Remote task management using Saltstack

Piotr Ćwiek, Marcin Baryłka
Samsung Electronics Polska

v.1.3

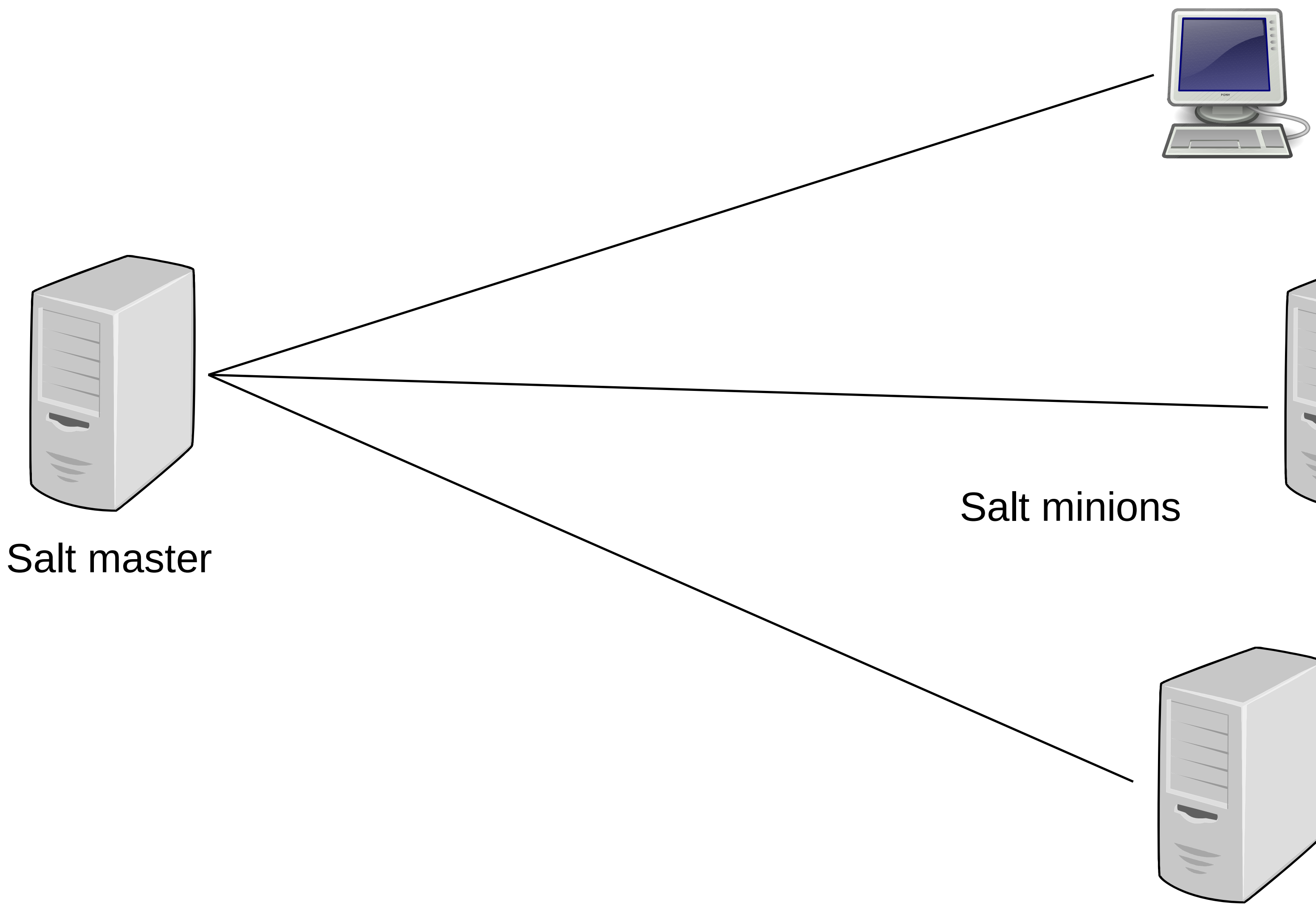
The story

- We have a lot of servers, doing a lot of things
- All servers can be groupped by their functionality
- We want to monitor and manage them
- Users should have access as an administrator or a watcher to see server statistics
- The system should be easily accessible via the web
- Adding machines to the app should be easy
- We wanted to do some research, after all we are **R&D.** :-)

Saltstack was the answer *).

*) your results may vary

Saltstack architecture overview



Remote execution

```
$ sudo salt \* cmd.run 'apt-get install tree'
```

Remote execution

```
host01:
    Reading package lists...
    Building dependency tree...
    Reading state information...
    The following packages were automatically installed and are no longer required:
      git git-man libjs-jquery python-async python-git python-gitdb python-smmap
    Use 'apt-get autoremove' to remove them.
    The following NEW packages will be installed:
      tree
    0 upgraded, 1 newly installed, 0 to remove and 62 not upgraded.
    Need to get 37.8 kB of archives.
    After this operation, 109 kB of additional disk space will be used.
    Get:1 http://us.archive.ubuntu.com/ubuntu/ trusty/universe tree amd64 1.6.0-1 [37.8 kB]
    Fetched 37.8 kB in 0s (0 B/s)
    Selecting previously unselected package tree.
    (Reading database ... 66298 files and directories currently installed.)
    Preparing to unpack .../tree_1.6.0-1_amd64.deb ...
    Unpacking tree (1.6.0-1) ...
    Processing triggers for man-db (2.6.7.1-1) ...
    Setting up tree (1.6.0-1) ...
host02:
    Reading package lists...
```

Execution modules

```
$ sudo salt \* pkg.install tree
host02:
    ----------
    tree:
        ----------
        new:
            1.6.0-1
        old:

host01:
    ----------
```

ubuntu

python™

PostgreSQL

django

Whisper, Carbon

jQuery
write less, do more.

SALTSTACK

Monitoring

- Custom Saltstack module
  - Collects statistics every 15 secs
  - Saves them to Graphite (a kind of round-robin db, a part of Carbon software)

Peter Baumgartner / LincolnLoop has made a presentation covering this subject:

https://speakerdeck.com/ipmb/monitoring-infrastructure-with-saltstack

Custom execution modules (1)

Create file “mypkg.py”:

```
def myinstall(name):
    return __salt__['pkg.install'](name)
```

Custom execution modules (2)

Place it:

- on salt master host

  or
- in a git repo.

