Odoo SaaS Kit

Noteworthy Points -

1. After purchasing the module, please raise a request with our support team to get the related modules if not received at the time of purchase.

2. We also provide one-time free installation.

3. All Servers i.e. Main, Remote Servers & Database Server are suggested to be in a local n/w i.e. locally reachable from each other using Private Ips.

4. We will provide a dedicated docker image for the SAAS kit. It does not support other publicly available docker images.

5. We recommend having a dedicated server for the Postgresql Database.

6. Please use a separate Dedicated Database user for the SAAS kit i.e. different from the one Main Odoo uses. Generally, we use “odoosaas” user.

7. On Remote servers, please ensure that you configure Firewall Rules to allow traffic/access on this Port I.e 2375 from Main Server only. And block access to other sources.

8. For SSL, please buy or provision Wildcard SSL certs for the wildcard domain name.

OverView

Once you purchase the module, you will get the odoo saas kit. If you have purchased the installation also, then we will install the modules on your behalf.

1. odoo_saas_kit : Main add-on folder to be installed at the server end.

After purchasing the module, please raise a request with our support team to get the other related required modules from us. If you have purchased the installation, then we will install the modules on your behalf.

2. wk_saas_tool : Module needed to allow users to login to the clients & templates containers directly from the SAAS kit panel. And will be installed at the client's end.

Our Odoo SaaS Kit module uses Docker to create unique Odoo SaaS Instance containers for each of your Clients on the designated server.

The instance is the container of the docker. You can also decide whether you wish to deploy on the same server or a remote server. Docker Community Edition is used to make this work.

Before Installing the Module in your Odoo

First install the following python libraries to your system -

- docker (For Docker Python APIs)
- erppeek (For Odoo managed Tasks)
- paramiko (For remote SSH connections) In case you plan to have remote containers.

Use the following command in your command prompt to install the libraries >>
pip3 install docker erppeek paramiko

```
root@ip-172-31-46-122:~#
root@ip-172-31-46-122:~# pip3 install docker erppeek paramiko
Collecting docker
  Downloading docker-6.1.3-py3-none-any.whl (148 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 148.1/148.1 KB 3.1 MB/s eta 0:00:00
Collecting erppeek
  Downloading ERPpeek-1.7.1-py2.py3-none-any.whl (22 kB)
Collecting paramiko
  Downloading paramiko-3.3.1-py3-none-any.whl (224 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 224.8/224.8 KB 27.9 MB/s eta 0:00:00
Requirement already satisfied: urllib3>=1.26.0 in /usr/lib/python3/dist-packages (from docker) (1.26.5)
Collecting packaging>=14.0
```

We have to check that python is running in a virtual environment or main environment.

If python is running on virtual environment then we have to run the command
>>> source odoo-venv/bin/activate
pip install docker erppeek paramiko on that virtual python environment.
>>> deactivate

Installation & Setup of Odoo SaaS Kit

1. Place the Odoo_SaaS_Kit folder in the appropriate addons path folder.
2. Update the Odoo app list in Odoo.
3. Install the module from Odoo Backend.
4. Once successfully installed. Next step is to create few needed folders:

Note:- Here we are using odoo17 so we will keep the main folder name odoo17 which we will create inside opt directory

```
root@ip-172-31-37-153:/opt/odoo17# ll
total 68
drwxr-x--- 12 odoo17 odoo17 4096 Dec  5 10:54 ./
drwxr-xr-x  5 root   root   4096 Dec  5 13:15 ../
-rw-------  1 odoo17 odoo17  104 Dec  5 10:53 .bash_history
-rw-r--r--  1 odoo17 odoo17  220 Jan  6  2022 .bash_logout
-rw-r--r--  1 odoo17 odoo17 3771 Jan  6  2022 .bashrc
drwxr-xr-x  3 odoo17 odoo17 4096 Dec  5 10:45 .local/
-rw-r--r--  1 odoo17 odoo17  807 Jan  6  2022 .profile
drwxr-xr-x  8 odoo17 odoo17 4096 Dec  7 08:47 Odoo-SAAS-Data/
drwxr-xr-x  3 odoo17 odoo17 4096 Dec  5 13:13 common-addons_v15/
drwxr-xr-x  3 odoo17 odoo17 4096 Dec  5 13:13 common-addons_v16/
drwxr-xr-x  3 odoo17 odoo17 4096 Dec  5 13:13 common-addons_v17/
-rwxr-xr-x  1 odoo17 odoo17  739 Dec  5 10:52 docker.sh*
drwxrwxrwx  3 odoo17 odoo17 4096 Jul  7 04:27 dockerv15/
drwxrwxrwx  3 odoo17 odoo17 4096 Jul  7 04:27 dockerv16/
drwxrwxrwx  3 odoo17 odoo17 4096 Nov  8 15:21 dockerv17/
drwxrwxr-x 10 odoo17 odoo17 4096 Dec  5 10:36 odoo/
drwxr-xr-x  7 odoo17 odoo17 4096 Dec  5 13:17 webkul_addons/
root@ip-172-31-37-153:/opt/odoo17#
```

a.) Odoo-SAAS-Data:

This folder is created in the /opt/odoo17 folder. This folder keeps all the necessary files & folders for SAAS clients e.g Client's odoo.conf , odoo-server.conf , odoo-template.conf their data directory and Nginx Vhosts ,etc.

```
root@ip-172-31-37-153:/opt/odoo17/Odoo-SAAS-Data# ll
total 44
drwxr-xr-x  8 odoo17 odoo17 4096 Dec  7 08:47 ./
drwxr-x--- 12 odoo17 odoo17 4096 Dec  5 10:54 ../
drwxr-xr-x  2 odoo17 odoo17 4096 Dec  7 08:47 docker_vhosts/
drwxr-xr-x  3 odoo17 odoo17 4096 Dec  6 13:53 now.simranpal.tk/
-rw-r--r--  1 odoo17 odoo17  313 Dec  5 13:14 odoo-server.conf
-rw-r--r--  1 odoo17 odoo17  333 Dec  5 13:14 odoo-template.conf
-rw-r--r--  1 odoo17 odoo17  312 Dec  5 13:14 odoo.conf
drwxr-xr-x  3 odoo17 odoo17 4096 Dec  5 13:49 odoo15_template_cont/
drwxr-xr-x  3 odoo17 odoo17 4096 Dec  5 13:45 odoo16_template_cont/
drwxr-xr-x  3 odoo17 odoo17 4096 Dec  5 13:30 odoo17_template_cont/
drwxr-xr-x  3 odoo17 odoo17 4096 Dec  7 08:47 ruthless.simranpal.tk/
root@ip-172-31-37-153:/opt/odoo17/Odoo-SAAS-Data#
```

b.) docker_vhosts:
This folder should be inside Odoo-SAAS-Data & will contain the Nginx Virtual hosts files for clients in order to have sub-domains per client.

c.) common_addons:
The folder will be shared amongst all the Clients & templates as an addons path. The directory will be created inside /opt/odoo17. The modules placed in this directory will be visible to all Clients & Templates.

