Developer Operator Ibexa Digital Experience Platform

Developer Operator

xrow GmbH

Copyright xrow GmbH

Table of contents

Copyright xrow GmbH

1. Developer Operator

PDF Download

1.1 Installation

1.1.1 Helm chart installation

The installation step needs to be done only once per cluster. You may skip this step if you have a cluster already enabled.

Install stable:

```
helm upgrade --install developer-operator-crd oci://registry.gitlab.com/xrow-public/repository/developer-operator-crd --version 1.1.21 -n kube-system
```

```
helm upgrade --install developer-operator oci://registry.gitlab.com/xrow-public/repository/developer-operator --version 1.1.21 -n kube-system
```

Install latest:

```
helm upgrade --install developer-operator-crd oci://registry.gitlab.com/xrow-shared/developer-operator/charts/developer-operator-crd --version 0.0.0+615cf75 -n kube-system
```

```
helm upgrade --install developer-operator oci://registry.gitlab.com/xrow-shared/developer-operator/charts/developer-operator --version 0.0.0+615cf75 -n kube-system
```

```
The latest unstable build is version 0.0.0+615cf75 from 2023-03-21 15:13:42+01:00 by bjoern.dieding
```

1.1.2 Helm chart uninstall

```
kubectl delete developer --all-namespaces --all
helm uninstall developer-operator -n kube-system
kubectl delete crd developers.xrow.com
```

1.1.3 Kubernetes Server Setup

Skip these steps if you have done it already

- Edit */etc/sysctl.conf* and add the line `fs.inotify.max_user_watches=524288`
- run `sudo sysctl -p`

Letztes Update: September 30, 2022

Copyright xrow GmbH

1.2 The Developer Workspace

1.2.1 Prerequisites

- A running ibexa installation set up via the ibexa operator or via the ibexa helm chart.

1.2.2 Workspace Setup

Take the sample YAML configuration and modify it:

- Look up the release name of your helm release and add it to use CRD as "release" under `spec`
- Enter your SSH public key, email, name and the desired container image
- Save your customized YAML configuration somewhere. Keep it for future reference.
- Open the `Kubernetes` console and add the `CRD`

Sample YAML to create a developer workspace

```
apiVersion: xrow.com/v1
kind: Developer
metadata:
  name: developer01
  namespace: ibexa-test
spec:
  release: ibexa
  key: "ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIOXGvbBWWfPKoMEU8zKMaKGLocGLVMAq4K24UZMyzjmc"
  email: "bjoern@xrow.de"
  name: "Björn Dieding"
  image:
    registry: registry.gitlab.com
    repository: xrow-shared/helm-ezplatform/ibexa-oss/dev
    tag: latest
  env:
    - name: APP_ENV
      value: dev
    - name: APP_DEBUG
      value: 1
```

1.2.3 Port Forward to your local maschine

```
kubectl config use-context 08.xrow.net
kubectl config set-context --current --namespace=project-test
kubectl port-forward service/bjoern-developer 2222:2222 &
kubectl port-forward service/bjoern-developer 8080:8080 &
kubectl port-forward service/bjoern-developer 8443:8443 &
kubectl port-forward service/$(kubectl get service | grep solr | awk '{print $1}') 8983:8983 &
kubectl port-forward service/$(kubectl get service | grep mysql | awk '{print $1}') 3306:3306 &
```

Alternativly you can forward with a tool like kubefwd

1.2.4 Connect to the Pod / development environment via an SSH Agent

Linux: Connect via SSH

```
[root@host ~]# ssh root@127.0.0.1 -p 2222 -A
Last login: Tue Feb 18 00:37:33 2020 from 10.128.0.1
Hi Björn Dieding <bjoern@xrow.de>, you can start coding
-bash-4.2$ ssh -T git@gitlab.com
Warning: Permanently added 'gitlab.com,35.231.145.151' (ECDSA) to the list of known hosts.
Welcome to GitLab, @xrow!
```

Connect via VS CODE

- Use the vscode kubernetes extension
- Select the desired SVC or POD and select forward all

Copyright xrow GmbH

1.2.5 Start to code

- Start additional tools needed for development

```
yarn install
$(yarn bin)/encore dev-server --host 0.0.0.0 --port 9000 --config-name ibexa --allowed-hosts 192.168.222.2 --public http://192.168.222.2:9000/
```

- Modify any file and commit to git.