Custom execution modules (3)

Make all minions download it:

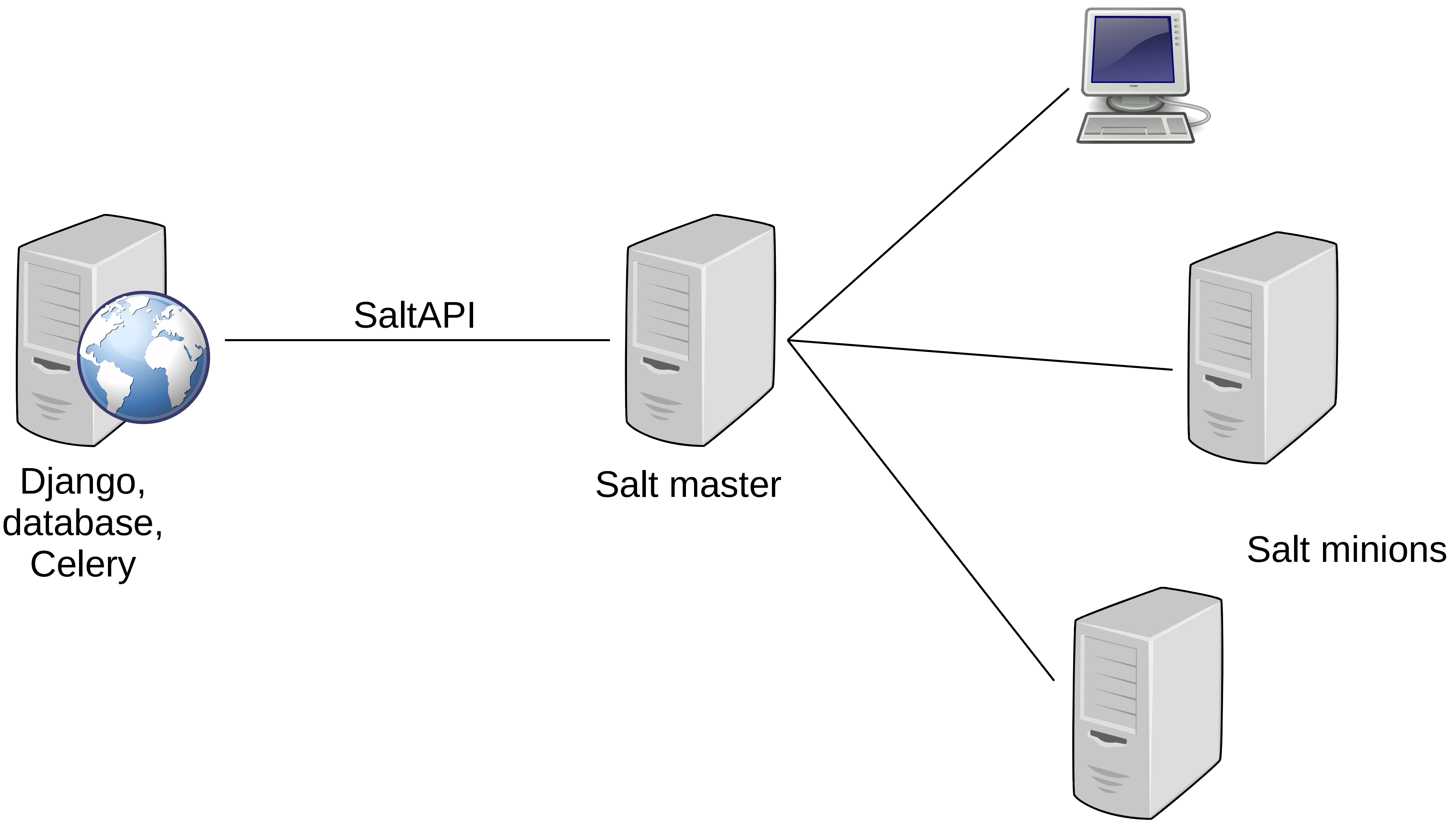
```
$ sudo salt \* saltutil.sync_modules
```

Custom execution modules (4)

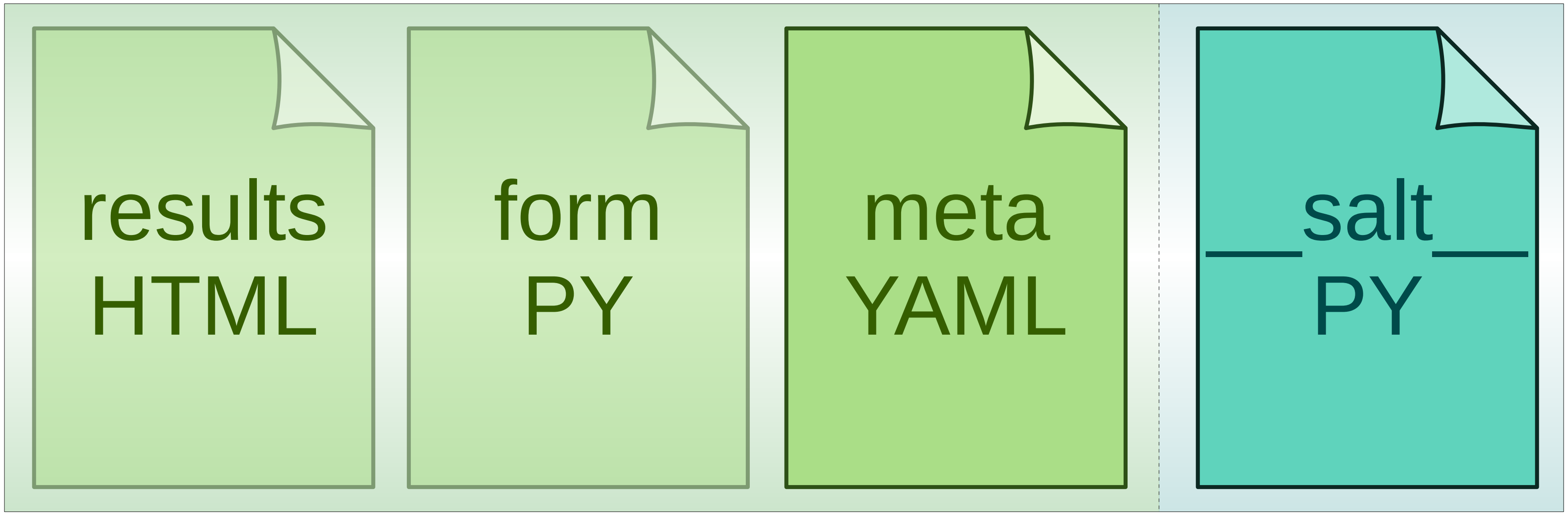
Use it:

```
$ sudo salt \* mypkg.myinstall tree
```

Architecture overview



Plugin structure



Plugin execution modules

__salt__
PY

- Enhance standard functionality.
- Conform to a naming scheme.
- Have common interface:
  - JSON parameters,
  - plugin version,
  - plugin configuration.
- Return well-defined values:
  - status (OK, warning, error, …),
  - any additional result data.

