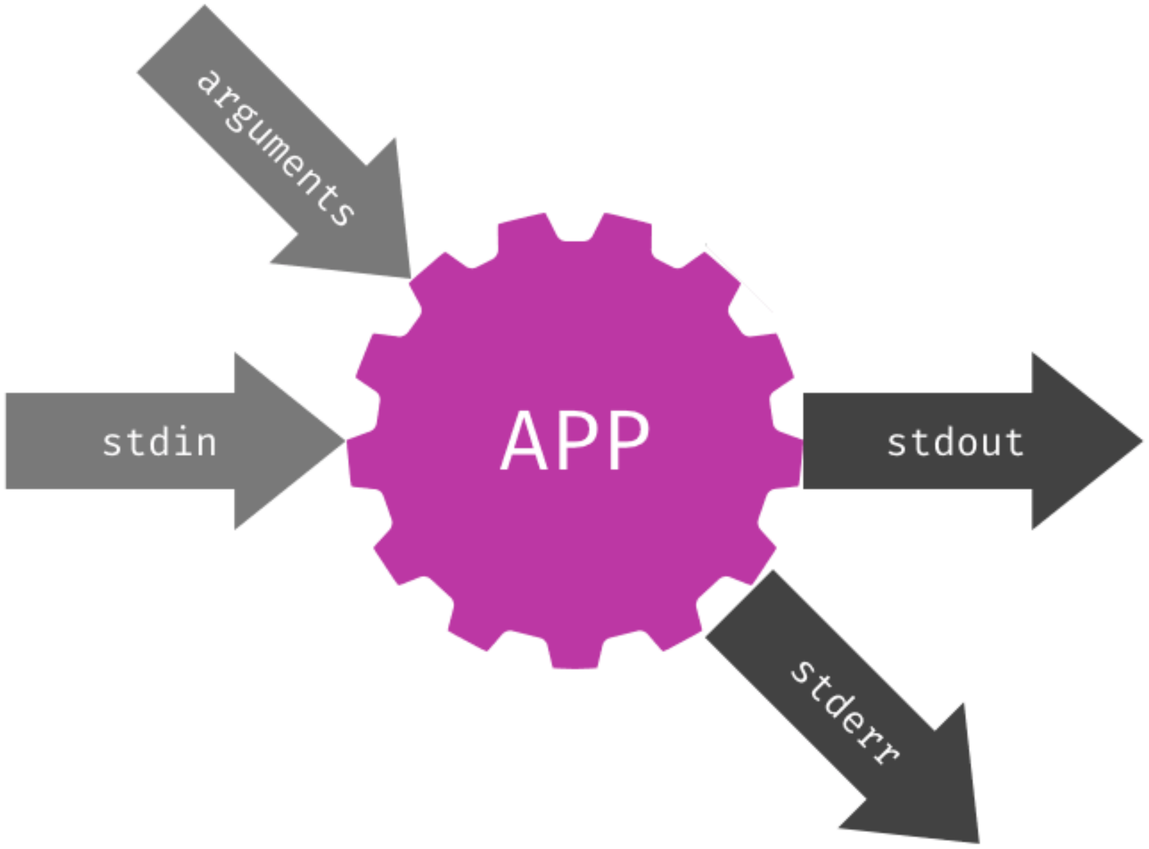
PROTECTING PLAINTEXT SECRETS IN CONFIGURATION FILES

Moisés Guimarães - Red Hat

APP


stdin
APP
stdout


stdin
APP
stdout
stderr


arguments
stdin
APP
stdout
stderr

ARGPARSE

ARGPARSE

```python
import argparse

parser = argparse.ArgumentParser(description='Process some integers.')

parser.add_argument('integers', metavar='N', type=int, nargs='+',
                    help='an integer for the accumulator')

parser.add_argument('--sum', dest='accumulate', action='store_const',
                    const=sum, default=max,
                    help='sum the integers (default: find the max)')

args = parser.parse_args()
print(args.accumulate(args.integers))
```

ARGPARSE

```
$
```

ARGPARSE

```
$ python prog.py -h
```

ARGPARSE

```
$ python prog.py -h
usage: prog.py [-h] [--sum] N [N ...]

Process some integers.

positional arguments:
N            an integer for the accumulator

optional arguments:
-h, --help  show this help message and exit
--sum       sum the integers (default: find the max)
```

ARGPARSE

```
$
```

ARGPARSE

```
$ python prog.py 1 2 3 4
```

ARGPARSE

```
$ python prog.py 1 2 3 4
4
$
```

ARGPARSE

```
$ python prog.py 1 2 3 4
4
$ python prog.py 1 2 3 4 --sum
```

ARGPARSE

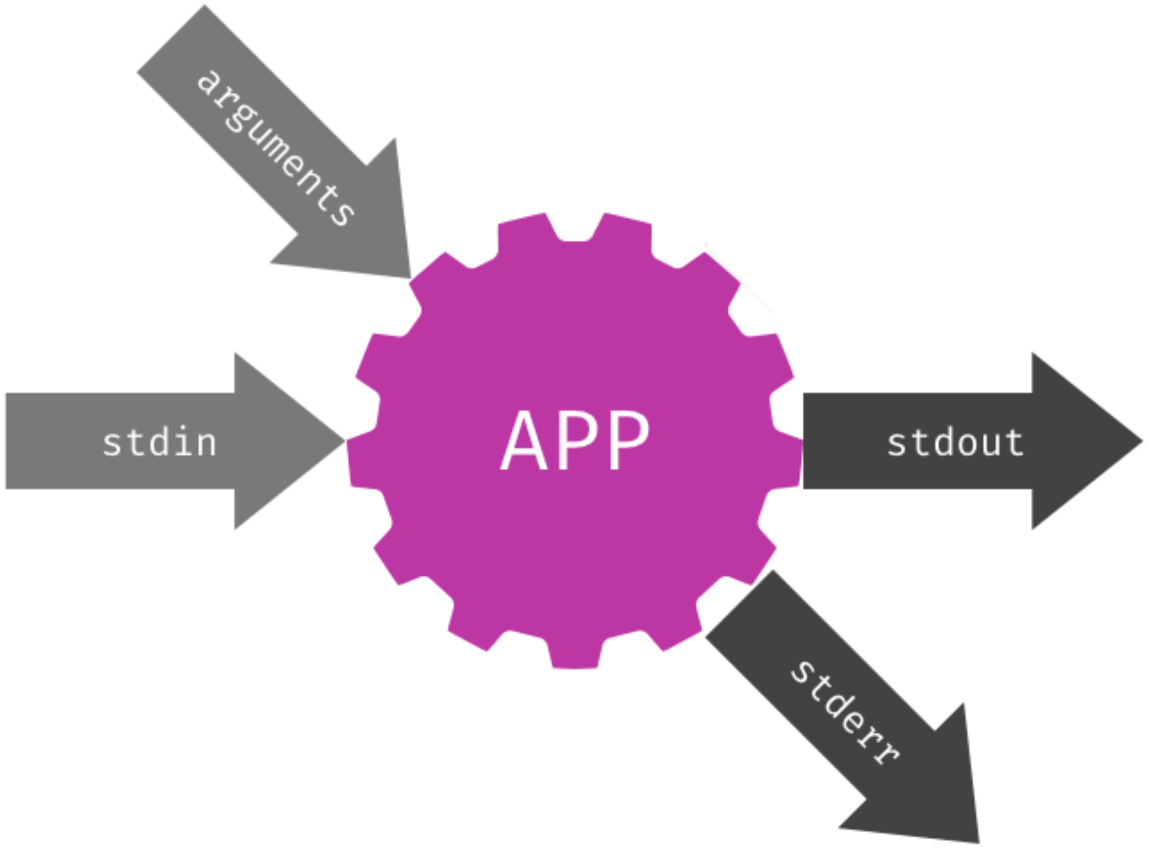
```
$ python prog.py 1 2 3 4
4
$ python prog.py 1 2 3 4 --sum
10
$
```