This directory must be named versioned specific like for example if we are using odoo of version 17 then the directory must be named as common-addons_v17. In this directory we will place the wk_saas_tool directory.

```
root@ip-172-31-37-153:/opt/odoo17# ll
total 68
drwxr-x--- 12 odoo17 odoo17 4096 Dec  5 10:54 ./
drwxr-xr-x  5 root   root   4096 Dec  5 13:15 ../
-rw-------  1 odoo17 odoo17  104 Dec  5 10:53 .bash_history
-rw-r--r--  1 odoo17 odoo17  220 Jan  6  2022 .bash_logout
-rw-r--r--  1 odoo17 odoo17 3771 Jan  6  2022 .bashrc
drwxr-xr-x  3 odoo17 odoo17 4096 Dec  5 10:45 .local/
-rw-r--r--  1 odoo17 odoo17  807 Jan  6  2022 .profile
drwxr-xr-x  8 odoo17 odoo17 4096 Dec  7 08:47 Odoo-SAAS-Data/
drwxr-xr-x  3 odoo17 odoo17 4096 Dec  5 13:13 common-addons_v15/
drwxr-xr-x  3 odoo17 odoo17 4096 Dec  5 13:13 common-addons_v16/
drwxr-xr-x  3 odoo17 odoo17 4096 Dec  5 13:13 common-addons_v17/
-rwxr-xr-x  1 odoo17 odoo17  739 Dec  5 10:52 docker.sh*
drwxrwxrwx  3 odoo17 odoo17 4096 Jul  7 04:27 dockerv15/
drwxrwxrwx  3 odoo17 odoo17 4096 Jul  7 04:27 dockerv16/
drwxrwxrwx  3 odoo17 odoo17 4096 Nov  8 15:21 dockerv17/
drwxrwxr-x 10 odoo17 odoo17 4096 Dec  5 10:36 odoo/
drwxr-xr-x  7 odoo17 odoo17 4096 Dec  5 13:17 webkul_addons/
root@ip-172-31-37-153:/opt/odoo17#
```

d.) webkul_addons:
The folder will be created inside the directory opt/odoo17. In this directory there will be a folder odoo_saas_kit that contains information about all saas files.

```
root@ip-172-31-37-153:/opt/odoo17/webkul_addons# ll
total 28
drwxr-xr-x  7 odoo17 odoo17 4096 Dec  5 13:17 ./
drwxr-x--- 12 odoo17 odoo17 4096 Dec  5 10:54 ../
drwxr-xr-x 10 odoo17 odoo17 4096 Dec  5 13:25 odoo_saas_kit/
drwxr-xr-x  9 odoo17 odoo17 4096 Dec  5 13:28 saas_kit_backup/
drwxr-xr-x 10 odoo17 odoo17 4096 Dec  5 13:27 saas_kit_custom_plans/
drwxr-xr-x  9 odoo17 odoo17 4096 Dec  5 13:28 saas_kit_trial/
drwxr-xr-x  8 odoo17 odoo17 4096 Dec  5 13:28 wk_backup_restore/
root@ip-172-31-37-153:/opt/odoo17/webkul_addons#
```

Note: The ownership of all created folders in the above steps should be given to the same User from which the Odoo is running.

5. Place the following files in the appropriate folders under the paths as shown below:

a.) Odoo-SAAS-Data/odoo.conf
The file is the default Odoo configuration file for SAAS clients. This file will be used as a reference at the time of Client container creation.

b.) Odoo-SAAS-Data/odoo-template.conf
The file is the default Odoo configuration file for the SAAS template. This file will be used as a reference at the time of Template container creation.

c.) Odoo-SAAS-Data/docker_vhosts/vhosttemplate.txt
The default template for Nginx vhosts. In this file you have to define your SSL certificates so your client instance and saas template can get the SSL certificates.

```
root@ip-172-31-37-153:/opt/odoo17/Odoo-SAAS-Data/docker_vhosts# ll
total 44
drwxr-xr-x 2 odoo17 odoo17 4096 Dec  7 08:47 ./
drwxr-xr-x 8 odoo17 odoo17 4096 Dec  7 08:47 ../
-rw-r--r-- 1 odoo17 odoo17 2267 Dec  5 13:49 db15_templates.simranpal.tk.conf
-rw-r--r-- 1 odoo17 odoo17 2267 Dec  5 13:45 db16_templates.simranpal.tk.conf
-rw-r--r-- 1 odoo17 odoo17 2267 Dec  5 13:30 db17_templates.simranpal.tk.conf
-rw-r--r-- 1 odoo17 odoo17 2223 Dec  6 13:53 now.simranpal.tk.conf
-rw-r--r-- 1 odoo17 odoo17 2243 Dec  7 08:47 ruthless.simranpal.tk.conf
-rw-r--r-- 1 odoo17 odoo17 2264 Dec  5 13:20 vhosttemplate.txt
-rw-r--r-- 1 odoo17 odoo17 1128 Dec  5 13:19 vhosttemplate.txt_bac
-rw-r--r-- 1 odoo17 odoo17 1187 Dec  5 13:19 vhosttemplatehttp.txt
-rw-r--r-- 1 odoo17 odoo17 2226 Dec  5 13:19 vhosttemplatehttps.txt
root@ip-172-31-37-153:/opt/odoo17/Odoo-SAAS-Data/docker_vhosts#
```

d.) Odoo-SAAS-Data/odoo-server.conf
The file is the default Odoo configuration file for the SAAS server. This file will be used as a reference at the time of Server container creation.

Creating Webkul Odoo Docker Images

You need to create webkul odoo images to be used by all Odoo SaaS clients.

First, you need to install the docker setup in your system.

1.) Visit https://docs.docker.com/engine/install/ubuntu/ to know how to install the docker.
2.) Once installed, create a docker image that will be used to launch all the containers.