Enable plugin in Django

meta
YAML

The meta YAML file is required but can be empty:

- title,
- description,
- enabled / disabled / debug,
- version,
- plugin-specific configuration.

Provide parameters form

(optional)

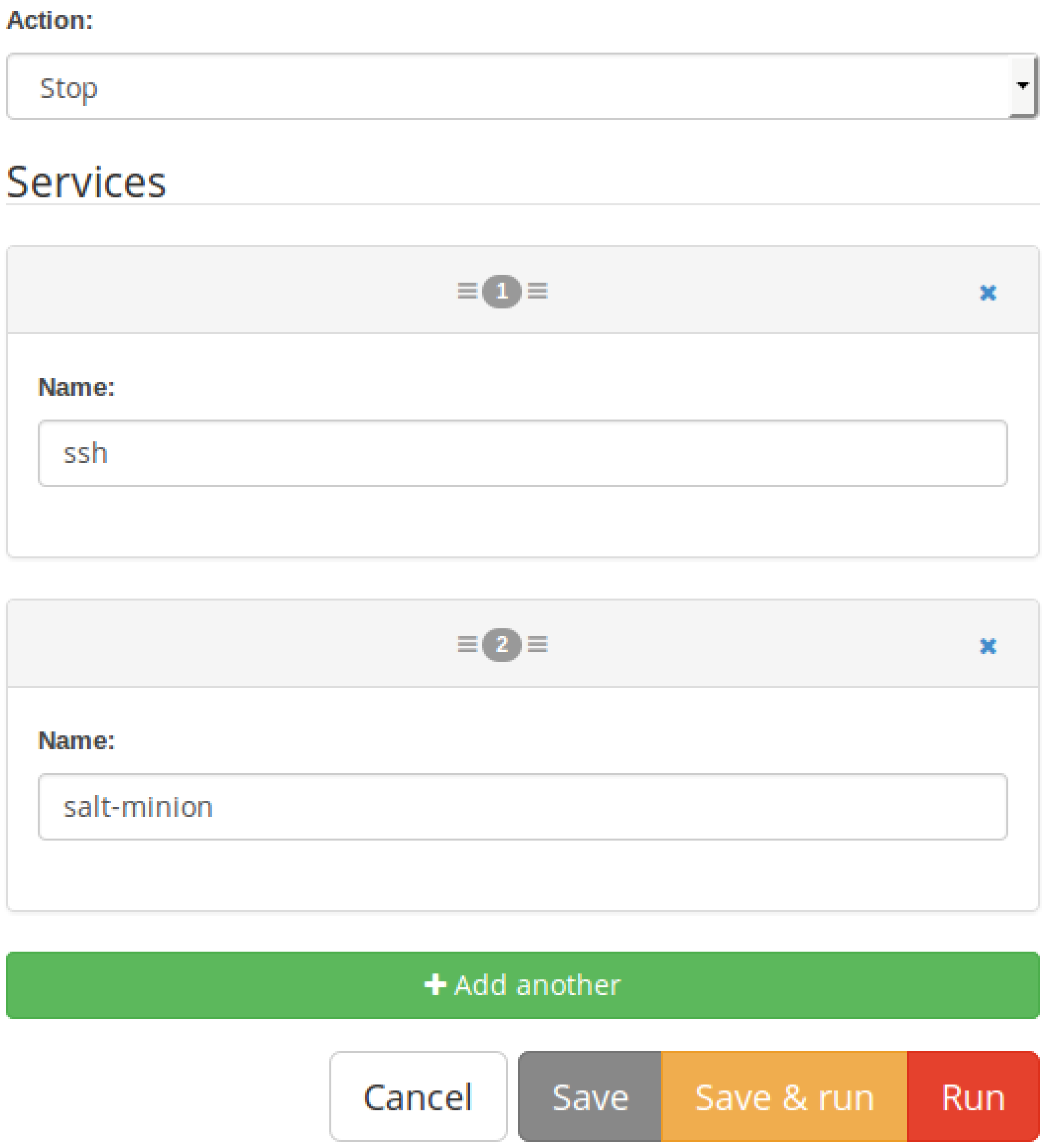
form
PY

```
class _ActionForm(Form):
    action = ChoiceField(choices=SERVICE_ACTIONS)


class _ServiceForm(Form):
    name = CharField()


_ServiceFormSet = formset_factory(_ServiceForm)


class PluginForm(CompoundForm):
    action = SubForm(_ActionForm)
    services = SubForm(_ServiceFormSet, title="Services")
```


