A Jobs Queue for processing tasks asynchronously

Guewen Baconnier & Leonardo Pistone

Camptocamp

About us

Guewen Baconnier

- Developer @ Camptocamp
- OCA committer, OCA Delegate
- Connector author
- @guewenb
- @guewen

Leonardo Pistone

- Developer @ Camptocamp
- OCA committer, OCA Delegate
- @lepistone

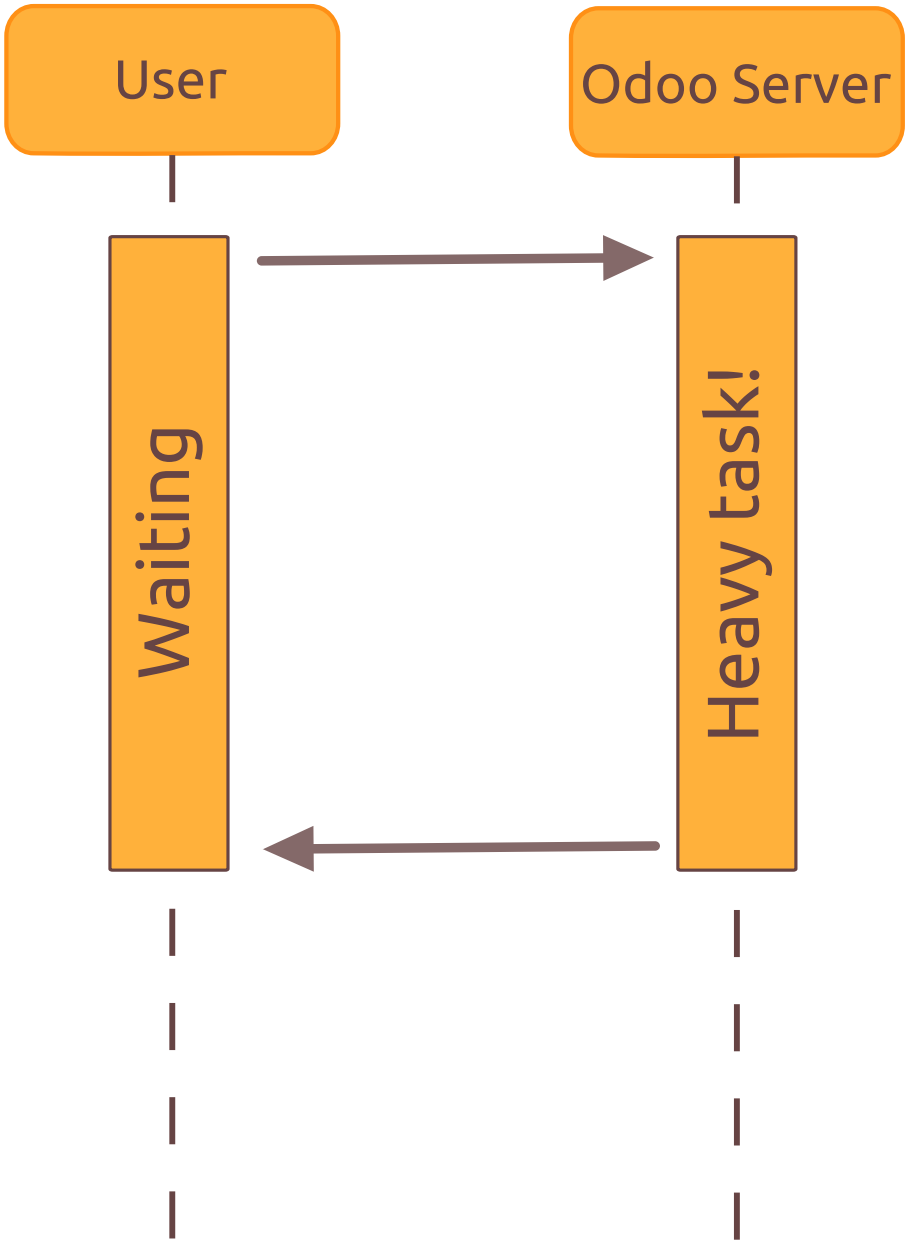