Note:- All the files are available in the provided zip file

Please keep the following points in mind while creating the Docker image.

A.) User ID & Group ID of Odoo Service User should be same on Host machine & Docker image:

The odoo_saas_kit module maintains a data directory(which includes filestore & sessions) , odoo configuration file and common addons on Host i.e. outside the containers. Appropriate paths are created & mounted for each client at runtime.

Since the files are being shared , these shared folders need to have proper ownership on Host & inside docker containers. So, one needs to ensure the owner (on Host and in containers) have the same user id & group id.

We can check it gid and uid by using command **id {your_odoo_user}**

B.) You need to update the User ID & Group ID in Dockerfile to match the same odoo Service user on the App server.
Refer to /etc/passwd file to check the userid & groupid of the Odoo service user.

```
root@ip-172-31-37-153:~# cat /etc/passwd | grep odoo
odoo17:x:998:998::/opt/odoo17:/bin/sh
root@ip-172-31-37-153:~#
```

C.) Odoo service users should own Dockerfile and other files kept adjacent to Dockerfile. There are some executable files which should be owned by Odoo user in order to run Odoo applications inside containers. The Dockerfile directory should have all the rwx-rwx-rwx permission.

The executables file will be entrypoint.sh and run_odoo.sh

Note:- In case you plan to use remote containers, you would need to configure docker daemon to listen to the main server using the 2375 port.

Command: -
id {your_odoo_user} - Using this command you can get the uid and gid of the odoo user which you can use while running the below mentioned command.

cd {your_docker_file_path}

docker build --build-arg ODOO_USER_UID=* --build-arg ODOO_USER_GID=*** -t {your_docker_image_name:your_docker_image_tag} .**

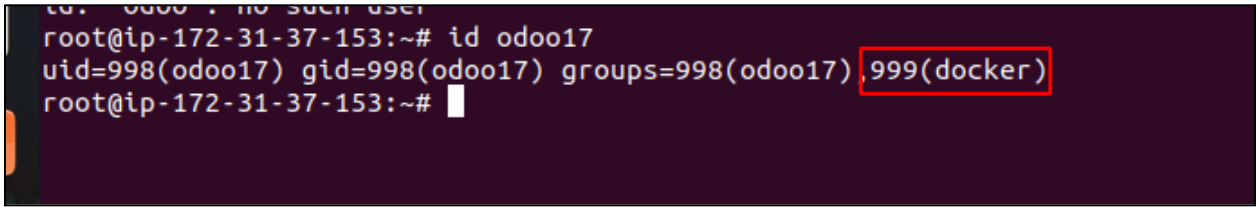
Here * represent UID and GID of Odoo service user we have to write uid and gid of odoo user instead of *****

Please add the period or dot i.e “.” at the end of the above command.

Additional Permissions To Be Given To The Odoo Service User

You need to provide additional permissions to the Odoo Service User

A.) Allow Odoo user to execute docker commands
Because the docker will be controlled by the Odoo application now.
Command- usermod -a -G docker {your_odoo_user}



B.) Allow odoo user to control Nginx as the odoo_saas_kit module will configure Nginx Vhosts at runtime.

Command- Append “{your_odoo_user} ALL=(ALL) NOPASSWD:/usr/sbin/nginx” to /etc/sudoers

```
# Members of the admin group may gain root privileges
%admin ALL=(ALL) ALL

# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL
odoo17 ALL=(ALL)NOPASSWD:/usr/sbin/nginx
odoo17 ALL=(ALL)NOPASSWD:/usr/bin/certbot
# See sudoers(5) for more information on "@include" directives:

@includedir /etc/sudoers.d
root@ip-172-31-37-153:~#
```

C.) Include the Nginx conf location for SAAS
The module maintains all Nginx Vhosts here. So, including the folder in main Nginx configuration will allow dynamically generated Vhosts to have effect.

Add “include {your_docker_vhosts_path}/*.conf;” in your main Nginx Server Block.

```
        # gzip_types text/plain text/css application/json application/javascript text/

        ##
        # Virtual Host Configs
        ##

        include /etc/nginx/conf.d/*.conf;
        include /etc/nginx/sites-enabled/*;
        include /opt/odoo17/Odoo-SAAS-Data/docker_vhosts/*.conf;
}


#mail {
#       # See sample authentication script at:
```

D.) Enable Connectivity with Postgresql Server because the clients & templates will need a Postgresql server. Ensure the connectivity to that Postgresql Server along with login & db creation permissions.

Update postgresql.conf & pg_hba.conf accordingly
In the pg_hba.conf file we have to add IP of docker and main server.

In postgresql.conf file we have to uncomment the listen_address and add '*' in it so that it can listen to all the IP addresses.

```
# TYPE  DATABASE        USER            ADDRESS                 METHOD

# "local" is for Unix domain socket connections only
local   all             all                                     peer
# IPv4 local connections:
host    all             all             127.0.0.1/32            md5
host    all             all             172.17.0.0/16           md5
host    all             all                     l/32            md5
# IPv6 local connections:
host    all             all             ::1/128                 scram-sha-256
# Allow replication connections from localhost, by a user with the
# replication privilege.
local   replication     all                                     peer
host    replication     all             127.0.0.1/32            scram-sha-256
host    replication     all             ::1/128                 scram-sha-256
root@ip-172-31-37-153:~#
```

E.) Create a new user for SAAS with appropriate user permissions & a password to have a separate Odoo user for all SAAS clients. Generally we use the "odoosaas" user.

```
root@ip-172-31-37-153:~# su - postgres
postgres@ip-172-31-37-153:~$ psql
psql (14.10 (Ubuntu 14.10-0ubuntu0.22.04.1))
Type "help" for help.

postgres=# \du
                                   List of roles
 Role name |                         Attributes                         | Member of
-----------+------------------------------------------------------------+-----------
 odoo17    | Superuser, Create role, Create DB                          | {}
 odoosaas  | Create DB                                                  | {}
 postgres  | Superuser, Create role, Create DB, Replication, Bypass RLS | {}

postgres=#
```

Finally ensure that the following paths are properly configured in saas.conf (in odoo_saas_kit/models/lib/)

1.) nginx_vhosts = /opt/odoo17/Odoo-SAAS-Data/docker_vhosts/
Path where Nginx vhosts will be created.

2.) odoo_saas_data = /opt/odoo17/Odoo-SAAS-Data/
Path where all folders/files related to SAAS Clients will be placed/kept.