Action: Stop

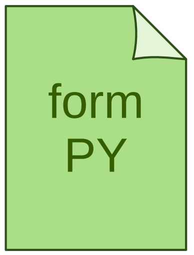
Services

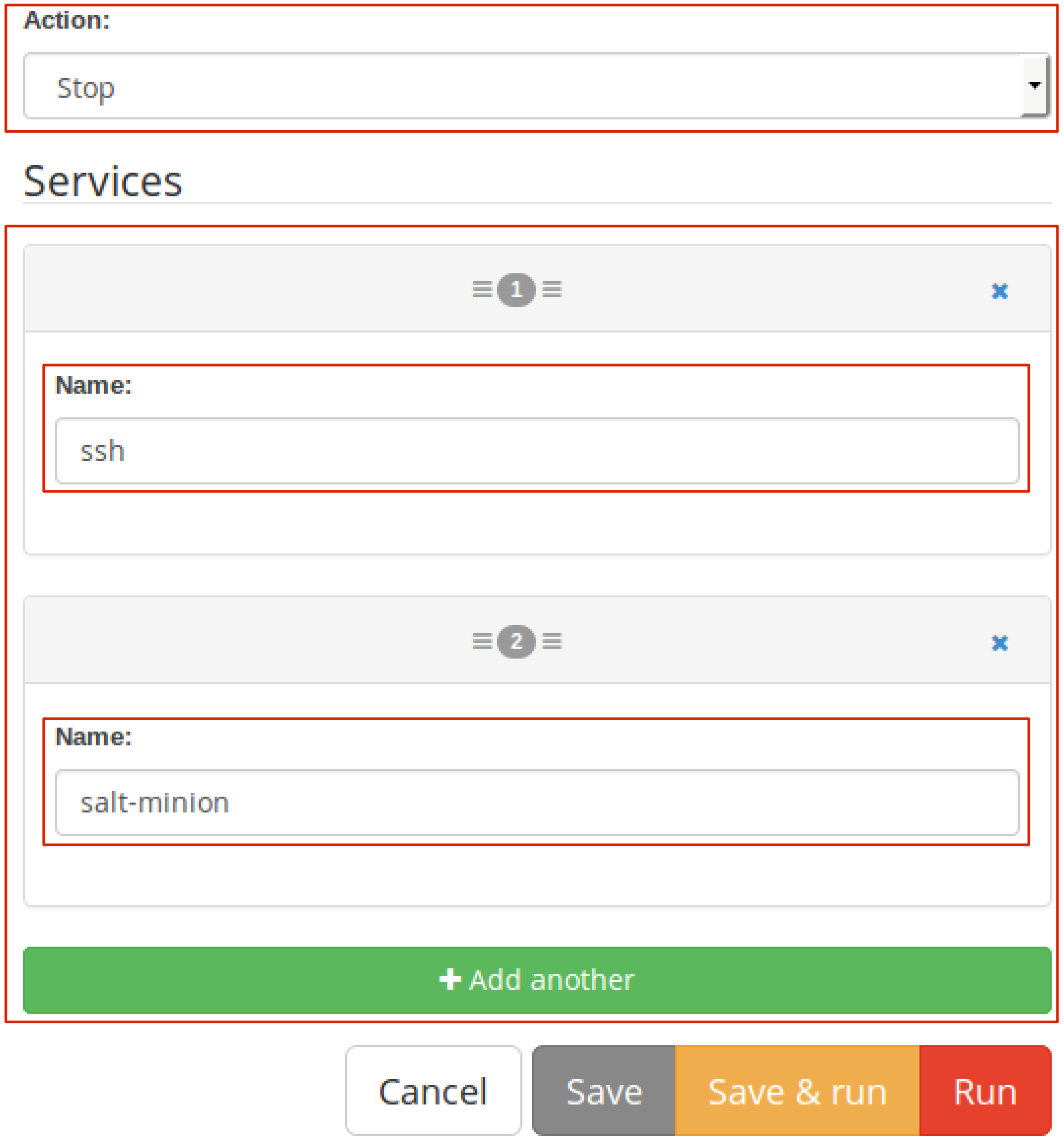
1

Name: ssh

2

Name: salt-minion





Action:

Stop

Services

1

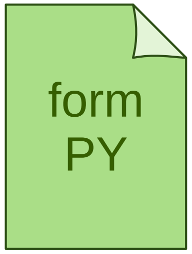
Name:

ssh

2

Name:

salt-minion



form
PY

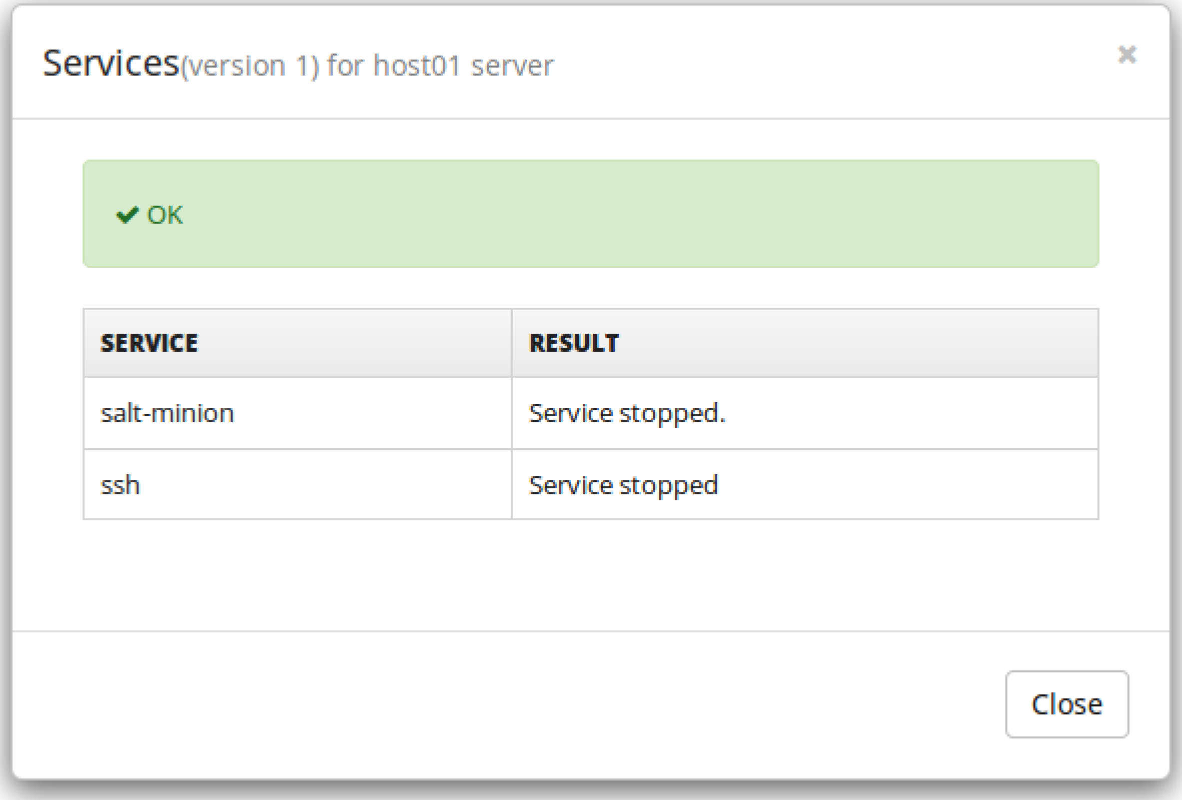
```
{
    "action": {
        "action": "stop"
    },
    "services": [
        {"name": "ssh"},
        {"name": "salt-minion"}
    ]
}
```

