# Execution abstraction layer for high-level system scripts

# Develop to a session

```
def greetings(session):
    """Write 'Hello world!' in 'greetings.txt' file relative to user's home."""
    home = session.user.home
    file_path = session.file.join(home, u'greetings.txt')
    file_resource = session.file(file_path)
    with file_resource.open('w'):
        file_resource.write(u'Hello world!')
```

## Run it anywhere

### Python shell

```
>>> from europython import greetings
>>> import xal
>>> session = xal.LocalSession()
>>> greetings(session)
```

#### zc.buildout

```
from europython import greetings
import xal

class HelloBuildout(object):
    def __init__(self, buildout, name, options):
        self.session = xal.BuildoutSession(buildout, name, options)

def install(self):
    greetings(self.session)
```

#### Fabric

```
from europython import greetings
import xal

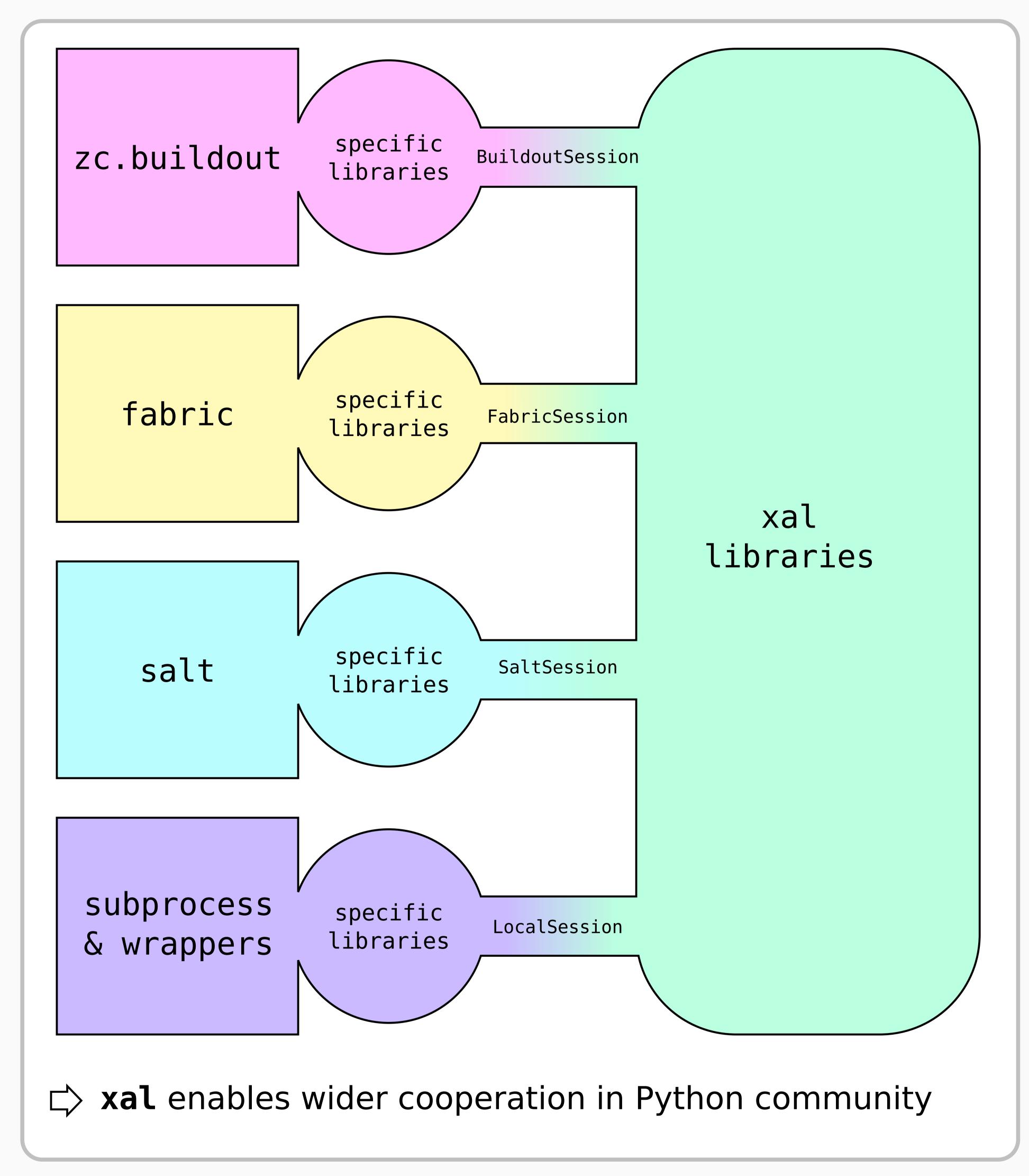
def hello_fabric():
    session = xal.FabricSession(sudoer=True)
    greetings(session)
```

