Pterodactyl addon - Server Importer Pro

Welcome to the Server Importer Pro installation file !

First of all thanks for your purchase we really appreciate.

Panel Installation

We are going to edit your panel files. For do that please use a **Visual studio code** app with SSH extension (**Recommended**),SFTP session (with FileZila for exemple) Or SSH.

Before start we ask you to upload the content of the provided `panelfiles` folder to your Pterodactyl folder.

1. In *resources/scripts/components/server/events.ts* after `BACKUP_RESTORE_COMPLETED = 'backup restore completed',` add :

```
IMPORT_STARTED = 'import started',
IMPORT_COMPLETED = 'import completed',
```

2. In *resources/scripts/api/server/types.d.ts* replace `export type ServerStatus = 'installing' | 'install_failed' | 'suspended' | 'restoring_backup' | null;` by :

```
export type ServerStatus = 'installing' | 'install_failed' | 'suspended' |
'restoring_backup' | 'import' | null;
```

3. This file is not as the same place on every Pterodactyl version. If you are on a version under 1.9 open the *resources/scripts/routers/ServerRouter.tsx* file else open the *resources/scripts/components/server/ConflictStateRenderer.tsx* file. In this file add under `import ServerErrorSvg from '@/assets/images/server_error.svg';` :

```
import ServerImportSvg from '@/assets/images/import.svg';
```

4. In the same file replace :

```
isNodeUnderMaintenance ? (
  <ScreenBlock
  title={'Node under Maintenance'}
  image={ServerErrorSvg}
  message={'The node of this server is currently under maintenance.'}
  />
) :
```

By :

```
isNodeUnderMaintenance ? (
  <ScreenBlock
  title={'Node under Maintenance'}
  image={ServerErrorSvg}
  message={'The node of this server is currently under maintenance.'}
  />
) : (
status === 'import' ?
  <ScreenBlock
  title={'Server import'}
  image={ServerImportSvg}
  message={'Your server is currently being imported, please check back in a few
minutes.'}
  />
) :
```

5. In *resources/scripts/components/server/settings/SettingsContainer.tsx* above `<Can action={'settings.reinstall'}>` add :

```
                    <Can action={'serverimporter.*'}>
                        <div css={tw`mb-6 md:mb-10`}>
                            <ImporterServerBox/>
                        </div>
                    </Can>
```

6. In same file after all import line add :

```
import ImporterServerBox from '@/components/server/settings/ImporterServerBox';
```

7. In *app/Models/Server.php* after `public const STATUS_RESTORING_BACKUP = 'restoring_backup';` add :

```
public const STATUS_IMPORT = 'import';
```

8. In *app/Repositories/Wings/DaemonServerRepository.php* above last `}` add :

```
public function importer(string $user, string $password, string $hote, int $port, string
$srclocation, string $dstlocation, bool $wipe, string $type): void {
    Assert::isInstanceOf($this->server, Server::class);
    try {
        $this->getHttpClient()->post(sprintf(
            '/api/servers/%s/import',
            $this->server->uuid
        ),
        [
            'json' => [
                'user' => $user,
                'password' => $password,
                'hote' => $hote,
                'port' => $port,
                'srclocation' => $srclocation,
                'dstlocation' => $dstlocation,
                'wipe' => $wipe,
                'type' => $type,
            ],
        ]);
    } catch (TransferException $exception) {
        throw new DaemonConnectionException($exception);
    }
}
```

9. In *app/Http/Controllers/Api/Client/Servers/FileController.php* above last `}` add :

```
    /**
    * Import sftp or ftp server file to server
    *
    * @throws \Pterodactyl\Exceptions\Http\Connection\DaemonConnectionException
    */
    public function importer(ImporterRequest $request, Server $server)
    {
        $this->importServerService->handle($server, $request->get('user'), $request-
>get('password'), $request->get('hote'), $request->get('port'), $request-
>get('srclocation'), $request->get('dstlocation'), $request->get('wipe'), $request-
>get('type') );
        return new JsonResponse([], Response::HTTP_ACCEPTED);
    }
```