Provide results template (optional)

results
HTML

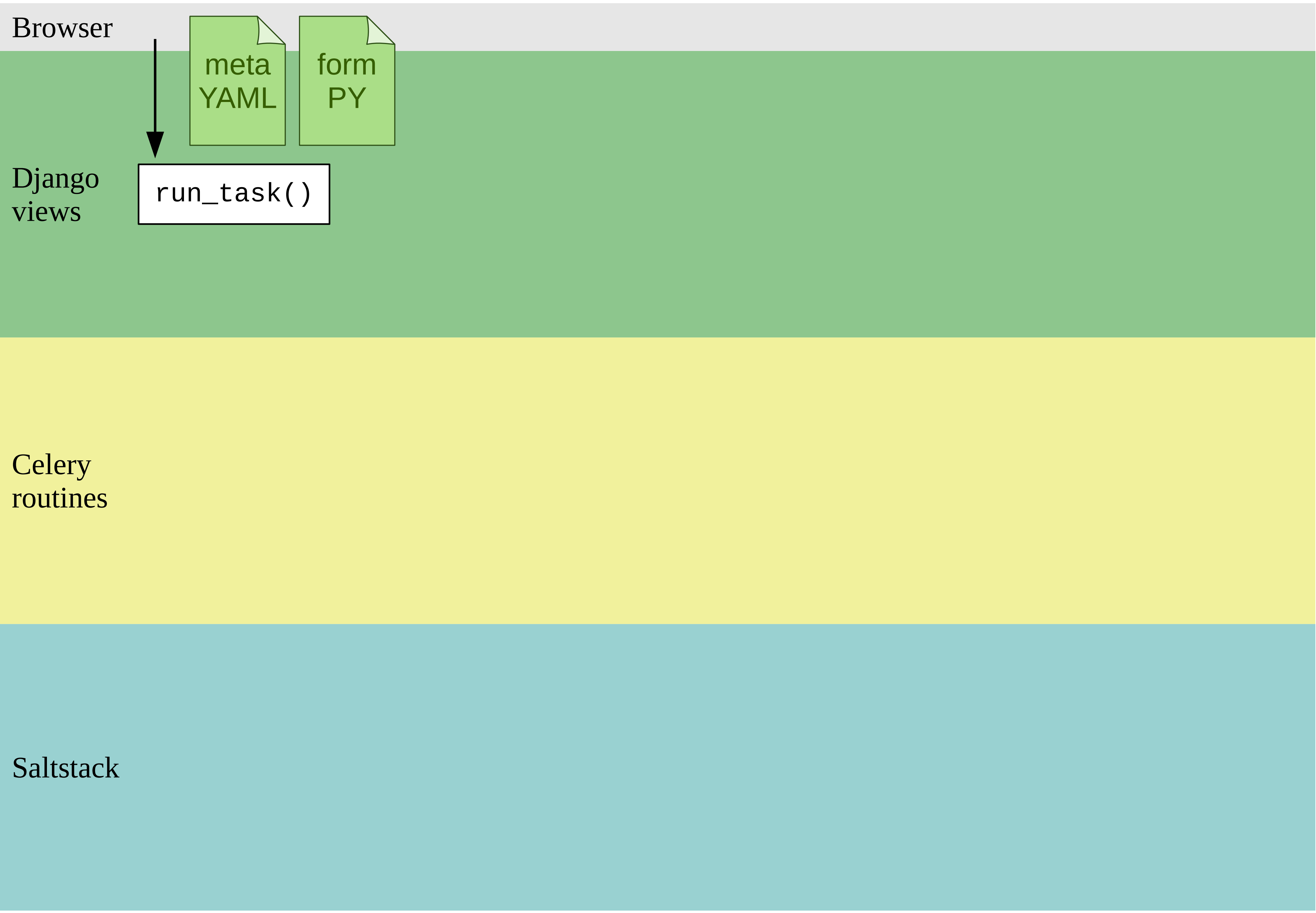
- Receives plugin results (parsed JSON).
- Visualizes them in any way.
- Generated HTML is embedded on the page.