camptocamp

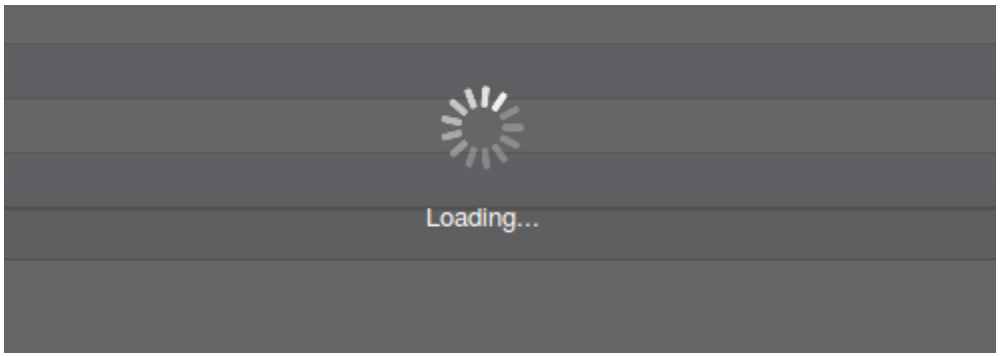
Computers are slow!

Computers are slow!

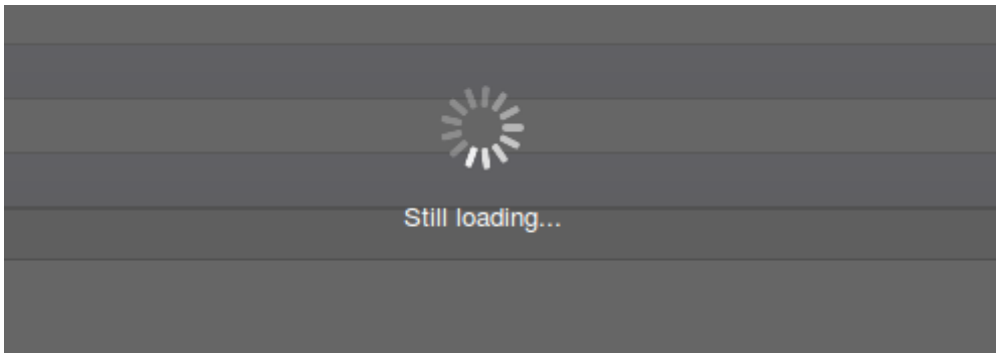
Humans want them to be fast!

The problem

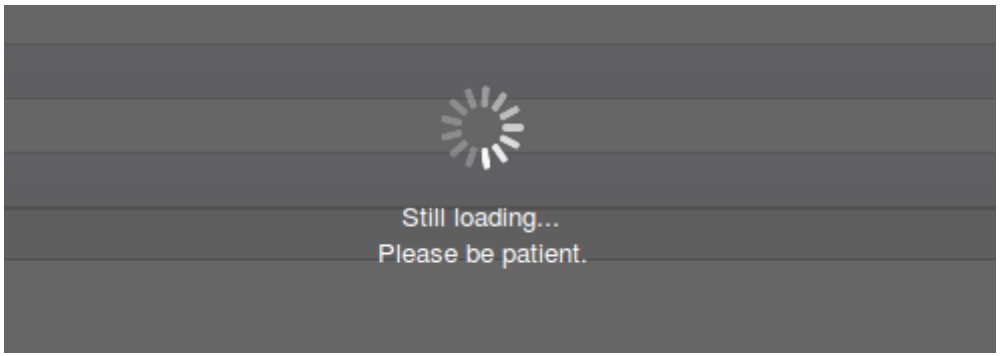

User
Odoo Server
Waiting
Heavy task!



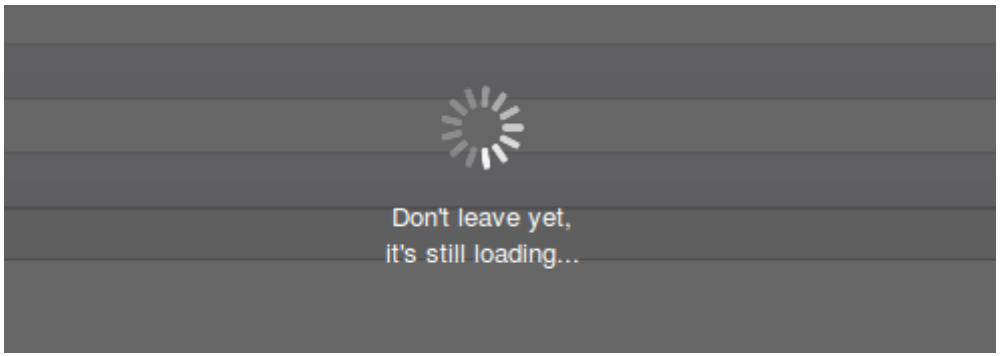
Loading...