3.) common_addons_v17 = /opt/odoo17/common_addons_v17

Path where all common addons will be kept/placed. Module odoo_saas_kit will mount this location to every SAAS clients for the purpose of common addons

4.) odoo_image_v17 = odoobywebkul:17.0
Docker Image that must be used for SAAS client creation.

5.) template_odoo_port_v17 = 8817
Port that will be occupied for template containers & it should not be occupied.

6.) odoo-template_v17 = odoo17_template_cont
Main template container

Note:- For Saas_kit_custom plans we have to add template name or other version as well

For ex:-

For version 16 we have to write

odoo_image_v16 = odoobywebkul:16.0

odoo_template_v16 = odoo16_template_cont

common_addons_v16 = /opt/odoo17/common-addons_v16

template_odoo_port_v16 = 8816

template_odoo_lport_v16 = 8826

For version 15 we have to write

odoo_image_v15 = odoobywebkul:15.0

odoo_template_v15 = odoo15_template_cont

common_addons_v15 = /opt/odoo17/common-addons_v15

template_odoo_port_v15 = 8815

template_odoo_lport_v15 = 8825

You can add another remote server to the SAAS kit. And choose to create SAAS Clients on that remote Server.

Setting up Remote Server for SaaS Kit in order to add it SAAS kit

1.) Firstly we add a group using command

groupadd --gid {odoo_user_gid} {your_odoo_user}

```
root@ip-172-31-46-122:~#
root@ip-172-31-46-122:~# cat /etc/passwd | grep odoo
root@ip-172-31-46-122:~# groupadd --gid 998 odoo17
root@ip-172-31-46-122:~# adduser --system --home /opt/odoo17 --shell /bin/bash --uid 998 --gid 998 odoo17
Adding system user `odoo17' (UID 998) ...
Adding new user `odoo17' (UID 998) with group `odoo17' ...
Creating home directory `/opt/odoo17' ...
```

2.) Create odoo System User.

Note: Odoo system user's uid and gid should match the uid and gid of odoo user on the main server.

Command: adduser --system --home /opt/{your_odoo_user} --shell /bin/bash --uid {odoo_user_uid} --gid {odoo_user_gid} {your_odoo_user}

```
root@ip-172-31-46-122:~#
root@ip-172-31-46-122:~# cat /etc/passwd | grep odoo
root@ip-172-31-46-122:~# groupadd --gid 998 odoo17
root@ip-172-31-46-122:~# adduser --system --home /opt/odoo17 --shell /bin/bash --uid 998 --gid 998 odoo17
Adding system user `odoo17' (UID 998) ...
Adding new user `odoo17' (UID 998) with group `odoo17' ...
Creating home directory `/opt/odoo17' ...
```

3.) Add password to the odoo user created and enable Password based SSH authentication:

Command: passwd {odoo_app_user}

```
root@ip-172-31-46-122:~#
root@ip-172-31-46-122:~# passwd odoo17
New password:
Retype new password:
passwd: password updated successfully
```

4.) Now, you need to install docker as Main Server.

Visit https://docs.docker.com/engine/install/ubuntu/ to know how to docker.

5.) After docker installation**, export image which was built on Main Server to Remote Server**

Command (On Main Server):-
docker save {odoo_saas_image}:{odoo_saas_image_tag} > {odoo_saas_image}.tar

Now, Copy tar file from main server to remote server. And load it **on a remote server.**

Command: docker load < {odoo_saas_image}.tar

Alternatively, images can also be pushed to docker hub and can be pulled on remote. Or, if none of the above options work, the image can be rebuilt using the same Dockerfile along with other files present in its folder e.g entrypoint.sh, run_odoo.sh etc.

6.) Allow odoo user to use docker by adding odoo user to docker group.

Command: usermod -a -G docker {your_odoo_user}

7.) You also have to make Remote Docker Daemon Listen on Port 2375 instead of Socket by updating the Service file.

Update ExecStart Option to below in file **/lib/systemd/system/docker.service:-**
ExecStart=/usr/bin/dockerd -H fd:// -H tcp://0.0.0.0:2375

Reload Systemctl Daemon:
systemctl daemon-reload

Restart Docker Service:
service docker restart

NOTE:- This would expose Docker Daemon and ensure that you configure Firewall Rules to block public traffic on this Port i.e 2375

8.) Now, you need to install a few pip3 dependencies.

I. docker (For Docker Python APIs)
II. erppeek (For Odoo managed Tasks)

III. paramiko (For remote SSH connections) In case you plan to have remote containers.
IV. psycopg2 (For connecting postgresql using python).

Use the following command in your command prompt to install the libraries

pip3 install docker erppeek paramiko psycopg2

Note: You might need to install libpq-dev before installing psycopg2

Command: sudo apt install libpq-dev

9.) You also need to Create **Odoo-SAAS-Data(just add odoo.conf, odoo-template.conf within it) and common_addons(including all files/folders)** same as the main server with the same **permissions and ownership.**

10.) We have to install ssh pass in main server

Command: apt install sshpass

Once all steps are performed, you can add this remote server to the saas kit.

NOTE:- Both Main & All your Remote Servers are suggested to be in a local n/w i.e locally reachable from each other using Private Ips.

Setup backup module

This is used to take backup of a database of clients created through the front end.