Letztes Update: March 2, 2023

Copyright xrow GmbH

1.3 Configuration settings

1.3.1 Common settings

Configuration	Default	Description	Sample
env	{}	Env Variables	-
ip	null	Optional remote IP	192.168.4.2
image.repository	registry.gitlab.com/xrow-shared/ezplatform/dev	image to use	-
image.tag	latest	Use hash or default	-
image.pullPolicy	Always	-	-
imageCredentials.registry	-	Not required since the service account has access	registry.gitlab.com
imageCredentials.username	-	Not required since the service account has access	gitlab+deploy-token-183447
imageCredentials.password	-	Not required since the service account has access	ABCDEF1234
storage.enabled	true	enable or disable persistance for your code	-
storage.class	-	set a custom storage class	-
storage.mount	/opt/app-root/src/public/var	set the nfs mount point for your project	-
claimName	-	Name of the mounted claim wiht data	-
configMap	-	Name of the config map with env settings	-

1.3.2 PHP Symfony

See https://gitlab.com/xrow-public/docker-php-ezplatform

```
apiVersion: xrow.com/v1
kind: Developer
....
  env:
    - name: APP_ENV
      value: dev
    - name: APP_DEBUG
      value: 1
...
```


Copyright xrow GmbH

1.3.3 Node

See https://gitlab.com/xrow-public/s2i-nodejs-container

```
apiVersion: xrow.com/v1
kind: Developer
....
  env:
    - name: NODE_ENV
      value: development
    - name: DEV_MODE
      value: 'true'
...
```

Letztes Update: August 10, 2022

Copyright xrow GmbH

1.4 Day two operations

1.4.1 Setting an ENV variable

You can set an ENV variable for different purposes in different ways. By default, helm, kubernetes and ConfigMap variables for services are injected by default.

View them by typing `env`.

If you want to set a static env variable, set them via .s2i/environment. It might be better though to set a static symfony parameter. If you want to set variables that differ per stage, set them in the helm chart or via ezplatform / developer operator.

```
env:
- name: APP_ENV
  value: "dev"
- name: APP_DEBUG
  value: "1"
```

1.5 Recreate all dev pods

To force the regenaration of all pods you can follow this logic.

```
oc get Developer --all-namespaces
oc get Developer --all-namespaces -o yaml > export.yaml
oc get developer --no-headers=true --all-namespaces | sed -r 's/(\S+)\s+(\S+).*/oc --namespace \1 delete developer \2/e'
oc apply -f export.yaml
```

Letztes Update: February 2, 2022

Copyright xrow GmbH

1.6 Working with VS Code

This guide helps you connet your recommended IDE VS Code.

1.6.1 Setup the IDE VS Code

- Install the IDE VS Code
- Install the kubernetes extension in VS Code
- Install the remote SSH extension in VS Code
- Aquire a kubeconfig from an administrator from a service account with sufficent rights to connect to the api.
- Place the kubeconfig in the standard kubernetes directory ~/.kube/config
- Use the kubernetes extension, select the proper kubeconfig and connect to kubernetes via the extension. Errors fetching namespaces might be desired from the administrator.
- Create a Developer CRD wiht your public key credentials
- Navigate to your developer pod created by the Developer CRD under workloads and right click to forward all ports
- Connect via remote SSH wiht this recommended settings:

~/.ssh/ssh_config

```
Host "Development Pod"
  HostName 127.0.0.1
  User root
  Port 2222
  ForwardAgent yes
  IdentityFile ~/.ssh/id_ed25519
  StrictHostKeyChecking no
  UserKnownHostsFile /dev/null
  # ProxyJump vagrant
```

- If you are connecting over a vs code remote shell, you can use the ProxyJump setting from above.
- If you are connected, open the projects root folder
- **You are now ready to start coding.** You can switch to the master branch and create for example a feature branch.
- You can access your webserver via http://127.0.0.1:8080 or https://127.0.0.1:8443

In case you use vs code via a remote host you can make kubectl port-forward listen to all interfaces via alias in your profile (e.g. /etc/profile.d/vscode.sh).