10. In same file after all lines that start by `use` (on beginning of file) :

```
use Pterodactyl\Http\Requests\Api\Client\Servers\Files\ImporterRequest;
use Pterodactyl\Services\Servers\ImportServerService;
```

11. In same file under `private NodeJWTService $jwtService,` add :

```
private ImportServerService $importServerService,
```

12. Follow the guide for your Pterodactyl version (You can get it on admin → Overview)

▼ For panel **above or equal to the 1.9 version**

1. In routes/api-client.php after `Route::get('/upload', [Client\Servers\FileUploadControllerclass, 'index']);` add :

```
Route::post('/importer', [Client\Servers\FileControllerclass, 'importer']);
```

▼ For panel **under the 1.9 version**

1. In routes/api-client.php after `Route::get('/upload', 'Servers\FileUploadController');` add :

```
Route::post('/importer', 'Servers\FileController@importer');
```

14. Follow the guide for your Pterodactyl version (You can get it on admin → Overview)

▼ For panel **above or equal to the 1.9 version**

1. In routes/remote.php after `Route::post('/install', [Remote\Servers\ServerInstallController::class, 'store');` add :

```
Route::get('/import', [Remote\Servers\ServerImportController::class, 'index']);
```

▼ For panel **under the 1.9 version**

1. In routes/remote.php after `Route::post('/install', 'Servers\ServerInstallController@store');` add :

```
Route::get('/import', 'Servers\ServerImportController@index');
```

15. In `app/Models/Permission.php` after :

```
'schedule' => [
  'description' => 'Permissions that control a user's access to the schedule management
for this server.',
  'keys' => [
    'create' => 'Allows a user to create new schedules for this server.', //
task.create-schedule
    'read' => 'Allows a user to view schedules and the tasks associated with them for
this server.', // task.view-schedule, task.list-schedules
    'update' => 'Allows a user to update schedules and schedule tasks for this
server.', // task.edit-schedule, task.queue-schedule, task.toggle-schedule
    'delete' => 'Allows a user to delete schedules for this server.', // task.delete-
schedule
  ],
],
```

Add :

```
'serverimporter' => [
  'description' => 'Permissions that control a user's access to the import system',
  'keys' => [
    'access' => 'Allows a user to use import system.', // task.create-schedule
  ],
],
```

Perfect you have edited all files successfully.

Now if this was not already made we are going to install `yarn` on your server. For do that please run these commands.

```
apt -y install curl dirmngr apt-transport-https lsb-release ca-certificates
curl -sL https://deb.nodesource.com/setup_14.x | sudo -E bash -
apt -y install nodejs
cd /var/www/pterodactyl
npm i -g yarn && yarn
```

Now we are going to compile all panel files and add all required tables to your database:

```
yarn build:production
php artisan migrate
chown -R www-data:www-data *
```

If you got a **error during the `yarn build:production` command:**

Check first if your issue was not already resolved on https://help.bagou450.com. If you don't find your issue please send a email to our teams (You can found contact details at the end of this documentation) with the link returned by:

```
yarn build:production | curl -X POST -H "Content-Type: text/plain" --data-binary @-
https://haste.bagou450.com/documents
```

Perfect we are installed the addon on your Pterodactyl panel.

Wings Installation

Now we need to install it on your nodes. Please **follow these step on ALL your nodes**.

First of all you need to download wings source on your node. For that get first your wings version on your panel (Admin → Nodes → Your node).

This is a exemple of how download wings source on your server with wings 1.11.8. If you use another wings verison please replace all `1.11.8` by your wings version:

```
cd
wget https://github.com/pterodactyl/wings/archive/refs/tags/v1.11.8.zip
unzip v1.11.8.zip
mv v1.11.8 wings
cd wings
```

Now you get wings source on your panel.