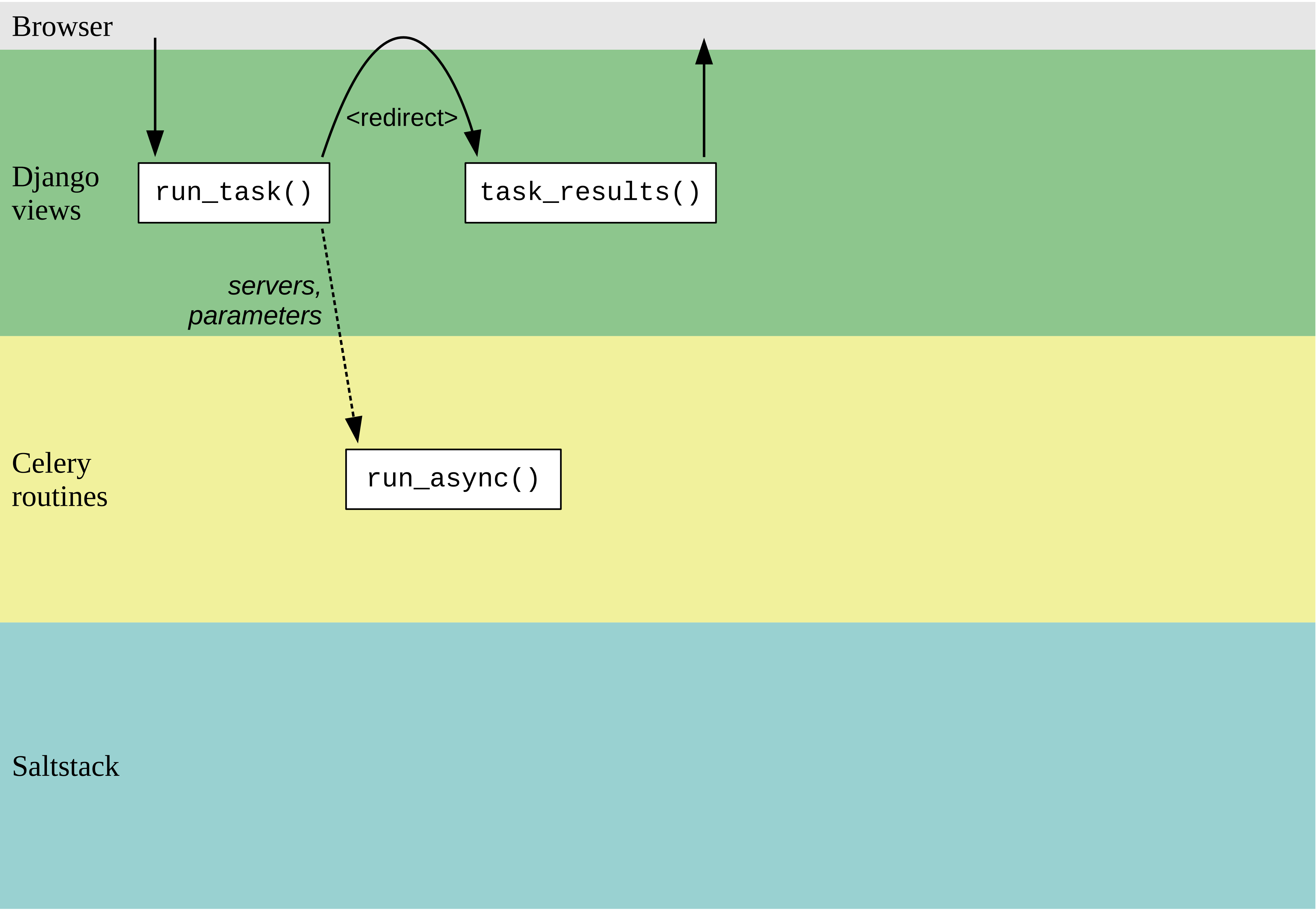
results HTML

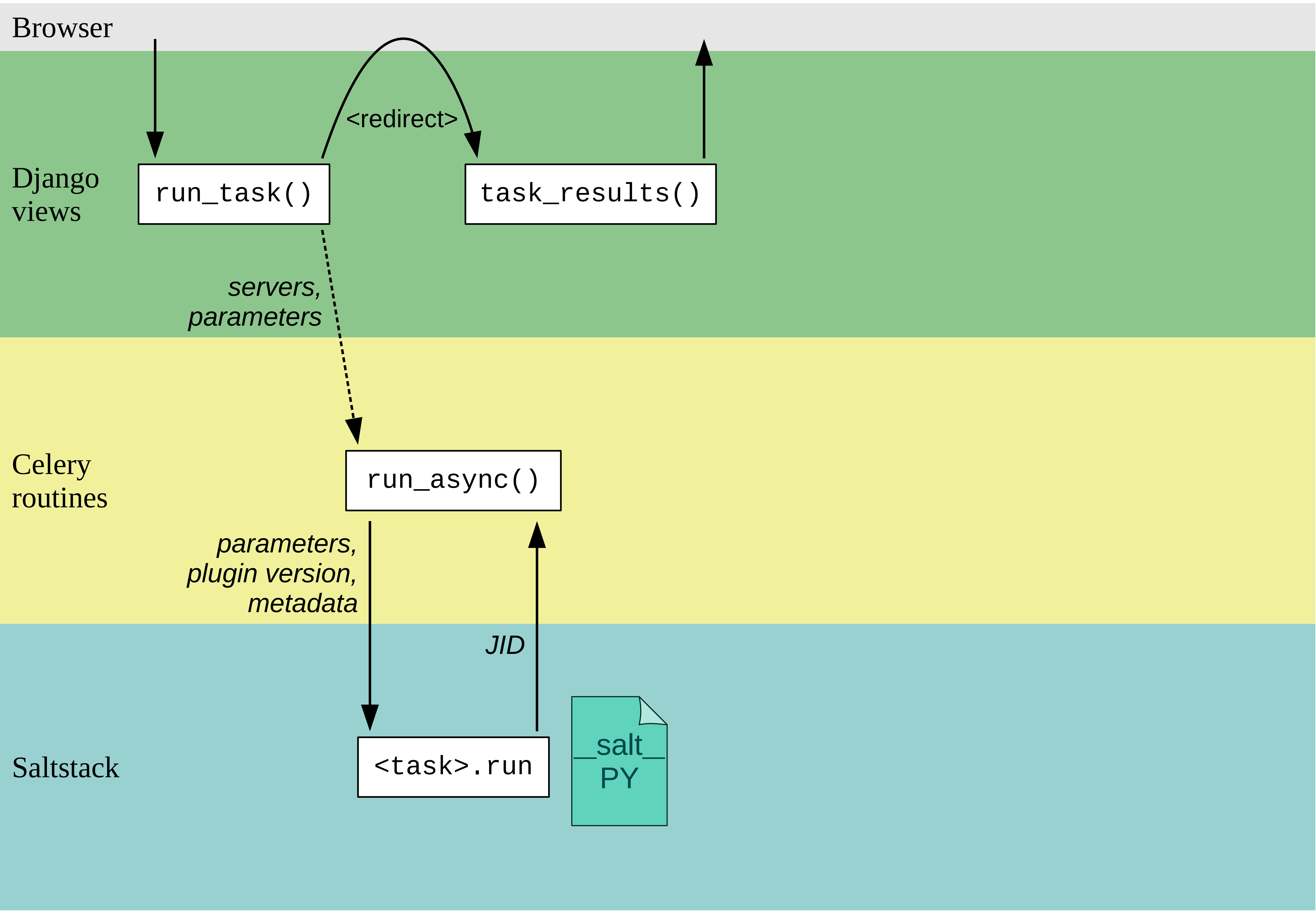

Services(version 1) for host01 server