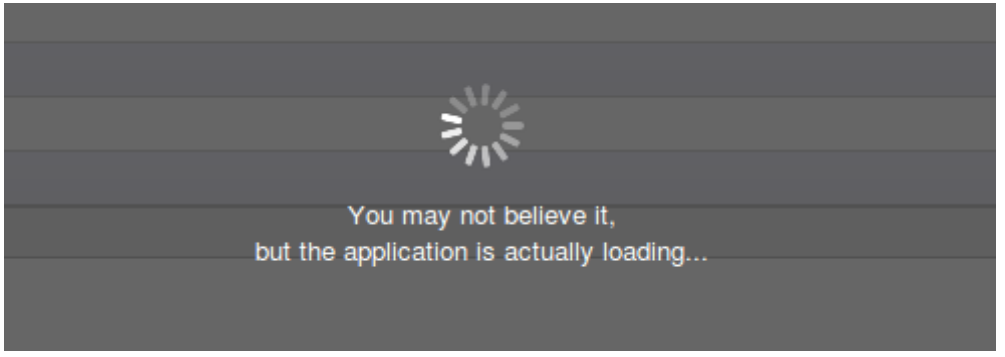
Still loading...



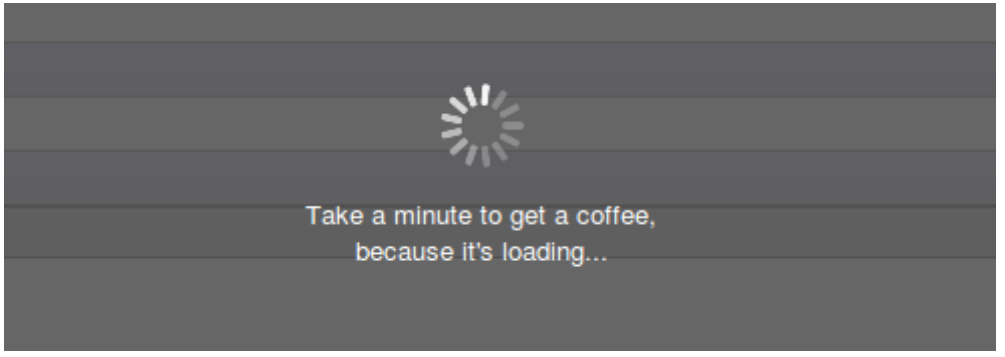
Still loading... Please be patient.



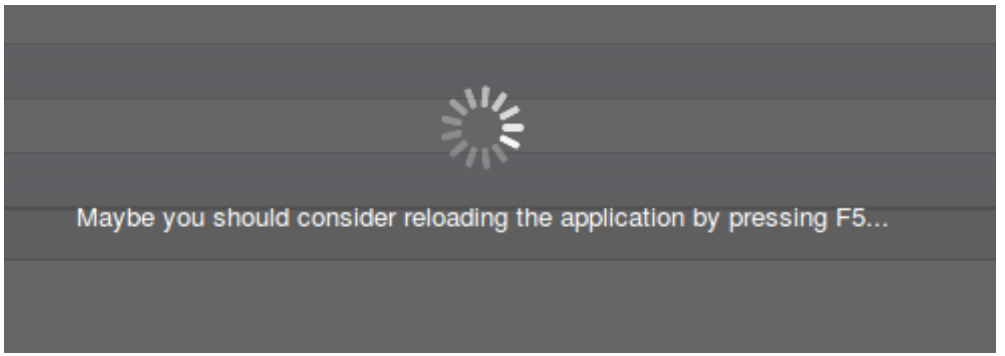
Don't leave yet, it's still loading



You may not believe it, but the application is actually loading...



Take a minute to get a coffee, because it's loading...