/etc/profile.d/vscode.sh

```
# Make vs code listen too all interfaces
kubectl() {
    if [ "$1" = "port-forward" ]
    then
        vars=${@:2}
        echo "Alias kubectl port-forward --address 0.0.0.0 $vars"
        command kubectl port-forward --address 0.0.0.0 $vars
    else
        command kubectl $@
    fi;
}
```

If you don`t use a ForwardAgent SSH agent you can add the key to development pod

1. Add a new connection
2. Place your key in `~/.ssh/id_ed25519` or create a new key with `ssh-keygen -t ed25519`
3. `chmod 600 ~/.ssh/id_ed25519`

Copyright xrow GmbH

1.6.2 Guide to create a kubeconfig from the default service account

This is not fully working or incomplete. We keep this part for future reference.

```
#!/bin/bash
mkdir -p ~/.kube
chmod 600 ~/.kube

# Point to the internal API server hostname
APISERVER=https://kubernetes.default.svc
# Path to ServiceAccount token
SERVICEACCOUNT=/var/run/secrets/kubernetes.io/serviceaccount
# Read this Pod's namespace
NAMESPACE=$(cat ${SERVICEACCOUNT}/namespace)
# Read the ServiceAccount bearer token
TOKEN=$(cat ${SERVICEACCOUNT}/token)
# Reference the internal certificate authority (CA)
CACERT=${SERVICEACCOUNT}/ca.crt
# Explore the API with TOKEN
curl --cacert ${CACERT} --header "Authorization: Bearer ${TOKEN}" -X GET ${APISERVER}/api

TOKEN=$(cat /var/run/secrets/kubernetes.io/serviceaccount/token)
NAMESPACE=$(cat /var/run/secrets/kubernetes.io/serviceaccount/namespace)
cat << EOF > ~/.kube/config
apiVersion: v1
clusters:
- cluster:
    server: https://kubernetes.default.svc:8443
  name: local
contexts:
- context:
    cluster: local
    namespace: $NAMESPACE
    user: namespace-token-user
  name: default
current-context: default
kind: Config
preferences: {}
users:
- name: namespace-token-user
  user:
    token: $TOKEN
EOF
chmod 600 ~/.kube/config
```

Letztes Update: February 2, 2023

Copyright xrow GmbH

1.7 How to use XDEBUG

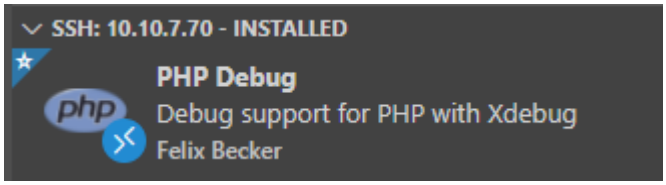
1.7.1 Modify the Developer CRD

See the list of all available modes .

```
apiVersion: xrow.com/v1
kind: Developer
....
  env:
    - name: XDEBUG_MODE
      value: debug
...
```

1.7.2 Setup VSCODE

- start Visual Studio Code
- go to **Extensions** in the toolbar on the left
- search for **PHP Debug** by Felix Becker
- click on **Install** on a remote host



Before actually using Xdebug a few things need to be prepared:

- In VS Code open the file `.vscode/launch.json` in your project's root directory and edit it`s content.

```
{
    "version": "0.2.0",
    "configurations": [
        {
            "name": "Listen for Xdebug",
            "type": "php",
            "request": "launch",
            "port": 9003,
            "stopOnEntry": false
        }
    ]
}
```

- Decide change `stopOnEntry` to **true** if needed.