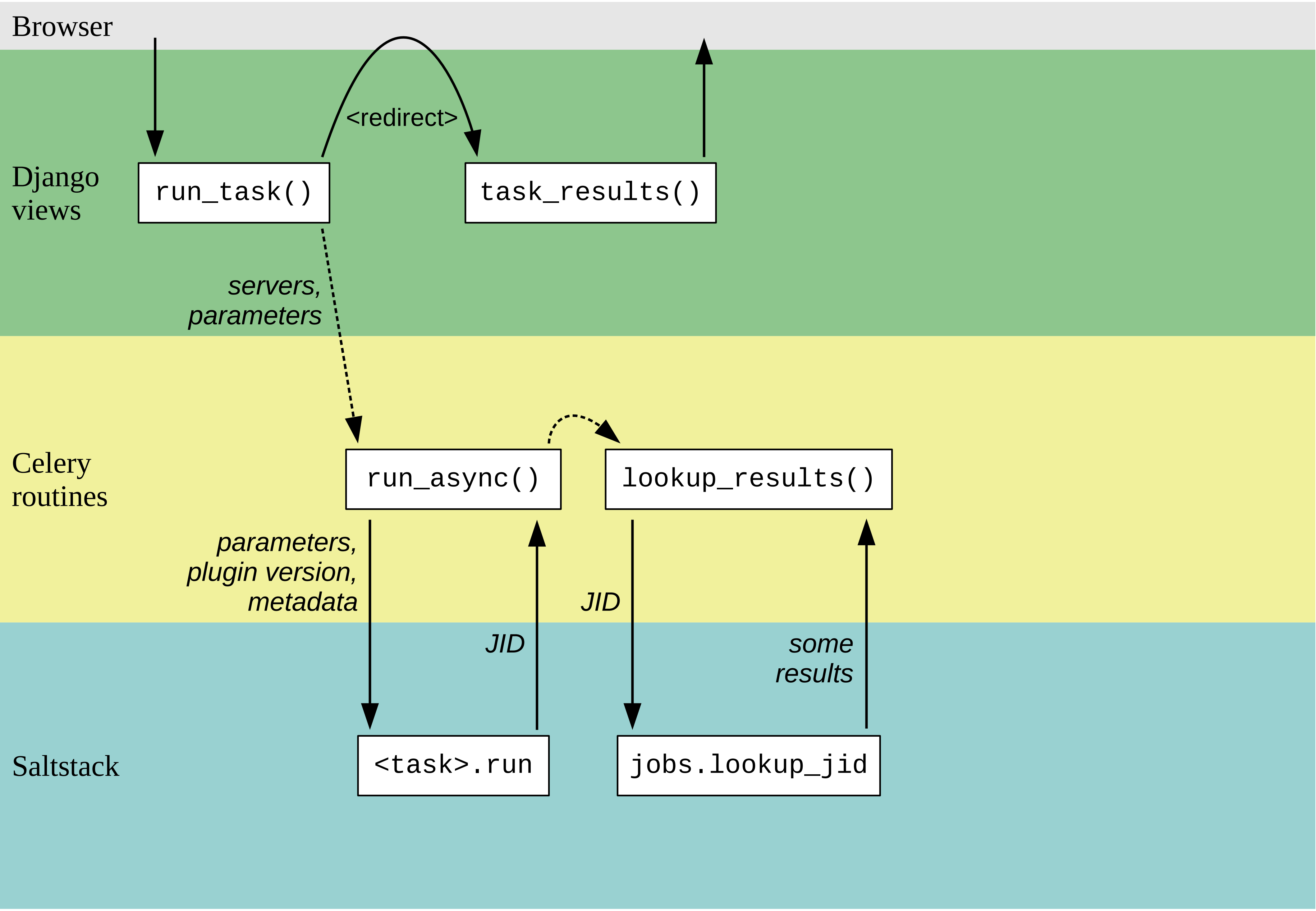
✓ OK

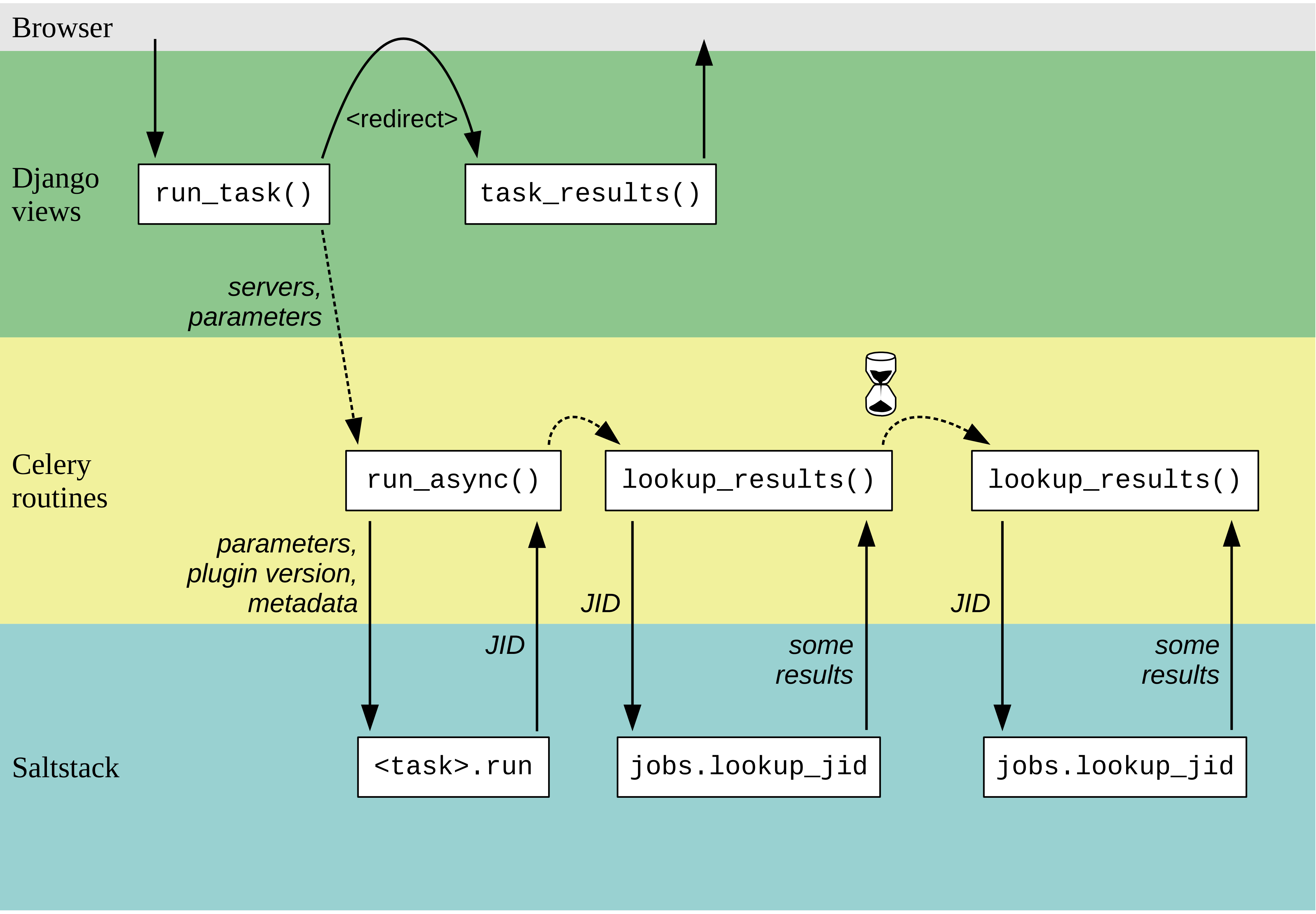
SERVICE	RESULT
salt-minion	Service stopped.
ssh	Service stopped