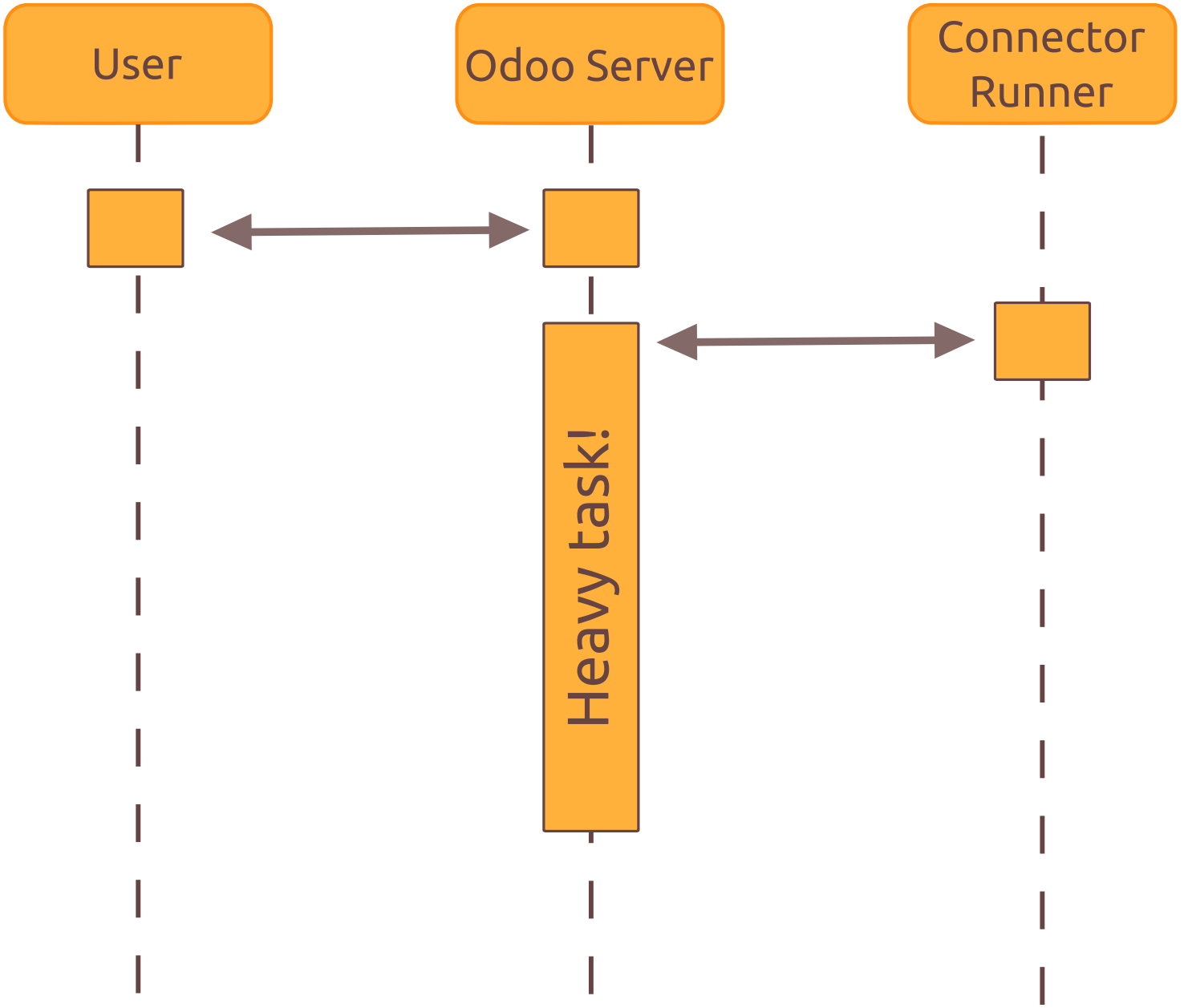

Maybe you should consider reloading the application by pressing F5...

Come on...

We can try to save a few seconds

We can try to save a few seconds

But we have more radical solutions


User
Odoo Server
Connector Runner
Heavy task!

Connector

odoo-connector.com

Queue it!

Dependency on connector

Queue it!

Dependency on connector

Declare a job:

```
from openerp.addons.connector.queue.job import job

@job
def a_heavy_task(session, model_name, record_id):
    # do an heavy task on record_id of model_name
```

Queue it!

Dependency on connector

Declare a job:

```
from openerp.addons.connector.queue.job import job

@job
def a_heavy_task(session, model_name, record_id):
    # do an heavy task on record_id of model_name
```

Delay a job:

```
session = ConnectorSession.from_env(self.env)
a_heavy_task.delay(session, 'res.partner', 1)
```

Dequeue it!