a.) Install python crontab using

command:- pip3 install python-crontab

b.) Edit the file manage-backup-crons.py on path

opt/odoo17/webkul_addons/wk_backup_restore/models/lib and check the parameter

Backup_Script_Path it is should be same as above mentioned path

opt/odoo17/webkul_addons/wk_backup_restore/models/lib/manage_backup_crons.py

Setup Custom Addons

This is used for back end setup of SSL certificate

a.) Installation of dependencies certbot and python3-certbot-nginx using command

apt install certbot python3-certbot-nginx

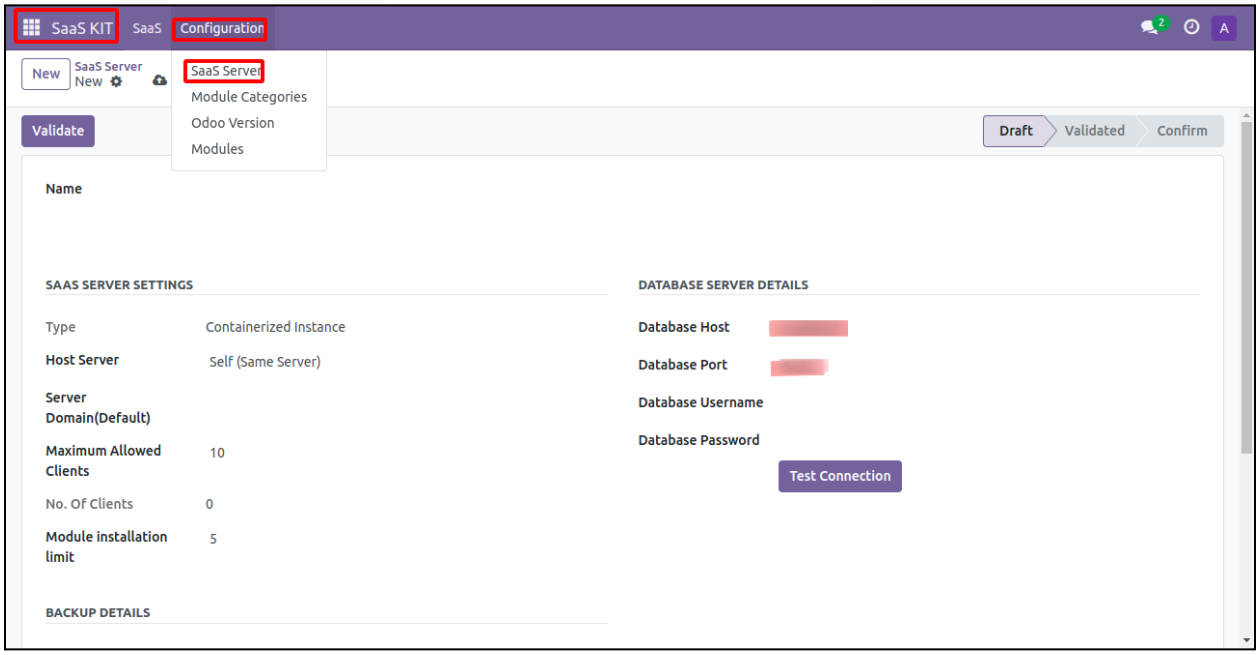
b.) Adding permission to use certbot for odoo user in visudo

{your_odoo_user} ALL =(ALL)NOPASSWD:/usr/bin/certbot

c.) Copy the files of directory vhosts (/opt/odoo17/webkul_addons/odoo-saas-kit/models/lib/vhosts) in directory docker_vhosts (/opt/odoo17/Odoo-SAAS-Data/docker_vhosts)

How To Configure SaaS Server

1.) Configure SaaS Kit module server by adding details of your server to it.



2.) Go to SaaS Kit >> Configuration >> SaaS Server

3.) Enter the **Name** of Server. Select the **Host Server**

- Self (for Same Server)
- Remote Server

4.) Then, enter all the following details in SaaS Server Settings.

a.) Host Server

Select Self(Same Server) as **Host Server** to create and run all the client's instances on the same server which you are using to run your Odoo.

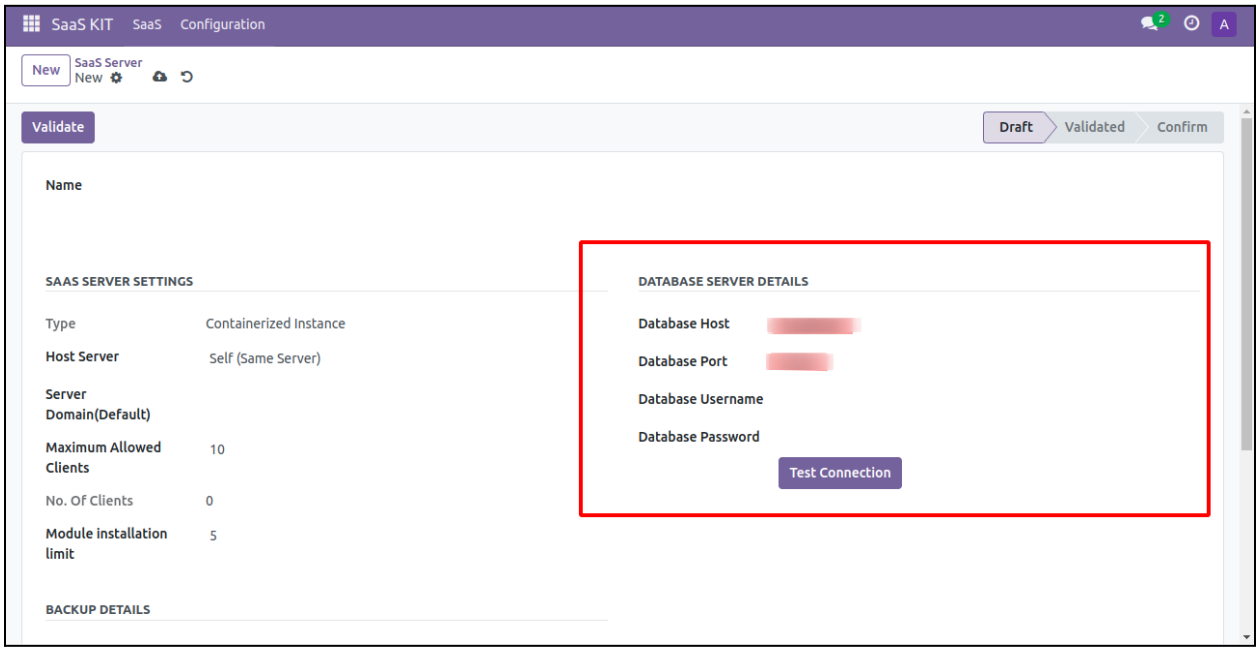
b.) Server Domain(Default)

Enter the Domain name of your server. Ex-(webkul.com, mobikul.com).

c.) Maximum Allowed Client

Set Maximum number of client's instances to be created on this server as per the size and load of your server.