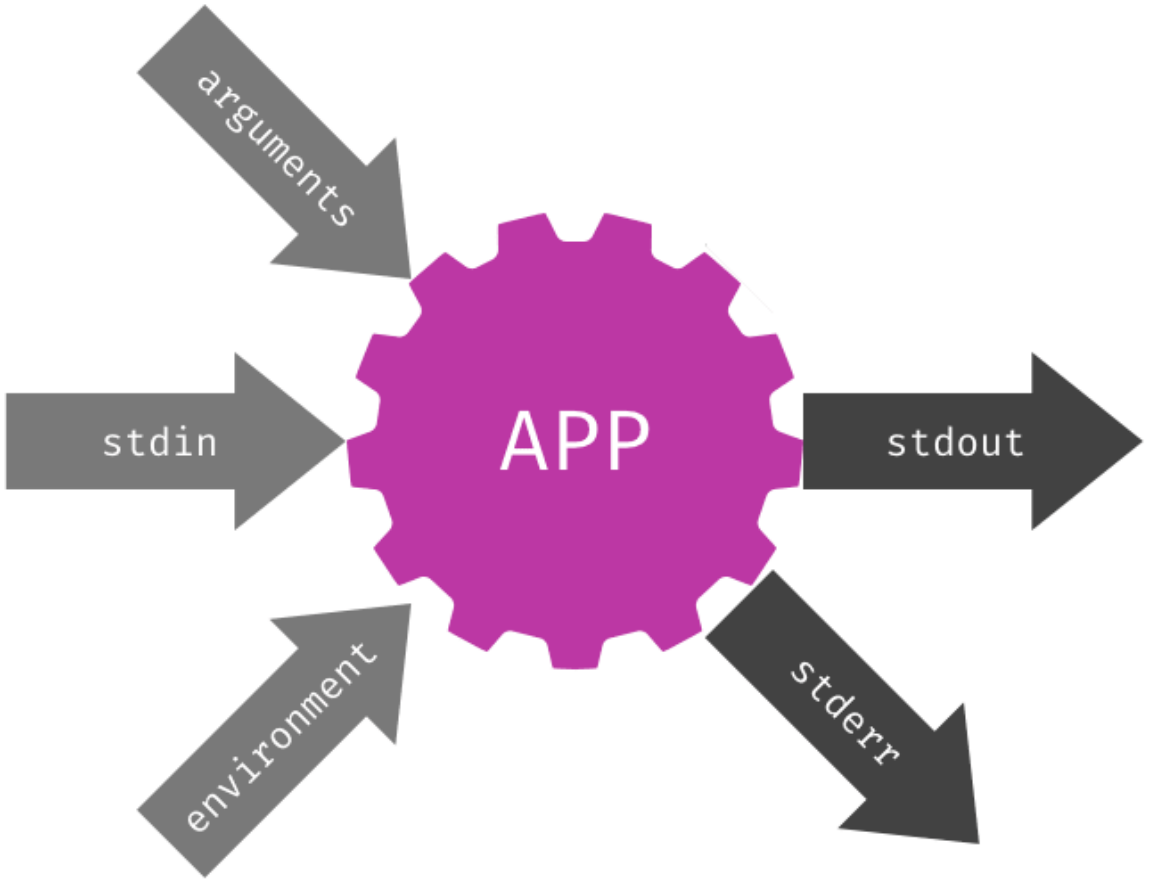
ARGPARSE

```
$ python prog.py 1 2 3 4
4
$ python prog.py 1 2 3 4 --sum
10
$ python prog.py a b c
```

ARGPARSE

```
$ python prog.py 1 2 3 4
4
$ python prog.py 1 2 3 4 --sum
10
$ python prog.py a b c
usage: prog.py [-h] [--sum] N [N ...]
prog.py: error: argument N: invalid int value: 'a'
$
```


arguments
stdin
APP
stdout
stderr


arguments
stdin
environment
APP
stdout
stderr

OS.ENVIRON

OS.ENVIRON

```
>>>
```

OS.ENVIRON

```
>>> import os
```

OS.ENVIRON

```
>>> import os
>>>
```

OS.ENVIRON

```
>>> import os
>>> os.environ["HOME"]
```

OS.ENVIRON

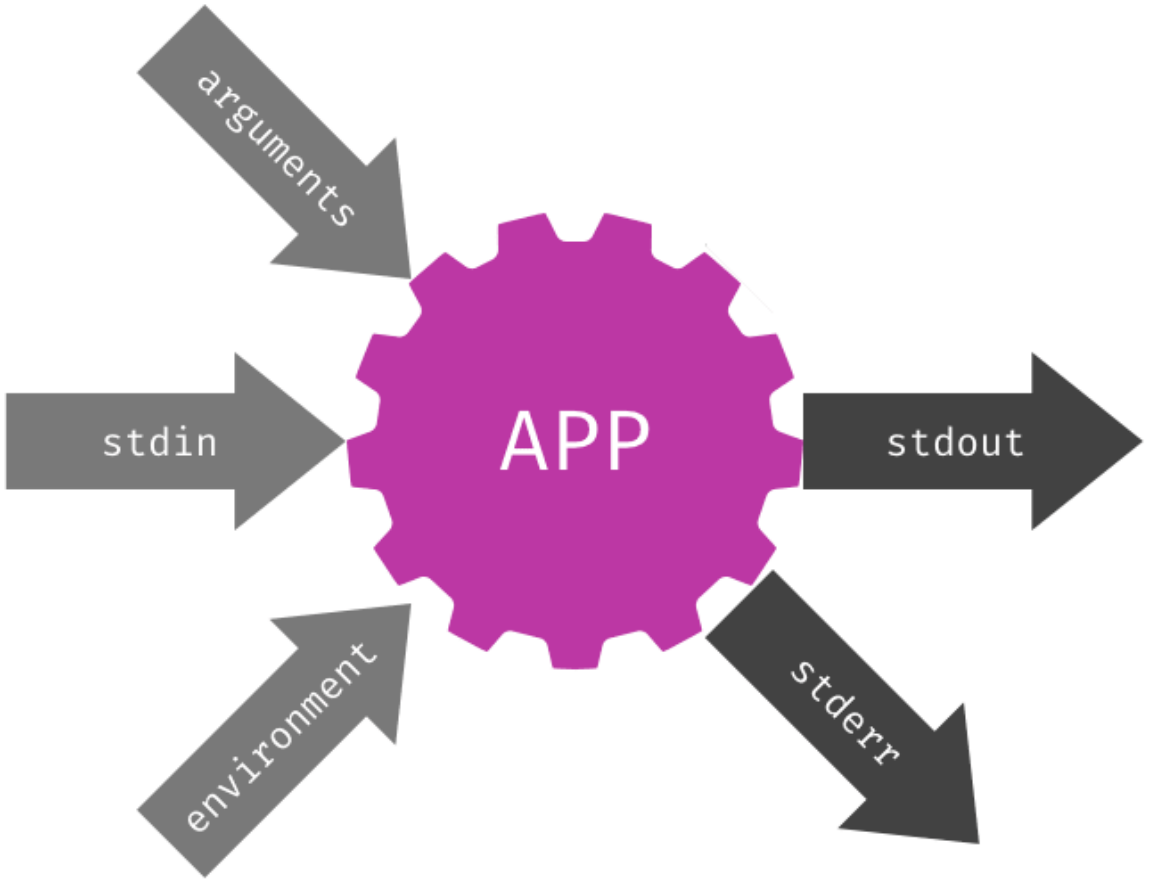
```
>>> import os
>>> os.environ["HOME"]
'/Users/moisesguimaraes'
>>>
```