We are now going to edit wings file. Same as for the panel you can use VSCode,SFTP or SSH for edit these files.

Before start don't forget to upload the content of `wingsfiles` folder to your wings folder (you can found your wings folder on your server user home folder)

1. In *router/router.go* after `server.POST("/reinstall", postServerReinstall)` add :

```
server.POST("/import", postServerImport)
```

2. At the end of *router/router_server.go* add:

```
// Reinstalls a server.
func postServerImport(c *gin.Context) {
    s := ExtractServer(c)
    var data struct {
        User        string `json:"user"`
        Password    string `json:"password"`
        Hote        string `json:"hote"`
        Port        int    `json:"port"`
        Srclocation string `json:"srclocation"`
        Dstlocation string `json:"dstlocation"`
        Wipe        bool   `json:"wipe"`
        Type        string `json:"type"`
    }
    if err := c.BindJSON(&data); err != nil {
        middleware.CaptureAndAbort(c, err)

    }
    if s.ExecutingPowerAction() {
        c.AbortWithStatusJSON(http.StatusConflict, gin.H{
            "error": "Cannot execute server import event while another power action is
running.",
        })
        return
    }
    go func(s *server.Server) {
        if err := s.ImportNew(data.User, data.Password, data.Hote, data.Port,
data.Srclocation, data.Dstlocation, data.Type, data.Wipe); err != nil {
            s.Log().WithField("error", err).Error("failed to complete server import
process")

        }
    }(s)
    c.Status(http.StatusAccepted)
}
```

3. In *server/events.go* after `TransferStatusEvent          = "transfer status"` add :

```
    ImportStartedEvent           = "import started"
    ImportCompletedEvent         = "import completed"
```

6. In `remote/servers.go` after

```
func (c *client) SetTransferStatus(ctx context.Context, uuid string, successful bool)
error {
    state := "failure"
    if successful {
        state = "success"
    }
    resp, err := c.Get(ctx, fmt.Sprintf("/servers/%s/transfer/%s", uuid, state), nil)
    if err != nil {
        fmt.Println(err)
        return err
    }
    _ = resp.Body.Close()
    return nil
}
```

Add:

```
func (c *client) SetImportStatus(ctx context.Context, uuid string, successful bool)
error {
    resp, err := c.Get(ctx, fmt.Sprintf("/servers/%s/import", uuid), nil)
    if err != nil {
        return err
    }
    _ = resp.Body.Close()
    return nil
}
```

7. In *remote/http.go* after `SetTransferStatus(ctx context.Context, uuid string, successful bool) error` add

```
SetImportStatus(ctx context.Context, uuid string, successful bool) error
```

Now we are going to install `go` on your server. If you already have it skip this (We install the version 1.22 of go do not hesitate to edit the download url if there are a more recent version on https://go.dev/dl):

```
cd
wget https://go.dev/dl/go1.22.0.linux-amd64.tar.gz
rm -rf /usr/local/go
tar -C /usr/local -xzf go1.22.0.linux-amd64.tar.gz
export PATH=$PATH:/usr/local/go/bin
```

You need to edit your .bashrc file for add "export PATH=$PATH:/usr/local/go/bin" in it (This is for keep go in next ssh session).

Now go is installed. We are going to compile wings files:

```
cd && cd wings
go get github.com/secsy/goftp
go build
```

If there are no error we can continue and replace the actual wings file

```
service wings stop
rm /usr/local/bin/wings
cp wings /usr/local/bin/wings
chmod u+x /usr/local/bin/wings
service wings start
```

Tips: If you don't want to do that on all your nodes you can simply copy past the compiled wings from `/usr/local/bin/wings` folder to your others nodes (You need to restart wings after that).

Thanks for following this guide. Your addon is now installed.

If you have any problems you can first check our helpdesk at https://help.bagou450.com.

If you still have a problem you can contact us at contact@bagou450.com or on https://bagou450.com/contact