

Deploying Plone and Volto – the Hard Way

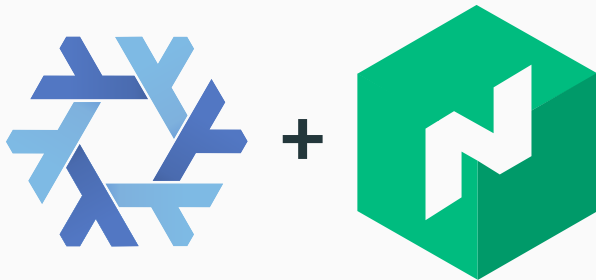
Plone Conference 2020

Asko Soukka

9.12.2020



JYVÄSKYLÄN YLIOPISTO
UNIVERSITY OF JYVÄSKYLÄ



**I DON'T ALWAYS
DEPLOY OUR PLONES**

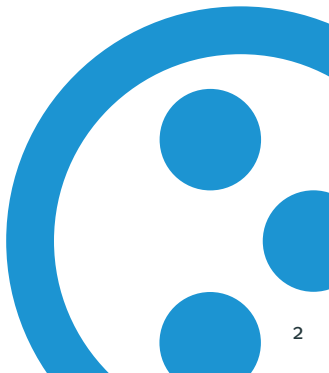


Asko Soukka

Software architect at University of Jyväskylä Digital Services

Background

- Python developer since 2002
- Plone developer since 2004
- Full-time professional since 2008
- Nix / NixOS user since 2015



Taking the Red Pill...

- ✗ Buildout
- ✗ WSGI
- ✗ Docker
- ✗ Registry

- ✓ Pip
- ✓ Nix
- ✓ TxZServer
- ✓ Nomad
- ✓ Levant

Nomad wonderland



One Job File to Rule Them All

- task groups
 - instance count
 - update policy
 - server resources
 - volume mounts
 - ...
- tasks
 - consul services
 - vault secrets
 - env variables
 - exec artifacts
 - ...

Nomad Isolated Fork/Exec Driver

Nix-built artifact

```
artifact {  
  source = "https://...app-[[ .app.version ]].tar.gz"  
  destination = "/"  
}
```

Runs on minimal chroot

```
/etc/group  
/etc/passwd  
/etc/nsswitch.conf  
/etc/resolv.conf  
/etc/ssl/certs
```


Canaries
1 / 1Placed
4Desired
4Healthy
3Unhealthy
0




Deployment is running pending automatic promotion

[Show deployment task groups and allocations](#)

Task Groups

| Name | Count | Allocation Status | Volume | Reserved CPU | Reserved Memory | Reserved Disk |
|-------------|-------|------------------------|--------|--------------|-----------------|---------------|
| camunda | 1 | <div><div></div></div> | | 300 MHz | 1152 MiB | 300 MiB |
| volto | 1 | <div><div></div></div> | | 200 MHz | 128 MiB | 300 MiB |
| zeoinstance | 1 | <div><div></div></div> | Yes | 100 MHz | 512 MiB | 300 MiB |
| zeoserver | 1 | <div><div></div></div> | Yes | 100 MHz | 512 MiB | 300 MiB |

Recent Allocations

| ID | Task Group | Created | Modified | Status | Version | Client | Volume | CPU | Memory |
|----------|-----------------------------|-----------------------|-------------------|---|---------|--------------------------|--------|------------------------|------------------------|
| eb8d544f | zeoinstance | Dec 03 21:01:40 +0200 | a few seconds ago |  pending | 4 | aafc341d | Yes | | |
| f327d81f | volto | Dec 02 20:34:54 +0200 | a few seconds ago |  running | 4 | 3646f8d5 | | <div><div></div></div> | <div><div></div></div> |
| 3bd54696 | camunda | Dec 02 | a few |  running | 4 | 3646f8d5 | | <div><div></div></div> | <div><div></div></div> |

Nix-built Nomad artifacts

One Package Manager to Rule Them All

Nix-built Nomad deployment artifacts

Advantages

- 100 % reproducible
- production equals development
- sandboxed offline builds
- full dependency graph
- standalone tarballs
- no Dockerfile
- no base images
- no surprises