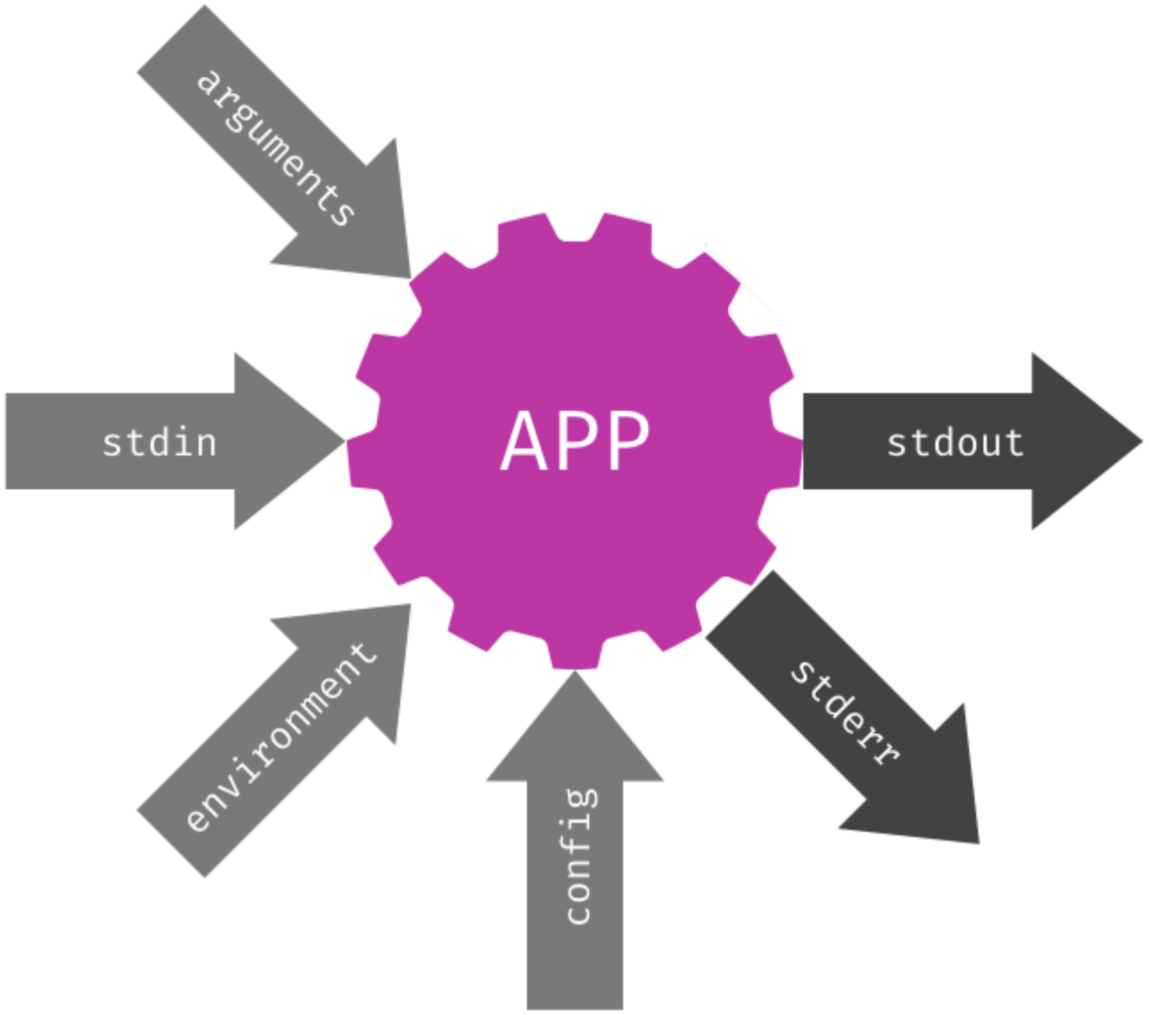
OS.ENVIRON

```
>>> import os
>>> os.environ["HOME"]
'/Users/moisesguimaraes'
>>> os.environ["HOMEBREW_GITHUB_API_TOKEN"]
```

OS.ENVIRON

```
>>> import os
>>> os.environ["HOME"]
'/Users/moisesguimaraes'
>>> os.environ["HOMEBREW_GITHUB_API_TOKEN"]
😏
>>>
```


arguments
stdin
environment
APP
stdout
stderr


arguments
stdin
environment
config
APP
stdout
stderr

CONFIGPARSER

CONFIGPARSER

```
[DEFAULT]
Compression = yes
CompressionLevel = 9
ForwardX11 = yes

[bitbucket.org]
User = hg

[topsecret.server.com]
Port = 50022
ForwardX11 = no
```

CONFIGPARSER

```
>>>
```

CONFIGPARSER

```
>>> import configparser
```

CONFIGPARSER

```
>>> import configparser
>>>
```

CONFIGPARSER

```
>>> import configparser
>>> config = configparser.ConfigParser()
```

CONFIGPARSER

```
>>> import configparser
>>> config = configparser.ConfigParser()
>>>
```

CONFIGPARSER

```
>>> import configparser
>>> config = configparser.ConfigParser()
>>> config.read('example.ini')
```

CONFIGPARSER

```
>>> import configparser
>>> config = configparser.ConfigParser()
>>> config.read('example.ini')
>>>
```

CONFIGPARSER

```
>>> import configparser
>>> config = configparser.ConfigParser()
>>> config.read('example.ini')
>>> config.sections()
```

CONFIGPARSER

```
>>> import configparser
>>> config = configparser.ConfigParser()
>>> config.read('example.ini')
>>> config.sections()
['bitbucket.org', 'topsecret.server.com']
>>>
```

CONFIGPARSER

```
>>> import configparser
>>> config = configparser.ConfigParser()
>>> config.read('example.ini')
>>> config.sections()
['bitbucket.org', 'topsecret.server.com']
>>> config['bitbucket.org']['User']
```

CONFIGPARSER

```
>>> import configparser
>>> config = configparser.ConfigParser()
>>> config.read('example.ini')
>>> config.sections()
['bitbucket.org', 'topsecret.server.com']
>>> config['bitbucket.org']['User']
'hg'
>>>
```

