Generic setup on Linux

- See separate manual for operations

Special server stuff for AWS (not ZEW server)

- Deploy in europe Zone B (Frankfurt)
- Request static IP address
- Connect with

```
ssh -i "giegerich-instance-priv-key.pem" ubuntu@[static-AWS-IP]
```

- Activate logins by password via
  https://aws.amazon.com/premiumsupport/knowledge-center/ec2-password-login/
- AWS connect by researchers:
  ssh ubuntu@[static-AWS-IP]
  Sparkish123

Server stuff continued - AWS or ZEW

- This is for Debian 11 - Bullseye
- This is a single user setup - all via user `otree`
- Based on otree server setup instructions
  https://otree.readthedocs.io/en/latest/server/ubuntu.html#server-ubuntu

System software

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install vim htop iotop tmux
```

App user `otree`

```
sudo adduser otree
sudo passwd  otree
# Koyaanis!32
# Login via password is disabled later on

# switch to user otree
sudo su -
su otree -
cd ~
echo 'PATH="$PATH:/etc/init.d"' >> ~/.bashrc

# as root
exit
visudo
# temporary add
```

```
# otree   ALL=(ALL) NOPASSWD: ALL

# fetch ~/.ssh/authorized_keys from another user...
cd /home/otree
cp -r  /home/pbu/.ssh .
chown -R otree:otree /home/otree

# exit SSH session
exit
exit

# check whether ssh login works without password
ssh otree@dmd.zew.de

# check whether sudo works
sudo su
exit
exit
```

From now on, continue as user otree.

Python setup

```
# as user otree
sudo apt-get install python3-pip git
sudo apt-get install python3-venv

python --version
# should be Python >= 3.9.2

# create a virtual environment 'venv_otree'
# to isolate the python module installation
python -m venv venv_otree

# make activation of venv permanent
vim ~/.bashrc
# add as last line
echo 'source ~/venv_otree/bin/activate' >> ~/.bashrc
# add execute for current session
source ~/venv_otree/bin/activate
# re-login via ssh - venv should be active
# (venv_otree) otree@dmd:~$
```

Postgres and database creation

```
sudo apt-get install postgresql postgresql-contrib

# password setting required for postgres user
sudo passwd postgres
# Koyaanis!32
# cannot login via SSH, thus no vulnerability

sudo systemctl status postgresql.service
sudo systemctl stop   postgresql.service
sudo systemctl start  postgresql.service
...

# log file of postgres
sudo less /var/lib/postgresql/13/main /var/log/postgresql/postgresql-13-main.log

# setting up a postgres database for otree
```

```
sudo su - postgres
psql
CREATE DATABASE django_db;
CREATE USER otree_user WITH PASSWORD 'sonne';
GRANT ALL PRIVILEGES ON DATABASE django_db TO otree_user;
\q
exit

# adding otree config settings to ssh shell environment permanently
vim ~/.bashrc
# add last line
export DATABASE_URL=postgres://otree_user:sonne@localhost/django_db

# also add to ~/.bashrc
export OTREE_ADMIN_PASSWORD=Crikey++1990
# uncomment this line to enable production mode
export OTREE_PRODUCTION=1
# export OTREE_AUTH_LEVEL=DEMO
export OTREE_AUTH_LEVEL=STUDY

# relogin via SSH to have settings take effect
# check using
export
# should show DATABASE_URL, OTREE_ADMIN_PASSWORD, OTREE_AUTH_LEVEL="STUDY", OTREE_PRODUCTION="1"
```

Whenever you change these ENV variables,
you also need to change them for existing `tmux` sessions.
See below...

Python to postgres libs for Debian

```
# We need psycopg2
# this is supposed to be covered by pip install psycopg2>=2.8.4 below,
# but since this fails
# I tried
#     sudo apt-get install python3-psycopg2
# this succeeds but does not resolve the issue
# but below solution does:
#     pip install --upgrade psycopg2-binary

# Those seem obsolete under debian 11
#     sudo apt-get install python3-dev
#     sudo apt-get install libpq-dev

# This seems obsolete under debian 11
#     sudo apt-get install redis-server
```

Otree setup

```
# assuming at least python 3.6.x

# Obsolete modules for older versions of otree:
# sudo pip install --upgrade otree-core
# pip uninstall otree-core

pip install otree
pip install --upgrade otree

otree startproject otree
# creating directory otree with lots of example projects
cd otree
```