#### Salt

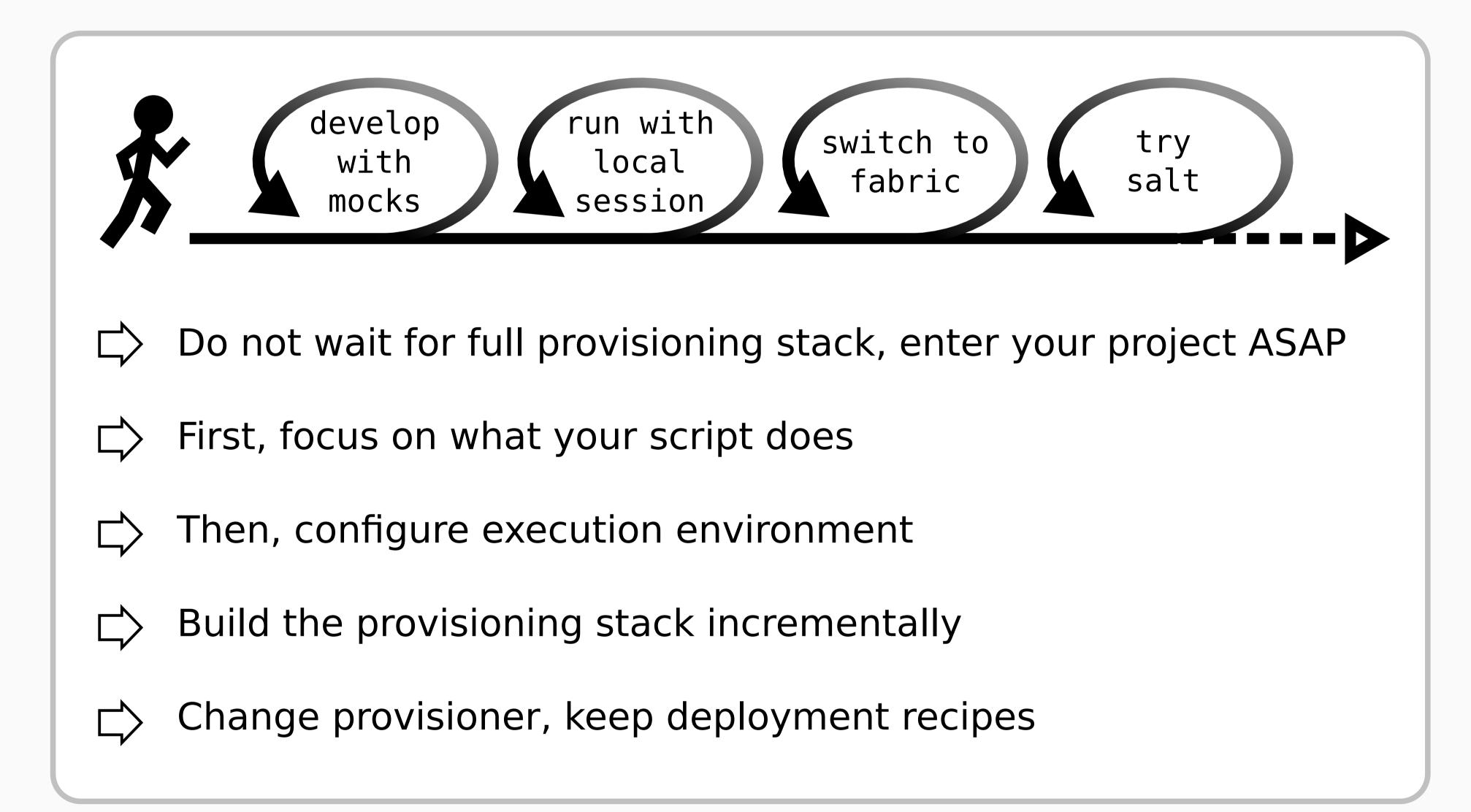
```
from europython import greetings
import xal

def hello_salt():
    session = xal.SaltSession(__salt__)
    greetings(session)
```