CONFIGPARSER

```
>>> import configparser
>>> config = configparser.ConfigParser()
>>> config.read('example.ini')
>>> config.sections()
['bitbucket.org', 'topsecret.server.com']
>>> config['bitbucket.org']['User']
'hg'
>>> config['DEFAULT']['Compression']
```

CONFIGPARSER

```
>>> import configparser
>>> config = configparser.ConfigParser()
>>> config.read('example.ini')
>>> config.sections()
['bitbucket.org', 'topsecret.server.com']
>>> config['bitbucket.org']['User']
'hg'
>>> config['DEFAULT']['Compression']
'yes'
>>>
```

CONFIGPARSER

```
>>> import configparser
>>> config = configparser.ConfigParser()
>>> config.read('example.ini')
>>> config.sections()
['bitbucket.org', 'topsecret.server.com']
>>> config['bitbucket.org']['User']
'hg'
>>> config['DEFAULT']['Compression']
'yes'
>>> for key in config['bitbucket.org']:
```

CONFIGPARSER

```
>>> import configparser
>>> config = configparser.ConfigParser()
>>> config.read('example.ini')
>>> config.sections()
['bitbucket.org', 'topsecret.server.com']
>>> config['bitbucket.org']['User']
'hg'
>>> config['DEFAULT']['Compression']
'yes'
>>> for key in config['bitbucket.org']:
...     print(key)
...
```

CONFIGPARSER

```
>>> import configparser
>>> config = configparser.ConfigParser()
>>> config.read('example.ini')
>>> config.sections()
['bitbucket.org', 'topsecret.server.com']
>>> config['bitbucket.org']['User']
'hg'
>>> config['DEFAULT']['Compression']
'yes'
>>> for key in config['bitbucket.org']:
...     print(key)
...
user
compressionlevel
compression
forwardx11
>>>
```

CONFIGPARSER

```
>>> import configparser
>>> config = configparser.ConfigParser()
>>> config.read('example.ini')
>>> config.sections()
['bitbucket.org', 'topsecret.server.com']
>>> config['bitbucket.org']['User']
'hg'
>>> config['DEFAULT']['Compression']
'yes'
>>> for key in config['bitbucket.org']:
...     print(key)
...
user
compressionlevel
compression
forwardx11
>>> config['bitbucket.org']['ForwardX11']
```

CONFIGPARSER

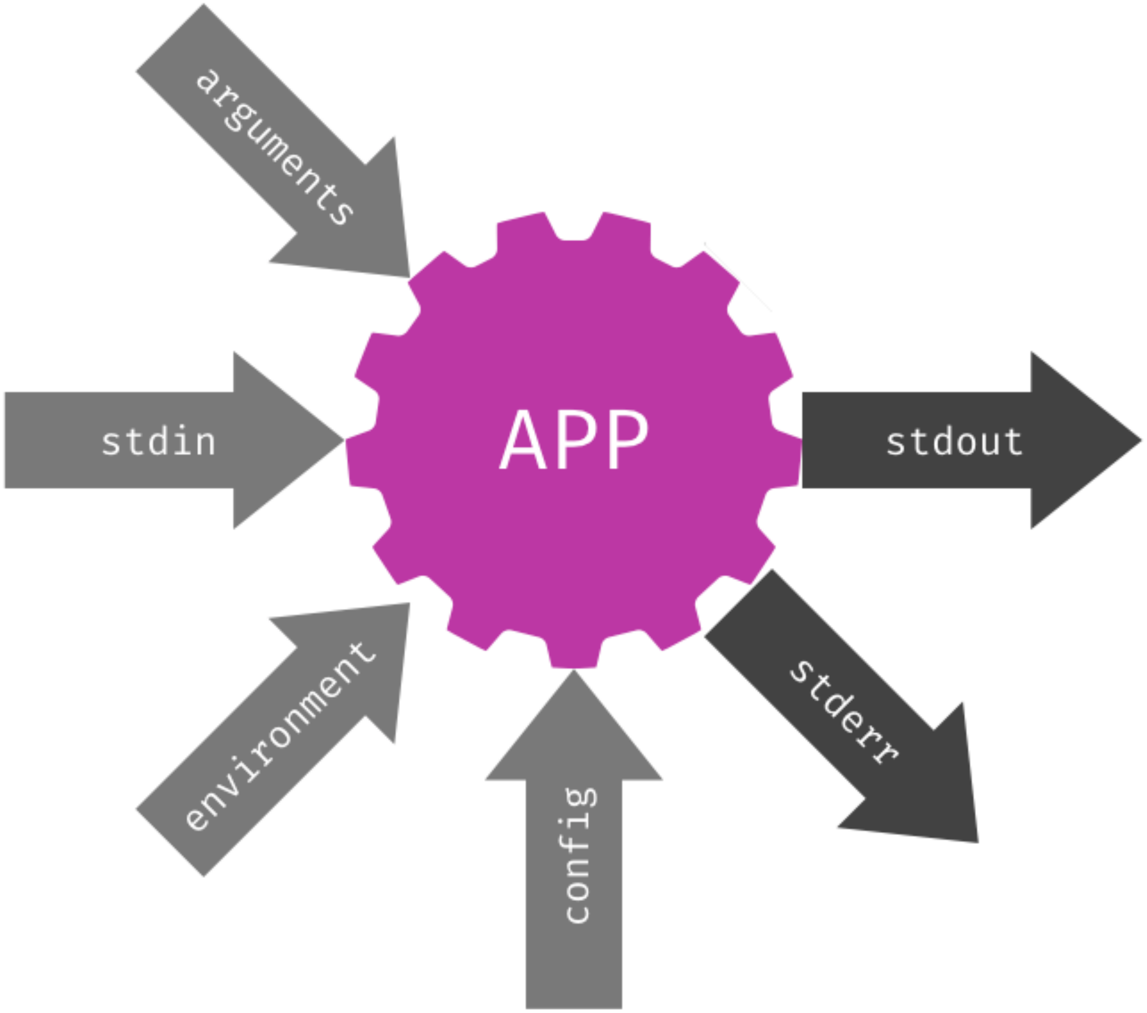
```
>>> import configparser
>>> config = configparser.ConfigParser()
>>> config.read('example.ini')
>>> config.sections()
['bitbucket.org', 'topsecret.server.com']
>>> config['bitbucket.org']['User']
'hg'
>>> config['DEFAULT']['Compression']
'yes'
>>> for key in config['bitbucket.org']:
...     print(key)
...
user
compressionlevel
compression
forwardx11
>>> config['bitbucket.org']['ForwardX11']
'yes'
>>>
```