Disadvantages

- no conventions
- no metadata
- no shared layers
- no documentation



One Package Manager to Rule Them All

Nix-built Nomad deployment artifacts

Advantages

- 100 % reproducible
- production equals development
- sandboxed offline builds
- full dependency graph
- standalone tarballs
- no Dockerfile
- no base images
- no surprises

Disadvantages

- no conventions
- no metadata
- no shared layers
- no documentation

Some documentation

- <https://nixos.org>
- <https://nix.dev>



volto.tar.gz

```
{ pkgs ? import ../nix { nixpkgs = sources."nixpkgs-20.09"; }  
, sources ? import ../nix/sources.nix  
, build ? import ../default.nix { inherit pkgs; }  
, name ? "artifact"  
}:
```

```
with pkgs;
```

```
let
```

```
env = buildEnv {  
  name = "env";  
  paths = [  
    bashInteractive  
    coreutils  
    findutils  
    gnused  
    volto  
  ];  
};
```

```
closure = (writeReferencesToFile env);
```

```
in
```

```
runCommand name {  
  buildInputs = [ makeWrapper ];  
} ''
```

```
mkdir -p local/bin
```

```
makeWrapper ${bashInteractive}/bin/sh local/bin/sh \
```

```
--prefix PATH : ${coreutils}/bin \
```

```
--prefix PATH : ${build}/bin \
```

```
--prefix PATH : ${findutils}/bin \
```

```
--prefix PATH : ${gnused}/bin \
```

```
--prefix PATH : ${env}/bin
```

```
tar cvzhP \
```

```
--hard-dereference \
```

```
--exclude="${env}" \
```

```
--exclude="*ncurses*/ncurses*/ncurses*" \
```

```
--exclude="/nix/store/*-node_volto-starter-git*" \
```

```
--files-from=${closure} \
```

```
--transform="s|^local/||" \
```

```
local > $out || true
```

```
''
```

```
pkgs.stdenv.mkDerivation {  
  name = "volto";  
  src = pkgs.lib.cleanSource ./.;  
  builder = builtins.toFile "builder.sh" ''  
    source $stdenv/setup;  
    mkdir -p $out/bin $out/lib  
    cp -a $src $out/lib/volto && chmod u+w -R $out/lib/volto  
    cd $out/lib/volto  
    cp -a $env node_modules  
    HOST=CUSTOM_RAZZLE_SERVER_HOST PORT=CUSTOM_RAZZLE_SERVER_PORT \  
    RAZZLE_API_PATH=CUSTOM_RAZZLE_API_PATH \  
    node_modules/.bin/razzle build  
    chmod u+w -R node_modules && rm -r node_modules  
    cat > $out/bin/volto << EOF  
    #bash/bin/env sh  
    RUNTIME="$(mktemp -d)"  
    cp -R $out/lib/volto/build/* "$RUNTIME"  
    chmod u+w -R "$RUNTIME"
```

```
    find "$RUNTIME" -name "*.js"|xargs sed -i "s|CUSTOM_RAZZLE_SERVER  
    find "$RUNTIME" -name "*.js"|xargs sed -i "s|CUSTOM_RAZZLE_SERVER  
    find "$RUNTIME" -name "*.js"|xargs sed -i "s|CUSTOM_RAZZLE_API_PA  
    find "$RUNTIME" -name "*.js"|xargs sed -i "s|$out/lib/volto/build  
    chmod u-w -R "$RUNTIME"  
    cd $out/lib/volto && node "$RUNTIME/server.js" $@  
    EOF  
    chmod u+x $out/bin/volto  
    wrapProgram $out/bin/volto \  
    --suffix PATH : $propagatedBuildInputs/bin \  
    --suffix NODE_ENV : production \  
    --suffix NODE_PATH : $env  
  '';  
  buildInputs = with pkgs; [ makeWrapper ];  
  propagatedBuildInputs = with pkgs; [ nodejs-14_x ];  
  inherit bash env;  
}
```

Nix – the assorted ugly parts

- every language has their own Nix-conventions
- Nix dependency generator ecosystem is complex
- Nix does not support cyclic dependencies
- no storage device is big enough for `/nix/store`
- many NPM packages want to call Internet on install
- some NPM packages ship with pre-built binaries
- ...