5.) Now, along with the server details you need to enter the database server details to complete the server configuration.



a.) Database Host

Enter the host name on which your database server is running.

b.) Database Port

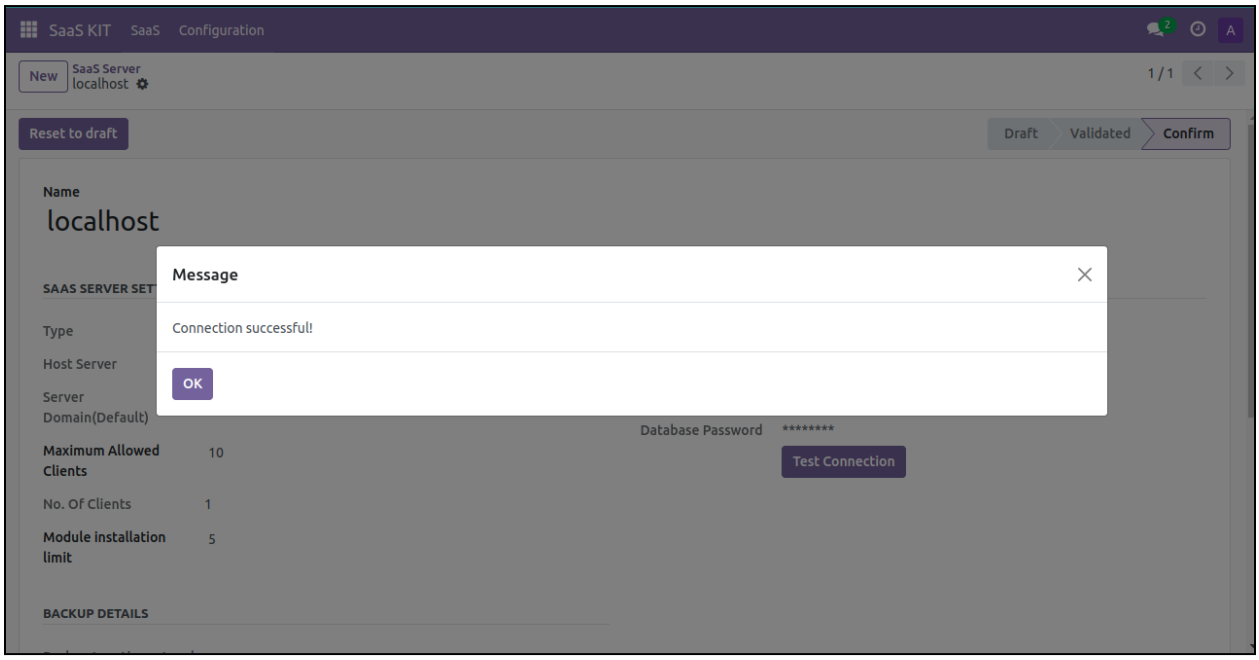
Enter the Port number on which the database server is listening.

c.) Database Username and Database Password

Enter DB username and password for that username.

6.) Once you enter the details, you can test the db connection by clicking on **'Test Connection'**.

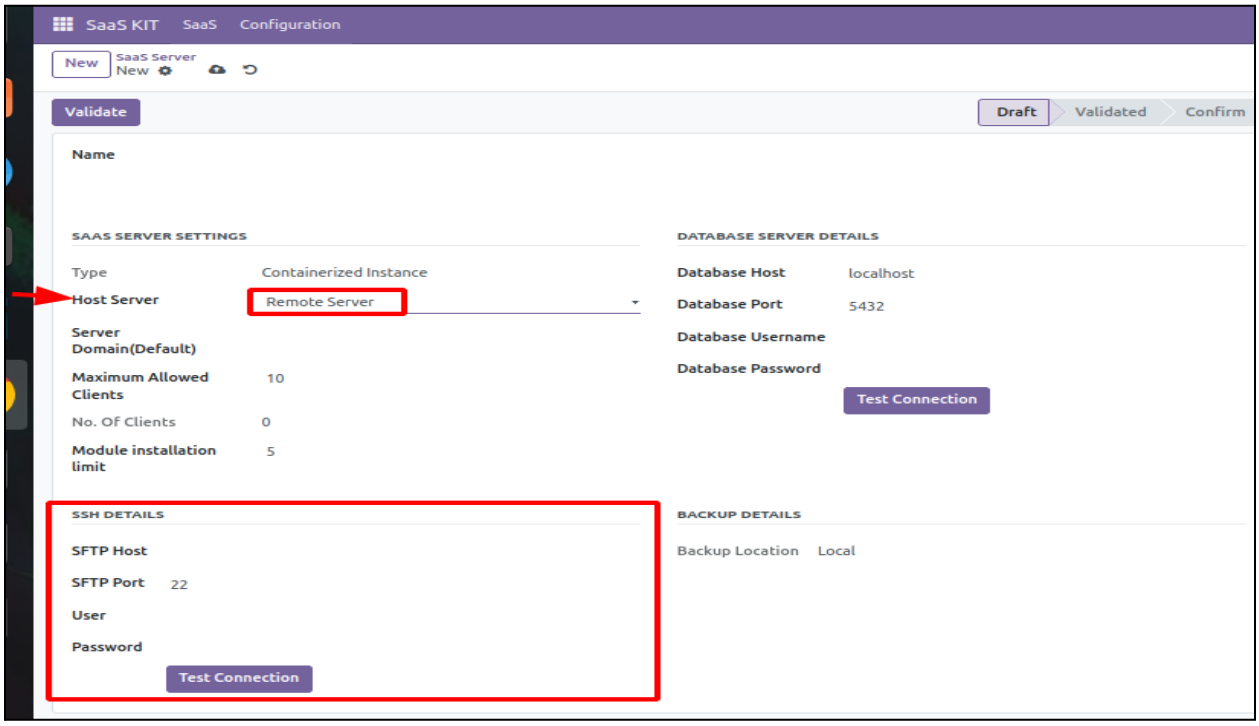
A pop-up with the message **Connection Successful !** will appear on your screen if all the entered details are correct.



7.) Save the details and click on the **'Confirm'** button to complete the configuration.

How To Configure SaaS Remote Server

1.) Create New configuration for the saas server by following the same steps as mentioned previously.



2.) Enter Name of the Server and select **Remote Host** in Host server field.

3.) Then add details in following field:

a.) **Server Domain**
Enter the Domain name of your server. Ex-(webkul.com, mobikul.com).

b.) **Maximum allowed Clients**
Set Maximum number of client's instances to be created on this server as per the size and load of your server.

4.) Now add the SSH details for the remote server.

a.) **SFTP Host**
Enter the local IP address of the remote server.

b.) **SFTP Port**

Enter the port for ssh connection (default 22).

c.) **User and Password**

Enter the username and password for ssh connection.

Also, add database server details as mentioned in previous steps and test DB connection.