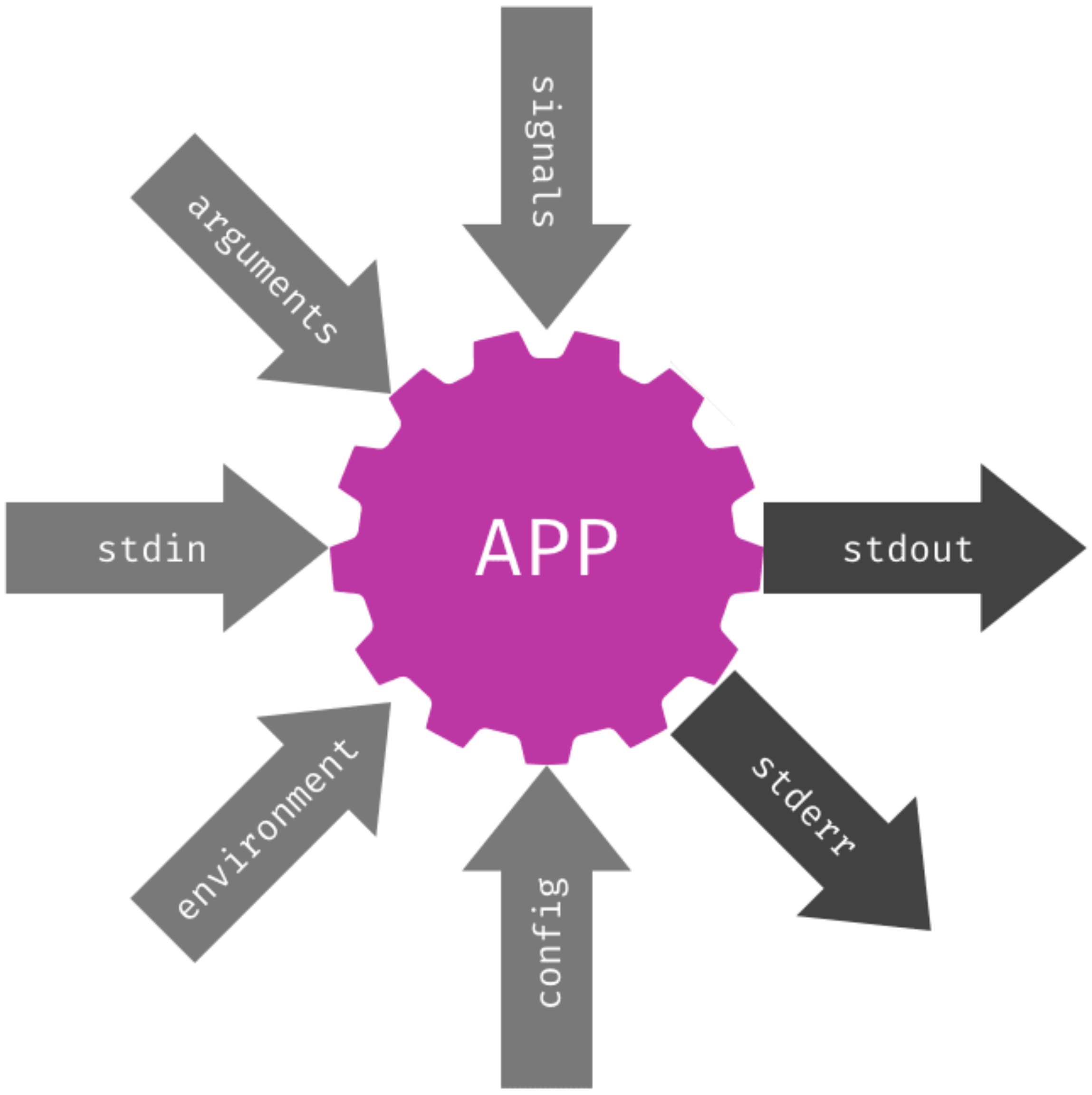
CONFIGPARSER

```
>>> import configparser
>>> config = configparser.ConfigParser()
>>> config.read('example.ini')
>>> config.sections()
['bitbucket.org', 'topsecret.server.com']
>>> config['bitbucket.org']['User']
'hg'
>>> config['DEFAULT']['Compression']
'yes'
>>> for key in config['bitbucket.org']:
...     print(key)
...
user
compressionlevel
compression
forwardx11
>>> config['bitbucket.org']['ForwardX11']
'yes'
>>> topsecret = config['topsecret.server.com']['ForwardX11']
```

CONFIGPARSER

```
>>> import configparser
>>> config = configparser.ConfigParser()
>>> config.read('example.ini')
>>> config.sections()
['bitbucket.org', 'topsecret.server.com']
>>> config['bitbucket.org']['User']
'hg'
>>> config['DEFAULT']['Compression']
'yes'
>>> for key in config['bitbucket.org']:
...     print(key)
...
user
compressionlevel
compression
forwardx11
>>> config['bitbucket.org']['ForwardX11']
'yes'
>>> topsecret = config['topsecret.server.com']['ForwardX11']
'no'
```