1.7.3 Run a debugging session

Keep in mind that you always need to trigger a function if you want to debug it.

1. either by reloading the frontend
2. or by executing it via terminal if it can't be triggered by refreshing the frontend

Starting a debugging session

- open a file in the IDE which should be debugged
- create one or multiple breakpoints by clicking in the beginning of the line which should be investigated
- go to tab **Run** and click on **Start Debugging**
- If necessary select "XDebug" in the new toolbar **Listen for Xdebug**
- If you use **the browser** open a URL like `http://192.168.222.2:8080/dashboard` to start debugging

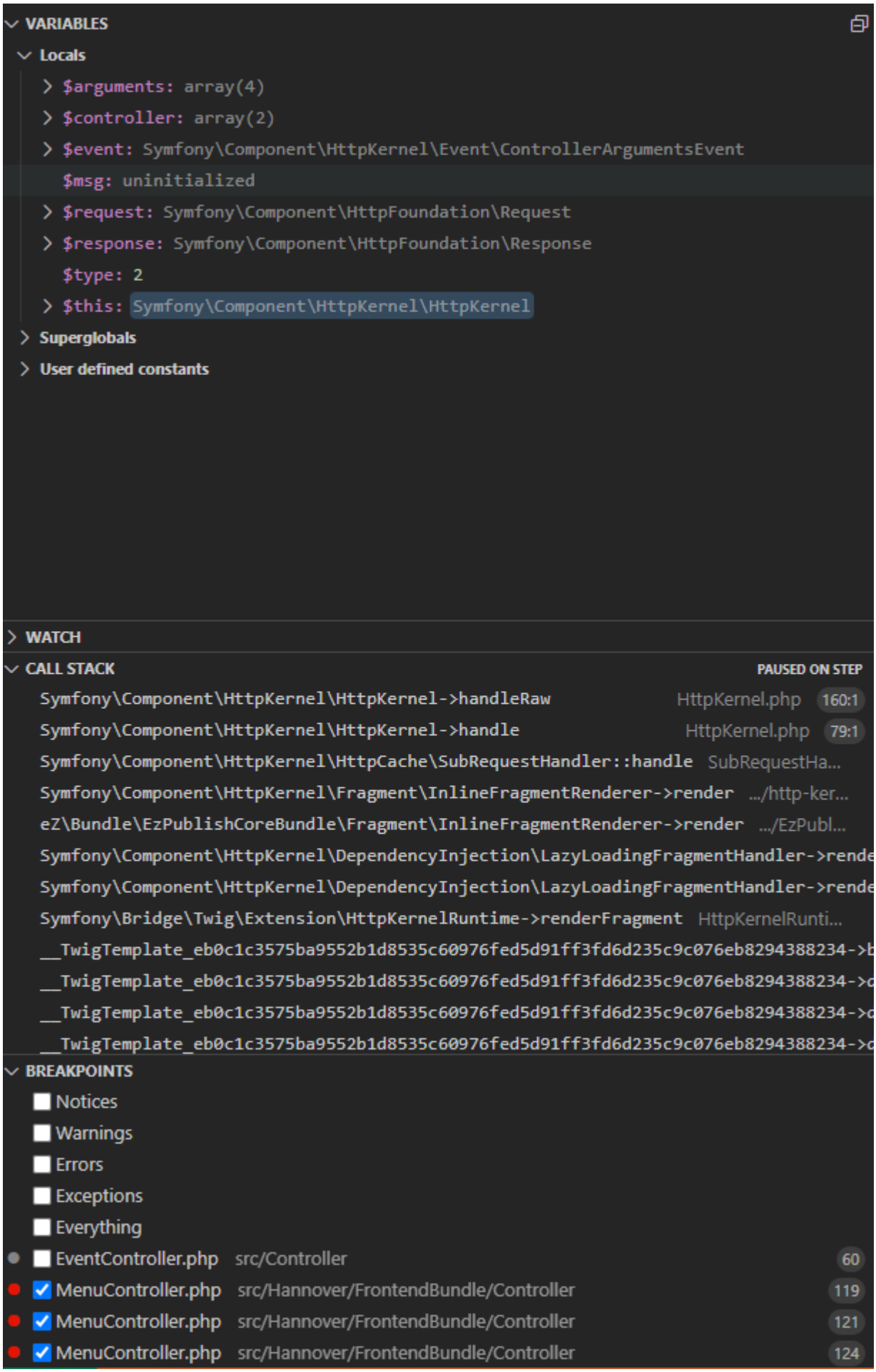
Copyright xrow GmbH

- If you use the **the CLI** run a command like `console ibexa:migrations:migrate` to start debugging

After this the following scenarios are possible:

1. the tool gives a result without doing anything additional
2. the tool gives a result only after refreshing the dev-pod's frontend
3. the tool gives a result not until the affected function was called via terminal
4. if the breakpoint was reached the results are listed in the menu on the left side
5. the first section **VARIABLES** shows the used variables during the debug call
6. the second section **CALL STACK** shows which classes and functions were used
7. the third section **BREAKPOINTS** allows to filter events

Copyright xrow GmbH



Copyright xrow GmbH

1.7.4 The debugging toolbar

After Xdebug reached a breakpoint different options become available. These options can be found in the Xdebug-Toolbar that appears in VS Code after starting the debug process.

- **Continue**: navigate to the next breakpoint
- **Step Over**: jump to the next function/variable/if-statement etc. (breakpoints won't be considered)
- **Step In**: go inside a statement and jump to the related function that is used