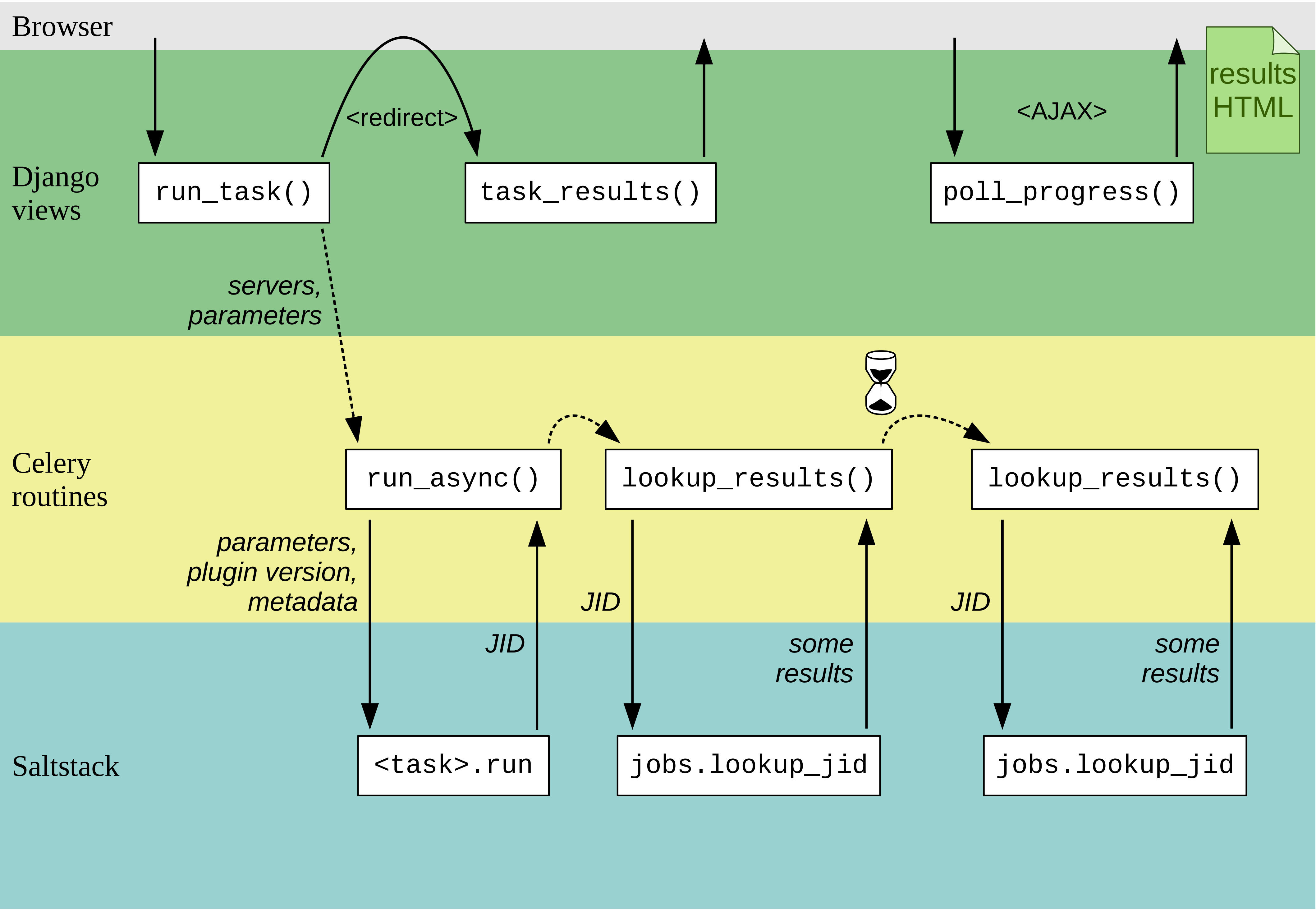

Browser
meta YAML
form PY
Django views
run_task()
Celery routines
Saltstack


Browser
Django views
run_task()
<redirect>
task_results()
servers, parameters
Celery routines
run_async()
Saltstack


Browser
Django views
run_task()
<redirect>
task_results()
servers, parameters
Celery routines
run_async()
parameters, plugin version, metadata
JID
Saltstack
<task>.run
__salt__
PY


Browser
Django views
Celery routines
Saltstack
run_task()
task_results()
<redirect>
servers, parameters
run_async()
lookup_results()
parameters, plugin version, metadata
JID
JID
some results
<task>.run
jobs.lookup_jid


Browser
Django views
Celery routines
Saltstack
run_task()
<redirect>
task_results()
servers, parameters
run_async()
lookup_results()
lookup_results()
parameters, plugin version, metadata
JID
JID
some results
JID
some results
<task>.run
jobs.lookup_jid
jobs.lookup_jid


Browser
Django views
Celery routines
Saltstack
run_task()
<redirect>
task_results()
<AJAX>
poll_progress()
results HTML
servers, parameters
run_async()
lookup_results()
lookup_results()
parameters, plugin version, metadata
JID
JID
some results
JID
some results
<task>.run
jobs.lookup_jid
jobs.lookup_jid

Known problems

- Tasks: if the value returned by the task can't be serialized to JSON, the master doesn't receive data, and logs stay silent.
- Celery + > 1 database = data integrity loss
- Saltstack master job cache: running out of I-nodes (yes, we've done it!)

Questions?