arguments
stdin
environment
config
APP
stdout
stderr



signals
arguments
stdin
APP
stdout
environment
config
stderr

UNIX SIGNALS

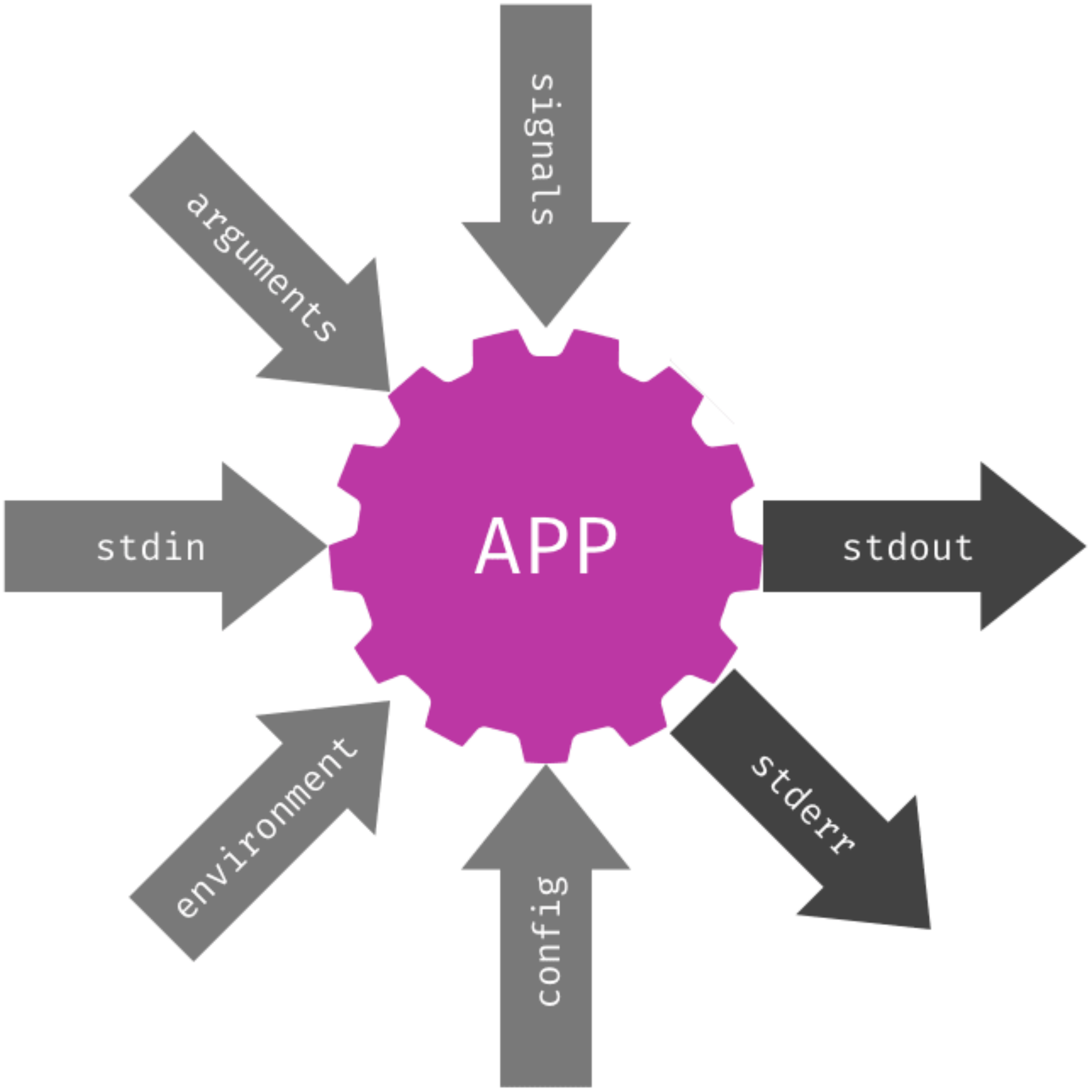
```
$
```

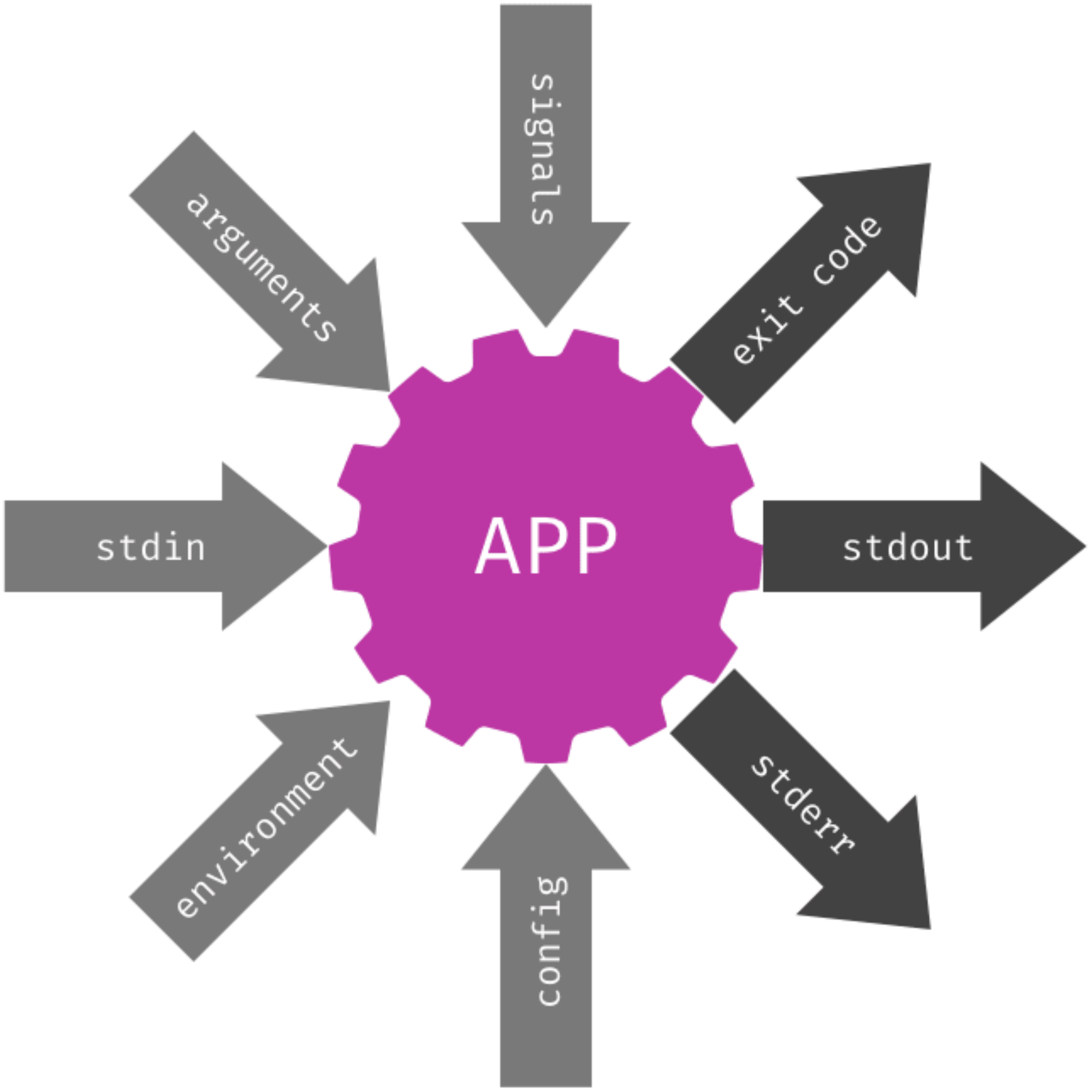
UNIX SIGNALS

```
$ kill -l
```

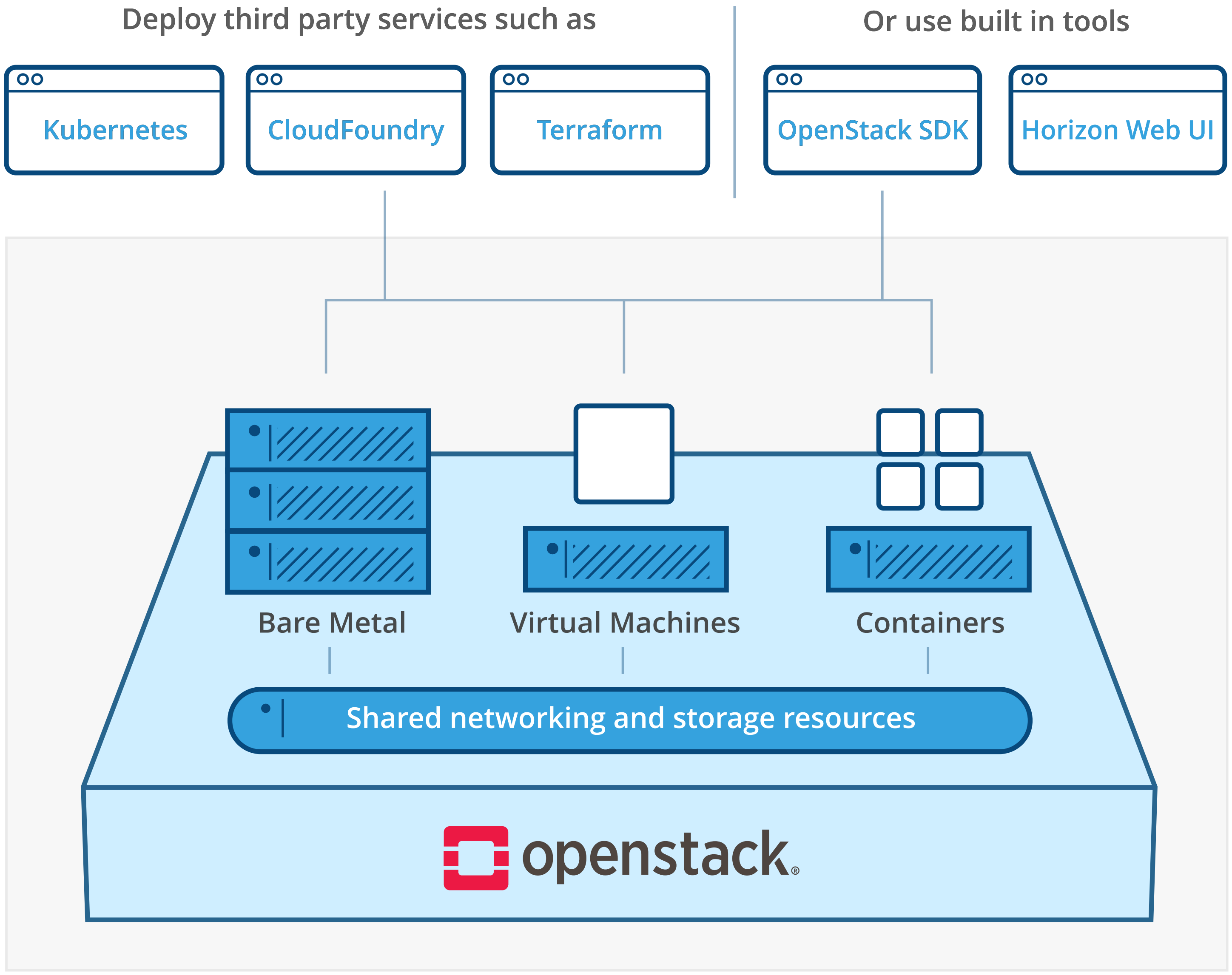
UNIX SIGNALS

```
$ kill -l
1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL
5) SIGTRAP      6) SIGABRT      7) SIGBUS       8) SIGFPE
9) SIGKILL      10) SIGUSR1     11) SIGSEGV     12) SIGUSR2
13) SIGPIPE     14) SIGALRM     15) SIGTERM     16) SIGSTKFLT
17) SIGCHLD     18) SIGCONT     19) SIGSTOP     20) SIGTSTP
21) SIGTTIN     22) SIGTTOU     23) SIGURG      24) SIGXCPU
25) SIGXFSZ     26) SIGVTALRM   27) SIGPROF     28) SIGWINCH
29) SIGIO       30) SIGPWR      31) SIGSYS      34) SIGRTMIN
35) SIGRTMIN+1  36) SIGRTMIN+2  37) SIGRTMIN+3  38) SIGRTMIN+4
39) SIGRTMIN+5  40) SIGRTMIN+6  41) SIGRTMIN+7  42) SIGRTMIN+8
43) SIGRTMIN+9  44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12