```
# we might add this `cd otree` to .bashrc later

# specific researcher goes to
cd ~/otree/[your testproject]

pip install -r requirements.txt
#    pip install --upgrade psycopg2
# fails for psycopg2>=2.8.4
# Thus we might try to apt-get psycopg2 as debian package above.
# This succeeds but does not solve the issue.
#
# This succeeds and solves the issue
pip install --upgrade psycopg2-binary

otree resetdb
runprodserver 8080
CTRL+C
```

runprodserver on port 80

- otree recommends using hosting with Heroku;
  Linux deployments are not really supported.
- Under Linux, running a service on a port under 1000
  requires excessive privileges
- Reading up on running technically related Python Django web applications;
  it can only be done using an intermediary `Apache` or `Nginx`.
- But this is not allowed for otree, since
  "You cannot use Apache or Nginx as your primary web server,
  because oTree must be run with an ASGI server"
  I suppose, this is due to otree's use of websockets.
- My solution as follows

```
# allow otree to aquire port 80
sudo setcap cap_net_bind_service=+eip /home/otree/venv_otree/bin/otree

# secondly, we need to allocate root privileges
# sudo visudo
otree   ALL=(ALL) NOPASSWD: ALL

# we could restrict ourselves to only a few command lines
# for instance, in order to run
#    sudo /usr/bin/apt-get update
# we could configure
otree   ALL=(ALL) NOPASSWD:/usr/bin/apt-get update

# but need to transfer the environment of otree user to the root-context
# and we *cannot* configure this directly to visudo
# because of the command params to /usr/bin/sudo:
#    /usr/bin/sudo -E env PATH=$PATH /home/otree/venv_otree/bin/otree runprodserver 80


# we could try putting otree execution into a bash script
touch ~/otree/run.sh
chmod +x ~/otree/run.sh
echo '#!/bin/bash' >> ~/otree/run.sh
echo '/usr/bin/sudo -E env PATH=$PATH /home/otree/venv_otree/bin/otree runprodserver 80' >> ~/otree/run.sh

# and give it root privileges
```

```
otree   ALL=(ALL) NOPASSWD:/home/otree/otree/run.sh

# and execute it
sudo /home/otree/otree/run.sh

# now we have sufficient right for the otree root,
# but insufficient rights for spawning all kinds of workers

# thus we need to go back, and allocate full root privileges
otree   ALL=(ALL) NOPASSWD: ALL

# now we can run
/usr/bin/sudo -E env PATH=$PATH /home/otree/venv_otree/bin/otree runprodserver 80

# or shorter
sudo -E env "PATH=$PATH" otree runprodserver 80
```

Wrap up security considerations

- => Removing otree from sudoers seems impossible
- At least we can disable SSH login with password

```
sudo su
vim /etc/ssh/sshd_config
PasswordAuthentication no
systemctl restart ssh
# now only public key login is possible



# copy authorized_keys from another machine
cat ~/.ssh/id_rsa.pub | ssh username@server.address.com 'cat >> ~/.ssh/authorized_keys'
```

Creating and configuring access using an additional shared key

```
# creating private key
openssl genpkey -algorithm RSA -out dmd-otree.pem -pkeyopt rsa_keygen_bits:2048

# convert the *.pem to id_rsa
openssl rsa -in dmd-otree.pem -out id_rsa

# extracting the public key - in good format
ssh-keygen -f dmd-otree.pem -y > id_rsa.pub

# add contents of id_rsa.pub to  ~/.ssh/authorized_keys

# now you can login using this key
ssh -i %USERPROFILE%\.ssh\dmd-otree\dmd-otree.pem otree@dmd.zew.de

# use windows program PuTTYgen to convert id_rsa to *.ppk
#   Key -  Import key
#   File - Save private key
#      no passphrase
%USERPROFILE%\.ssh\dmd-otree\dmd-otree.ppk
# use this in TotalCommander
#   SFTP4TC - SSH - Auth - Private key file for authentication
#   but it must refer to ~\.ssh\id_rsa
#   no other directory possible :-(
```

systemd or circus

- For several years, the otree docs recommend `circus` to `demonize` otree.
- But none of their recipes work for Debian

Operation by researchers

- See separate manual for operations