## Share libraries



## Improve your workflow



# Challenges

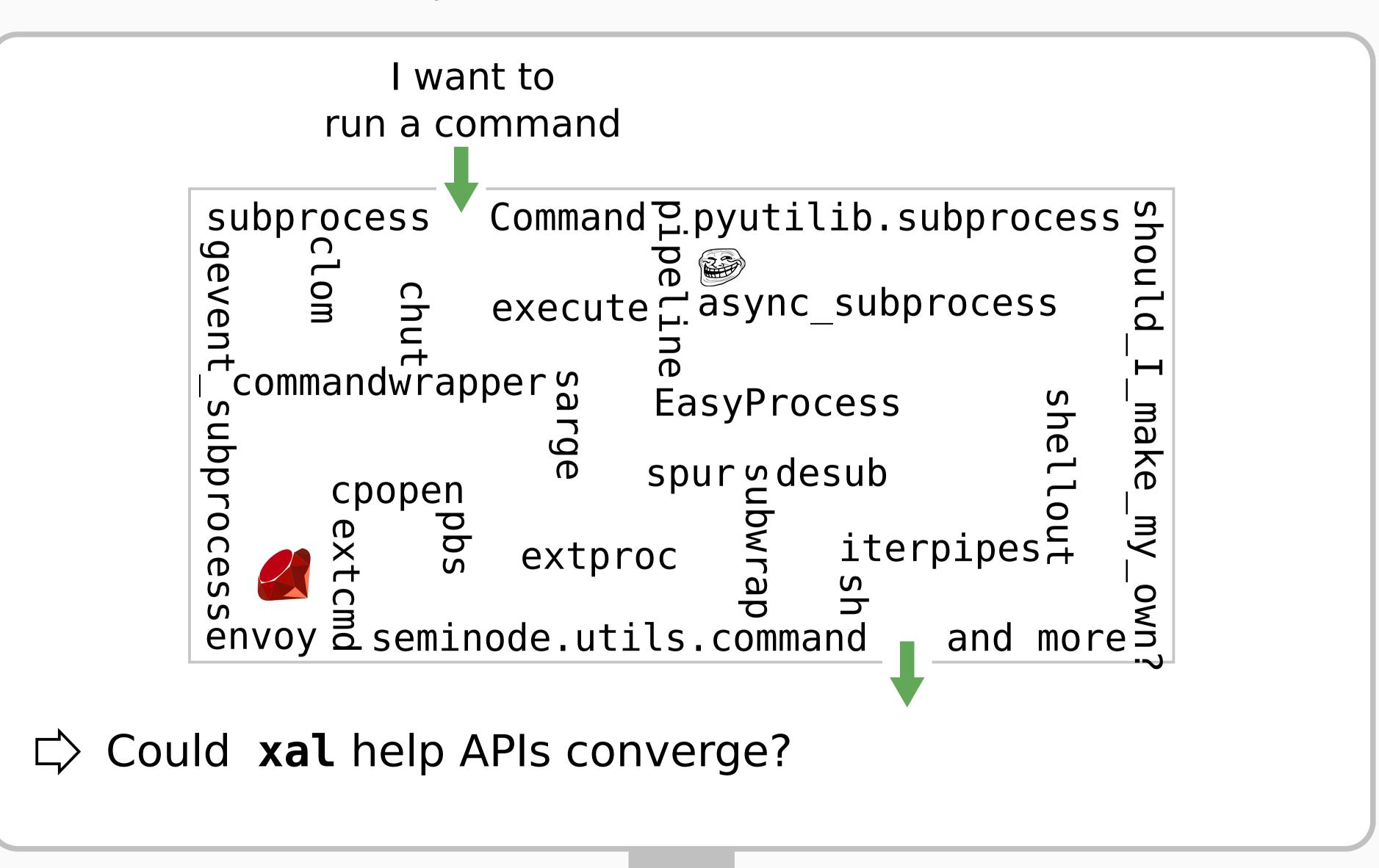
xal needs good APIs:

 run commands (subprocess wrapper)
 consistent set of resources (files, users...)

 Efficient and comprehensive session registry
 Preconfigured registries: fabric, salt, buildout, local session...
 Resolution of session's dependencies
 Smart handling of NotImplementedError

1> Unit tests with mocks, integration tests

## Exit the subprocess labyrinth



xal is a proof of concept https://xal.readthedocs.org