47) SIGRTMIN+13 48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14
51) SIGRTMAX-13 52) SIGRTMAX-12 53) SIGRTMAX-11 54) SIGRTMAX-10
55) SIGRTMAX-9  56) SIGRTMAX-8  57) SIGRTMAX-7  58) SIGRTMAX-6
59) SIGRTMAX-5  60) SIGRTMAX-4  61) SIGRTMAX-3  62) SIGRTMAX-2
63) SIGRTMAX-1  64) SIGRTMAX
$
```


signals
arguments
stdin
APP
stdout
environment
config
stderr


signals
arguments
exit code
stdin
APP
stdout
environment
stderr
config

openstack®


Deploy third party services such as
Kubernetes
CloudFoundry
Terraform
Or use built in tools
OpenStack SDK
Horizon Web UI
Bare Metal
Virtual Machines
Containers
Shared networking and storage resources
openstack

OSLO.CONFIG

OSLO.CONFIG

CONFIG

OSLO.CONFIG

ARGS + CONFIG

OSLO.CONFIG

ARGS + ENV + CONFIG

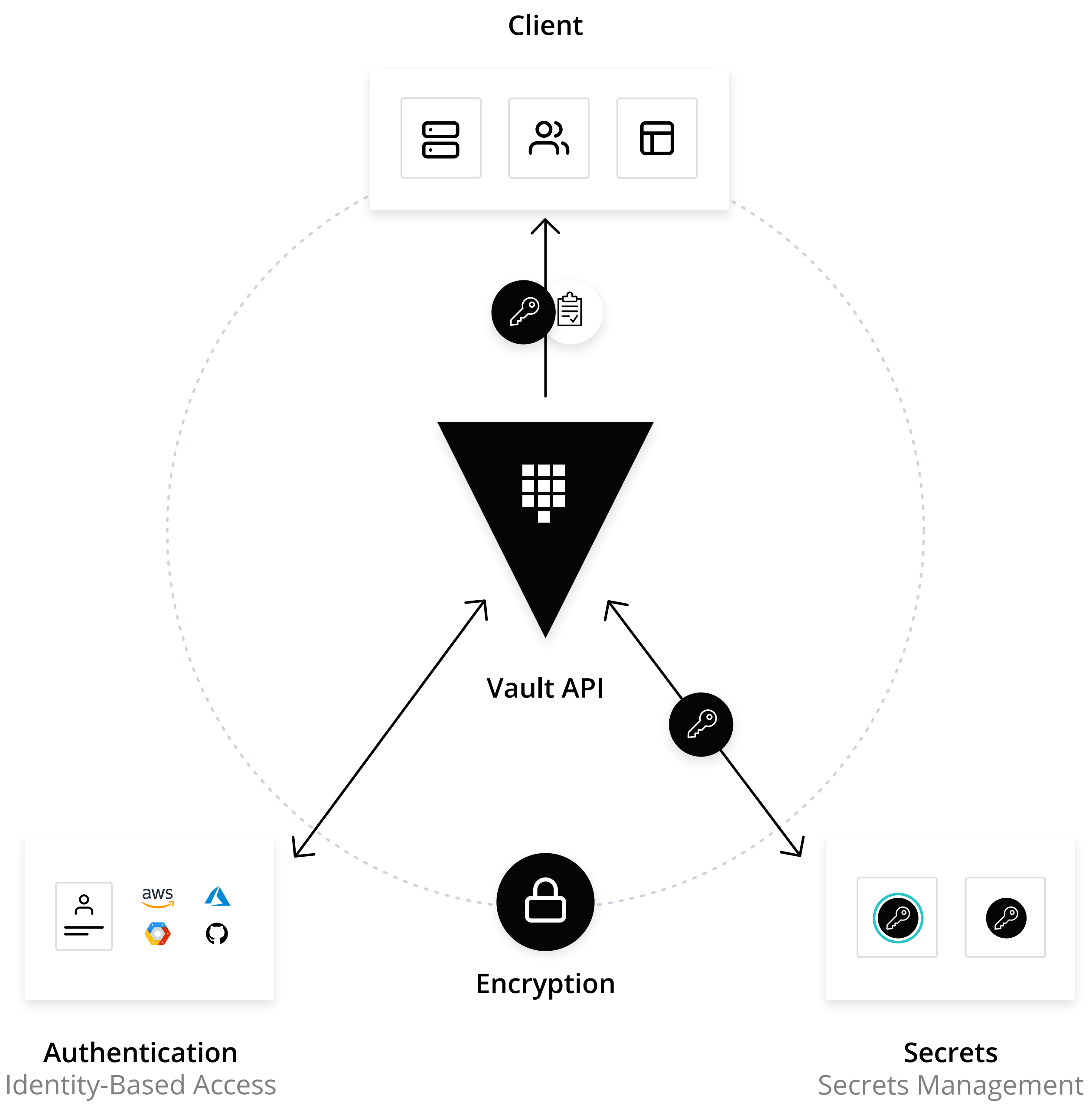
OSLO.CONFIG
ARGS + ENV + CONFIG + …

CASTELLAN

BARBICAN

HashiCorp

Vault


Client
Vault API
aws
Encryption
Authentication
Identity-Based Access
Secrets
Secrets Management