- **Step Out**: go outside of a statement and jump either to the previous breakpoint or to the overlaying function
- **Restart**: start the debug process from the beginning (first breakpoint)
- **Stop**: end the debug process and close the toolbar

Letztes Update: August 11, 2022

Copyright xrow GmbH

1.8 How to use Blackfire

1.8.1 Modify the Developer CRD

Copy your credendials.

```
apiVersion: xrow.com/v1
kind: Developer
....
  env:
  - name: BLACKFIRE_CLIENT_ID
    value: XXX
  - name: BLACKFIRE_CLIENT_TOKEN
    value: XXX
  - name: BLACKFIRE_SERVER_ID
    value: XXX
  - name: BLACKFIRE_SERVER_TOKEN
    value: XXX
...
```

1.8.2 Run some requests

You need to copy the cookie from your browser into the env frist.

```
COOKIE="eZSESSID21232f297a57a5a743894a0e4a801fc3=6a496ea21b98c357ff1e7c0cc0082a57"

blackfire client:curl \
  -H "x-siteaccess: admin" \
  -H "Cookie: ${COOKIE}" \
  http://127.0.0.1:8080/dashboard
```

```
COOKIE="eZSESSID21232f297a57a5a743894a0e4a801fc3=6a496ea21b98c357ff1e7c0cc0082a57"

blackfire client:curl \
  -H "x-siteaccess: admin" \
  -H "Cookie: ${COOKIE}" \
  -H "accept: application/vnd.ez.api.View+json"  \
  -H "content-type: application/vnd.ez.api.ViewInput+json;version=1.1" \
  --data-raw '{"ViewInput":{"identifier":"navigation","public":false,"LocationQuery":{"FacetBuilders":{},"Filter":
{"ContentTypeIdentifierCriterion":"page","SubtreeCriterion":"/1/2/","VisibilityCriterion":false},"SortClauses":{"LocationDepth":"ascending"},"limit":
300,"offset":0}}}' \
  "http://127.0.0.1:8080/api/ezp/v2/views" \
  --compressed
```

Visit the Blackfire dashboard.

Letztes Update: August 11, 2022

Copyright xrow GmbH

1.9 Troubleshooting

1.9.1 Image Pull Backoff

This could have the following reasons:

- You are notified that your images repositories credentials are incorrect or the image doen`t exists.
- Your image is very big.
- You use persistent storage for your code. Which slows down the inital startup.