Plone without buildout



Plone 5.2.1 without Buildout

Our (legacy) approach for Plone with pip

- generated requirements.txt with buildout
- created Python environment with pip / Nix
- used pip-branch of z3c.autoinclude
- disabled `<includeDependencies />`
- generated instance skeleton with Nix
- forked `plone.recipe.zope2instance` into `plonectl`

zope.conf

```
{ pkgs ? import <nixpkgs> {}  
, generators ? import ./generators.nix {}  
, instancehome ? import ./instancehome.nix {}  
, var ? "$(PLONE_VAR)"  
};  
  
let configuration = generators.toZConfig {  
  
  # ...  
  
  zodb_db = {  
    main = {  
      cache-size = 40000;  
      mount-point = "/";  
      zeoclient = {  
        read-only = false;  
        read-only-fallback = false;  
        blob-dir = "${var}/blostorage";  
        shared-blob-dir = true;  
        server = "$(PLONE_ZEOSERVER_ADDRESS)";  
        storage = 1;  
        name = "zeostorage";  
        var = "${var}";
```

```
      cache-size = "128MB";  
    };  
  };  
  temporary = {  
    temporarystorage = {  
      name = "temporary storage for sessioning";  
    };  
    mount-point = "/temp_folder";  
    container-class = "Products.TemporaryFolder.TemporaryContainer";  
  };  
}; in  
  
pkgs.stdenv.mkDerivation {  
  name = "zope.conf";  
  builder = builtins.toFile "builder.sh" ''  
    source $stdenv/setup  
    cat > $out << EOF  
    $configuration  
    EOF  
  '';  
  inherit configuration;  
}
```

/bin/zeoinstance

```
zeoinstance = stdenv.mkDerivation {  
  name = "plone-zeoinstance";  
  zope_conf = import ./zconfig/zeoinstance.nix {};  
  plonesite_py = ./zconfig/plonesite.py;  
  builder = builtins.toFile "builder.sh" ''  
    source $stdenv/setup  
    mkdir -p $out/bin  
    cat > $out/bin/zeoinstance << EOF  
    #!$bash/bin/sh  
    mkdir -p \${PLONE_VAR}/filestorage  
    if [ ! -f \${PLONE_VAR}/.sentinel ]; then  
      $env/bin/python -m plonectl.cli instance -C $zope_conf run $plonesite_py  
      touch \${PLONE_VAR}/.sentinel  
    fi  
    $env/bin/python -m plonectl.cli instance -C $zope_conf console  
    EOF  
    chmod a+x $out/bin/zeoinstance  
  '';  
  inherit bash env;  
};
```

Plone 6 without Buildout

✓ Plone 6 is pip installable (hearsay)

```
$ python3 -m venv py
```

```
$ ./py/bin/pip install Plone Paste -c ...
```

```
$ ./py/bin/mkwsgiinstance -d .
```

```
$ ./py/bin/runwsgi -v etc/zope.ini
```

✗ instance templates and scripts are still maintained
in `plone.recipe.zope2instance`

TxZServer in Production

Plone 5.2.1 / Zope 4.1.3 / Twisted / WebSockets + ZMQ PubSub

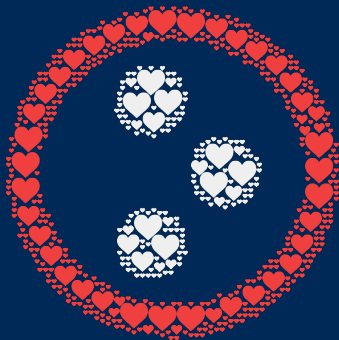
[sources]

```
ZServer = git git@github.com:datakurre/ZServer  
         branch=datakurre/master
```

```
collective.wsevents =  
    git git@github.com:datakurre/collective.wsevents
```

```
plonectl = git git@github.com:datakurre/plonectl
```

- ✓ in production since March 2020 without known issues
- ✗ upgrade to Plone > 5.2.1 and Zope > 4.1 still pending



datakurre.github.io/ploneconf2020/alt