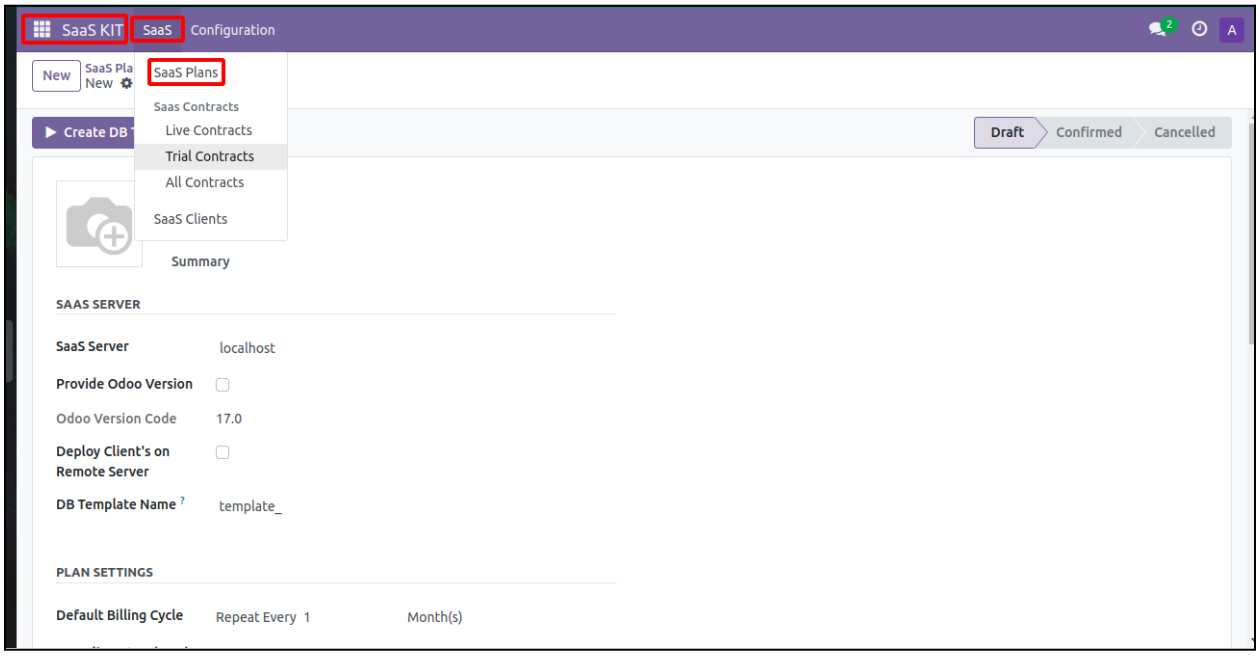
Note:- Never enter localhost here, enter the local/Private IP of the Postgresql Server. You have to update Postgresql config to allow connections from 172.17.0.0/16 CIDR.

5.) Save Details and Click on confirm button to complete configuration.
If there is any error, a message would pop-up.

Creating SaaS Subscription Plans

1.) The plans would allow the clients to create his/her Odoo instance and use the modules mentioned in the plans for their business.

2.) Go to SaaS Kit >> SaaS >> SaaS Plan



3.) Enter the Name of Plan. Add the description for plan in **Summary** if needed.

4.) Add the following details:

a.) SaaS Server

Select a server among the list of servers which you have configured and confirmed before, so that odoo's instances created by this plan can run over that particular server.

b.) DB Template

Enter some unique name for db_template of this plan, if you leave this field blank then it will auto assign a name by the name of plan and a unique number.

c.) Billing Cycle/ Repeat Every

Select time period from (Days, Weeks, Months, Year etc) and enter the value for that time period. This field represents the time period of a billing cycle of this plan.

d.) Number Of Cycles

Enter number of billing cycles you want to sell of odoo instances associated to this plan.

e.) Trial Period(in days)

Enter the number of days for the trial period. If you enter days then the calculation of the billing cycle will start after the mentioned days.

f.) Default Billing Criteria

Select Fixed Rate or Based on the No. of users -

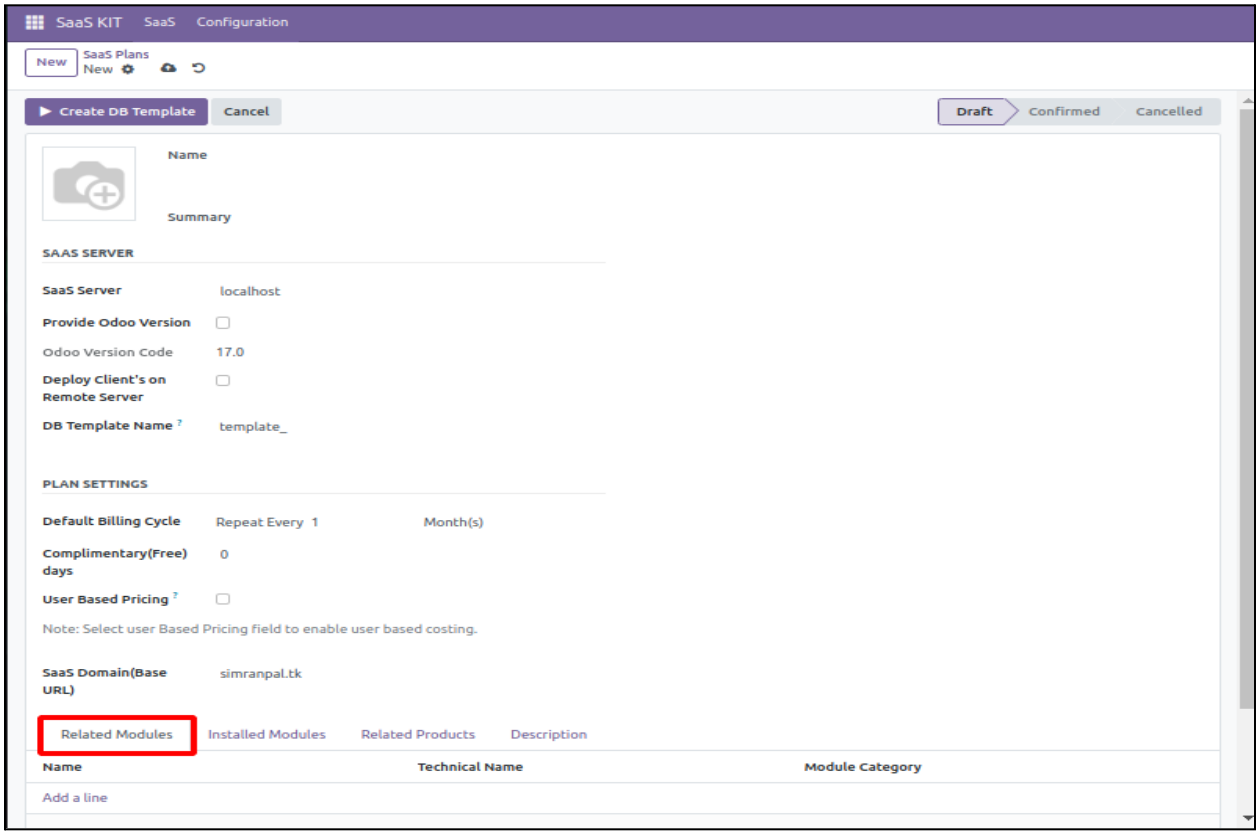
For Fixed Rate: Invoice of a fixed amount will be generated after every billing cycle.