1.9.2 Image tag latest gives 500

- Try to find an older version of that image that works.

1.9.3 Your pod returns a 500 errors / Using an image with code errors

Things to consider:

- latest image is broken
- try a different image
- your code might be broken, try to fix the code
- the data in the database might be awkward and might cause the application to fail
- binary data might be missing that causes an exception in the code

1.9.4 Your POD IP is not reachable

- You might be running two developer CRCs that use the same external IP

1.9.5 The result you see in the browser is not the result you expect

- You have been using your OS hosts file to address the POD, meanwhile you changed the IP of the POD and your hosts file, but browser doesn`t notice. Restart the browser.

1.9.6 Can`t login via ssh

Error: child_set_env: too many env vars

https://github.com/brl/obfuscated-openssh/blob/master/session.c#L972

This can be fixed by removing the amount of pods in a namespace.

Copyright xrow GmbH

1.9.7 Dev the system is slow in general

Here are hint on how fast a System should be

Scenario	Result
APP_ENV=prod, XDEBUG_MODE=, BLACKFIRE-*=	Request will take 1 sec it has no overhead
APP_ENV=dev,APP_DEV=1, XDEBUG_MODE=, BLACKFIRE-*=	Request will take 3 sec it has overhead of the webdeveloper toolbar
APP_ENV=dev,APP_DEV=1, XDEBUG_MODE=develop, BLACKFIRE-*=	Request will take 6, if the IDE is connected. **It will take 60+ sec if the IDE is not listending to the request, because XDEBUG can`t reach the IDE (timeout). See /tmp/xdebug.log**
APP_ENV=prod,APP_DEV=,XDEBUG_MODE=, BLACKFIRE-*=XXX	Request will take 4 sec, has overhead of the Blackfire runnign in the background

1.9.8 If you use the env APP_DEV set to dev the system is slow

- Check the logs. There might be to many unseen errors.

1.9.9 If you use the env XDEBUG_MODE set to dev the system is slow

- Check /tmp/xdebug.log for connection errors, this mean you have not startet the remote debugging
- If you do not need to debug pph code remove XDEBUG_MODE from env

1.9.10 You see an out of memory error

- You might have var_dump in the code.

Letztes Update: September 2, 2022

Copyright xrow GmbH

1.10 Guide for Administrators

1.10.1 Grant Access via a auth provider

- Grant a Gitlab User Access

Setup a new Application Token

Add the gitlab auth provider

```
oauthConfig:
  assetPublicURL: https://openshift.XXX.xrow.net:8443/console/
  grantConfig:
    method: auto
  identityProviders:
  - challenge: true
    login: true
    mappingMethod: lookup
    name: gitlab
    provider:
      apiVersion: v1
      clientID: XXX
      clientSecret: XXX
      kind: GitLabIdentityProvider
      legacy: false
      url: https://gitlab.com/
```

Restart Controller

For Openshift 3.11:

```
master-restart api
master-restart controllers
```

1.10.2 Map an identity

Create a cluster admin user

```
oc create identity gitlab:44745
oc create useridentitymapping gitlab:44745 admin
```

Create a namespace admin

```
oc create identity gitlab:44745
oc create user service@xrow.de --full-name="xrow GmbH"
oc policy add-role-to-user admin service@xrow.de -n project-name-production
oc create useridentitymapping gitlab:44745 service@xrow.de
```

Create a namespace developer:

Uses the special role developer-operator-developer.

```
oc create identity gitlab:44745
oc create user service@xrow.de --full-name="xrow GmbH"
oc policy add-role-to-user developer-operator-developer service@xrow.de -n project-name-production
oc create useridentitymapping gitlab:44745 service@xrow.de
```

Revoke access

```
oc delete useridentitymapping gitlab:44745
```

Letztes Update: February 2, 2022

Copyright xrow GmbH