Start the server with:

```
ODOO_CONNECTOR_CHANNELS=root:2 ./openerp-server --load=web,connector --workers=4
```

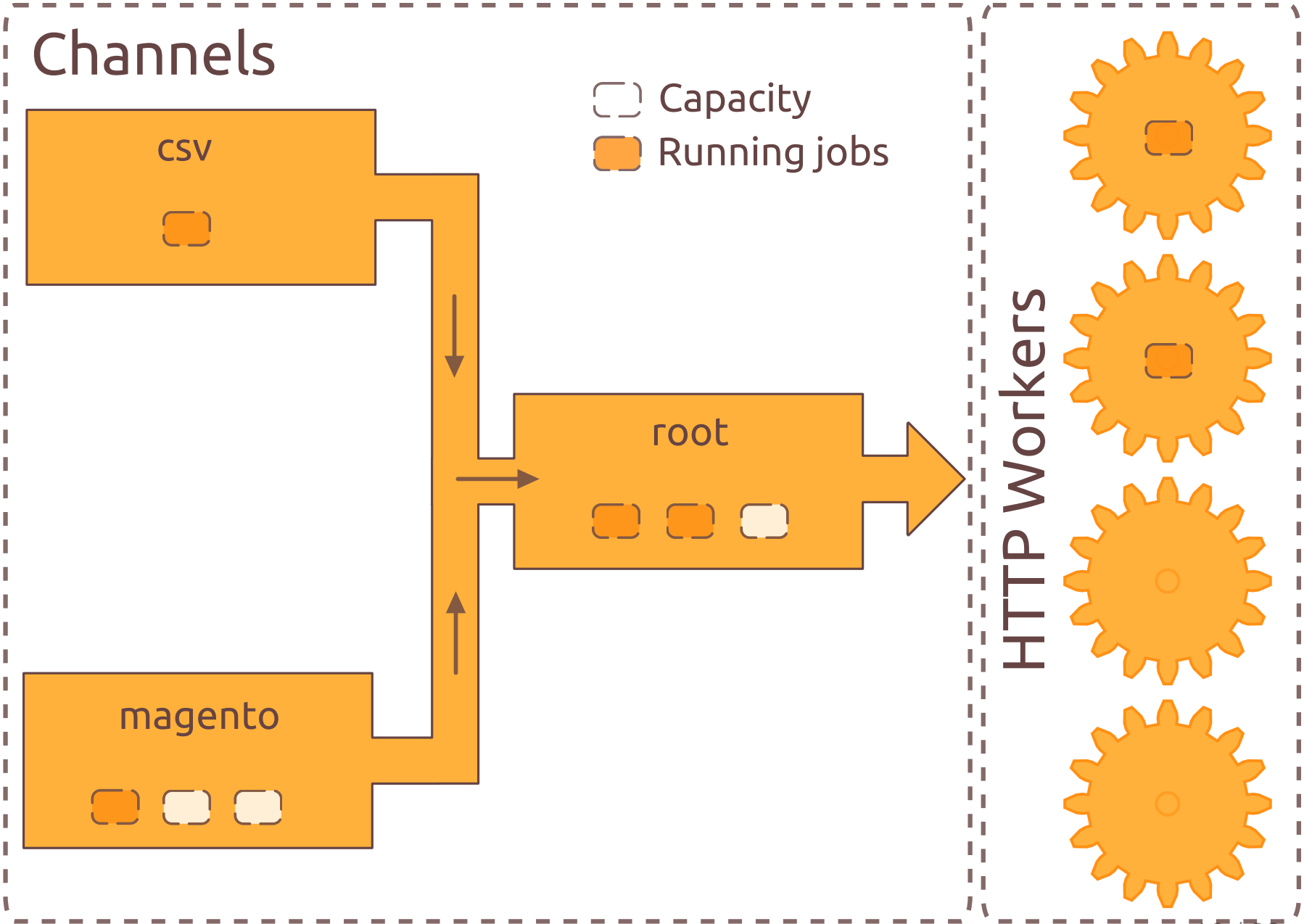
Dequeue it!

Start the server with:

```
ODOO_CONNECTOR_CHANNELS=root:2 ./openerp-server --load=web,connector --workers=4
```

```
ODOO_CONNECTOR_CHANNELS=root:3,root.csv:1,root.magento:3 \
    ./openerp-server --load=web,connector --workers=4
```