OSLO.CONFIG TYPES

String | Boolean | Integer | Float | List | Dict

OSLO.CONFIG TYPES

URI | Hostname | IPAddress | HostAddress | Port

OSLO.CONFIG

```python
#!/usr/bin/env python
from oslo_config import cfg

common_opts = [
    cfg.StrOpt('name',
               positional=True,
               default='world',
               help='Name to greet'),
]

greeting_opts = [
    cfg.StrOpt('greeting',
               default='Hello',
               help='Greeting to use.'),

    cfg.IntOpt('times',
               short='n',
               default=1,
               help='Times to greet.'),
]
```

OSLO.CONFIG

```python
def main():
    conf = cfg.ConfigOpts()

    conf.register_cli_opts(common_opts)

    conf.register_opt(greeting_opts[0], "greeting")
    conf.register_cli_opt(greeting_opts[1], "greeting")

    conf()

    for i in range(conf.greeting.times):
        print("{} {}!".format(
            conf.greeting.greeting,
            conf.name.capitalize()))


if __name__ == "__main__":
    main()
```

OSLO.CONFIG

```
$
```

OSLO.CONFIG

```
$ python hello.py --help
```

OSLO.CONFIG

```
$ python hello.py --help
usage: hello [-h] [--config-dir DIR] [--config-file PATH]
             [--greeting-times GREETING_TIMES]
             [name]

positional arguments:
  name                  Name to greet

optional arguments:
  -h, --help            show this help message and exit
  --config-dir DIR      Path to a config directory to pull `*.conf` files
                        ...
  --config-file PATH    Path to a config file to use. Multiple config fi
                        ...

greeting options:
  --greeting-times GREETING_TIMES, -n GREETING_TIMES
                        Times to greet.
```

OSLO.CONFIG

```
$
```

OSLO.CONFIG

```
$ python hello.py
```

OSLO.CONFIG

```
$ python hello.py
Hello World!
$
```

OSLO.CONFIG

```
$ python hello.py
Hello World!
$ python hello.py Brussels
```

OSLO.CONFIG

```
$ python hello.py
Hello World!
$ python hello.py Brussels
Hello Brussels!
$
```

OSLO.CONFIG

```
$ python hello.py
Hello World!
$ python hello.py Brussels
Hello Brussels!
$ python hello.py -n 3 Brussels
```

OSLO.CONFIG

```
$ python hello.py
Hello World!
$ python hello.py Brussels
Hello Brussels!
$ python hello.py -n 3 Brussels
Hello Brussels!
Hello Brussels!
Hello Brussels!
$
```

OSLO.CONFIG

```
$ python hello.py
Hello World!
$ python hello.py Brussels
Hello Brussels!
$ python hello.py -n 3 Brussels
Hello Brussels!
Hello Brussels!
Hello Brussels!
$ cat fosdem.conf
```

OSLO.CONFIG

```
$ python hello.py
Hello World!
$ python hello.py Brussels
Hello Brussels!
$ python hello.py -n 3 Brussels
Hello Brussels!
Hello Brussels!
Hello Brussels!
$ cat fosdem.conf
[DEFAULT]
name = Brussels

[greeting]
greeting = Hallo
times = 3
$
```

OSLO.CONFIG

```
$ python hello.py
Hello World!
$ python hello.py Brussels
Hello Brussels!
$ python hello.py -n 3 Brussels
Hello Brussels!
Hello Brussels!
Hello Brussels!
$ cat fosdem.conf
[DEFAULT]
name = Brussels

[greeting]
greeting = Hallo
times = 3
$ python hello.py --config-file fosdem.conf
```

OSLO.CONFIG

```
$ python hello.py
Hello World!
$ python hello.py Brussels
Hello Brussels!
$ python hello.py -n 3 Brussels
Hello Brussels!
Hello Brussels!
Hello Brussels!
$ cat fosdem.conf
[DEFAULT]
name = Brussels

[greeting]
greeting = Hallo
times = 3
$ python hello.py --config-file fosdem.conf
Hallo Brussels!
Hallo Brussels!
Hallo Brussels!
$
```

OSLO.CONFIG

```
$
```

OSLO.CONFIG

```
$ python hello.py --config-file fosdem.conf -n 30 "FOSDEM'20!"
```

OSLO.CONFIG

```
$ python hello.py --config-file fosdem.conf -n 10 "FOSDEM'20!"
Hallo FOSDEM'20!
Hallo FOSDEM'20!
Hallo FOSDEM'20!
Hallo FOSDEM'20!
Hallo FOSDEM'20!
Hallo FOSDEM'20!
Hallo FOSDEM'20!
Hallo FOSDEM'20!
Hallo FOSDEM'20!
Hallo FOSDEM'20!
$
```

OSLO.CONFIG

```
$
```

OSLO.CONFIG

```
$ OS_GREETING__TIMES=30 python hello.py
```

OSLO.CONFIG

```
$ OS_GREETING__TIMES=10 python hello.py
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
$
```

EXTRA CONFIG SOURCES

```
# app.conf

[app]
hostname = 0.0.0.0
port = 5000

[db]
hostname = "localhost"
username = "postgres"
password = "changeme"
```

```
$ python app.py --config-file=app.py
 * Serving Flask app "app" (lazy loading)
 * Environment: production
 * Debug mode: off
 * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
```

EXTRA CONFIG SOURCES

```
# app.conf

[app]
hostname = 0.0.0.0        <--- default
port = 5000               <--- default

[db]
hostname = "localhost"  <--- default
username = "postgres"
password = "changeme"

$ python app.py --config-file=app.py
 * Serving Flask app "app" (lazy loading)
 * Environment: production
 * Debug mode: off
 * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
```

EXTRA CONFIG SOURCES

```
# app.conf

[db]
username = "postgres"
password = "changeme"

$ python app.py --config-file=app.py
 * Serving Flask app "app" (lazy loading)
 * Environment: production
 * Debug mode: off
 * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
```

EXTRA CONFIG SOURCES

```
# app.conf

#
#
#

$ OS_DB__USERNAME=postgres OS_DB_PASSWORD=changeme python app.py
 * Serving Flask app "app" (lazy loading)
 * Environment: production
 * Debug mode: off
 * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
```

EXTRA CONFIG SOURCES

```
# app.conf

[DEFAULT]
config_source = "secrets"

[secrets]
driver=castellan
config_file = "castellan.conf"
mapping_file = "mapping.conf"

# castellan.conf

[key_manager]
backend=vault

# mapping.conf

[db]
username = "3846d164d5a146eb9c27695637bec4b3"
password = "c2aa215115594fba8549e22dfb52e82a"
```

EXTRA CONFIG SOURCES

```
$ OS_VAULT__ROOT_TOKEN_ID=s.FSUHTpovzLBhu4ENA5evXWdC \
      python app.py --config-file=app.py
 * Serving Flask app "app" (lazy loading)
 * Environment: production
 * Debug mode: off
 * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
```

THANK YOU!

GUIMARAES+TALKS@PM.ME

HTTPS://MOISESGUIMARAES.COM/TALKS/