Channels


Channels
Capacity
Running jobs
csv
root
magento
HTTP Workers

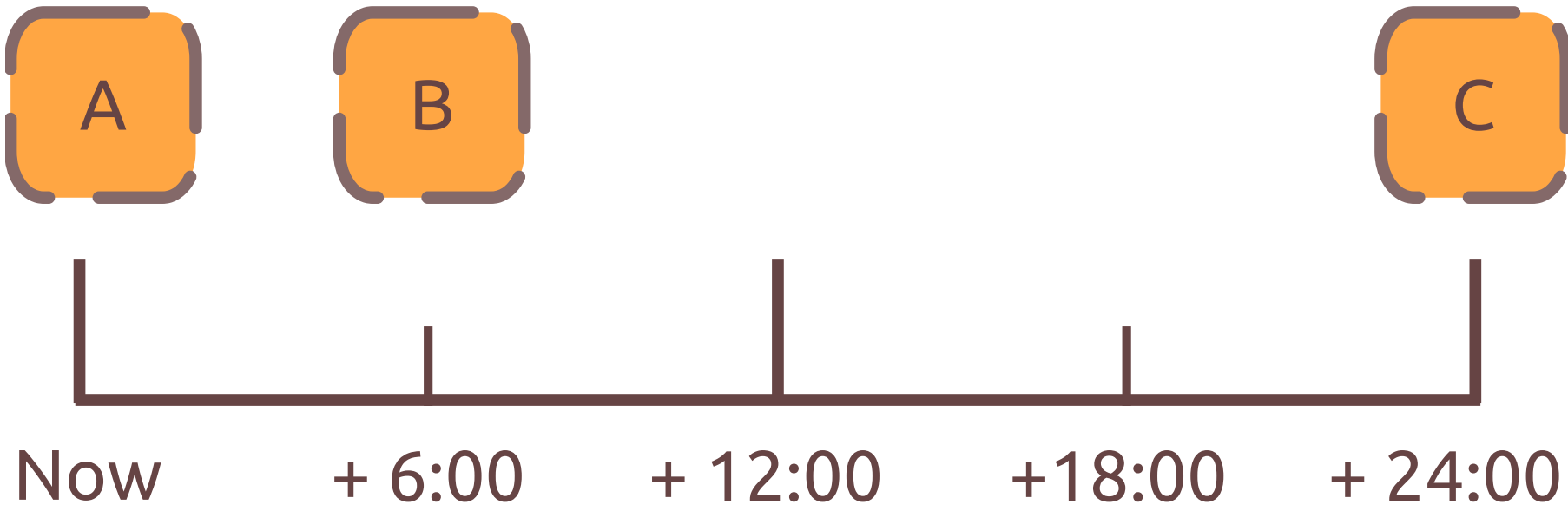
Properties

Priority

```
import_order.delay(session, 1)  # default is 10
import_order.delay(session, 2, priority=50)
import_order.delay(session, 10, priority=999)
```

ETA

```
import_order.delay(session, 1)  # A
import_order.delay(session, 1, eta=6*60*60)  # B
import_order.delay(session, 2, eta=datetime.now() + timedelta(days=1))  # C
```



Retries

```
import_order.delay(session, 1, max_retries=3)
```

Retries

```
import_order.delay(session, 1, max_retries=3)
```

Invoke a retry

```
@job
def import_order(session, args):
    try:
        do_operation()
    except (socket.gaierror, socket.error, socket.timeout) as err:
        raise RetryableError(
            'A network error caused the failure of the job: '
            '%s' % err)
```

Best Practices

Outdating

Data in jobs can become outdated.

No:

```
@job
def example(session, record_id, vals):
    export(record_id, vals)
```

Yes:

```
@job
def example(session, record_id):
    export(session.env['model'].browse(record_id))
```

Outdating

Existence

A job can refer to a record which has been deleted. Always check if it still exists.

No:

```
@job
def example(session, record_id):
    export(session.env['model'].browse(record_id))
```

Yes:

```
@job
def example(session, record_id):
    record = session.env['model'].browse(record_id)
    if record.exists():
        export(record)
```

Outdating

Existence

Idempotence

A job should, when possible, produce the same result when executed several times.

No:

```
@job
def example(session, record_id):
    export(session.env['model'].browse(record_id))
```

Yes:

```
@job
def example(session, record_id):
    record = session.env['model'].browse(record_id)
    if record.exists():
        if not record.exported:
            export(session.env['model'].browse(record_id))
```

Useful Patterns

Fanout Job

A job generating other jobs.

```
@job
def import_file(session, filepath):
    with open(filepath) as f:
        for line in f:
            import_line.delay(session, line)
```

Try or delay

If an operation failed, try it later.

```
@job
def do_operation(session, args):
    # work

try:
    do_operation(session, args)
except TimeoutError:
    do_operation.delay(session, args, eta=10*60)
```

Extract highly concurrent tasks

And put them in a one-by-one channel.

Thanks!

Thanks!

OCA Sponsors