For Based on the No. of users: Invoice is generated according to the number of users created on the customer's instance.

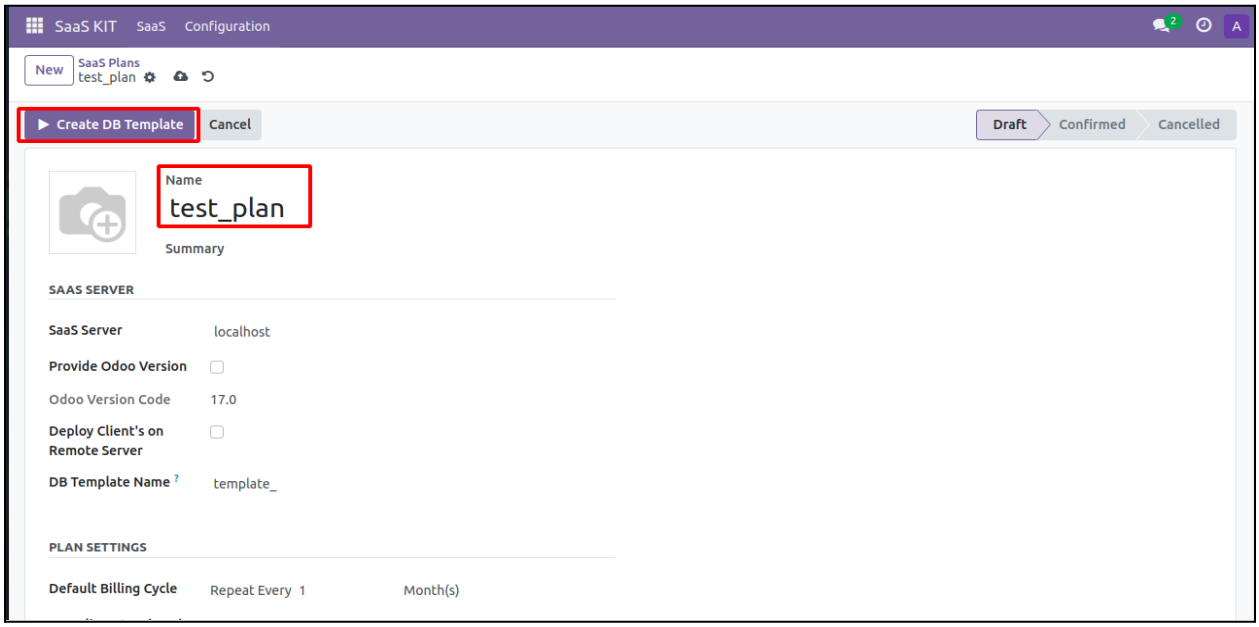
g.) SaaS Domain(Base URL)

Enter domain in this field although it auto fills with the domain name you have entered in server configuration.

5.) Add **Related Modules** in the tab which you want to sell in this plan, only mentioned modules will be installed on the client's instance.

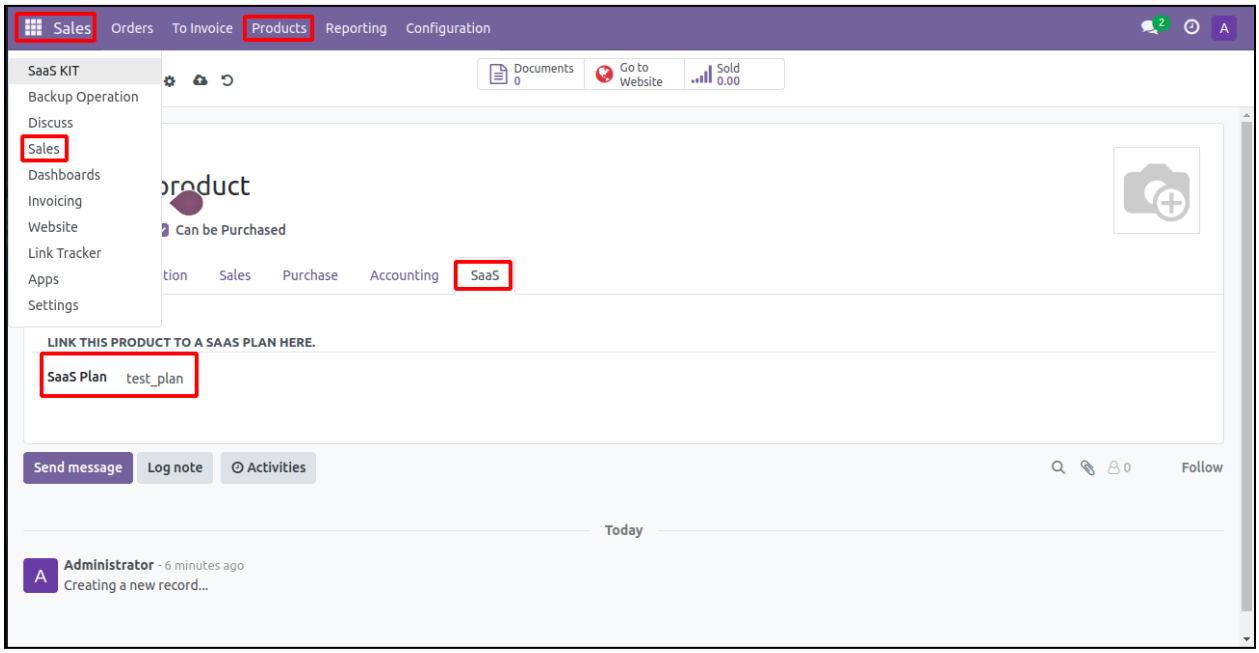


6.) Now click on the Create **DB Template** button. It will create the db template of this plan having all the modules installed which you have listed in **'Related Modules'** section

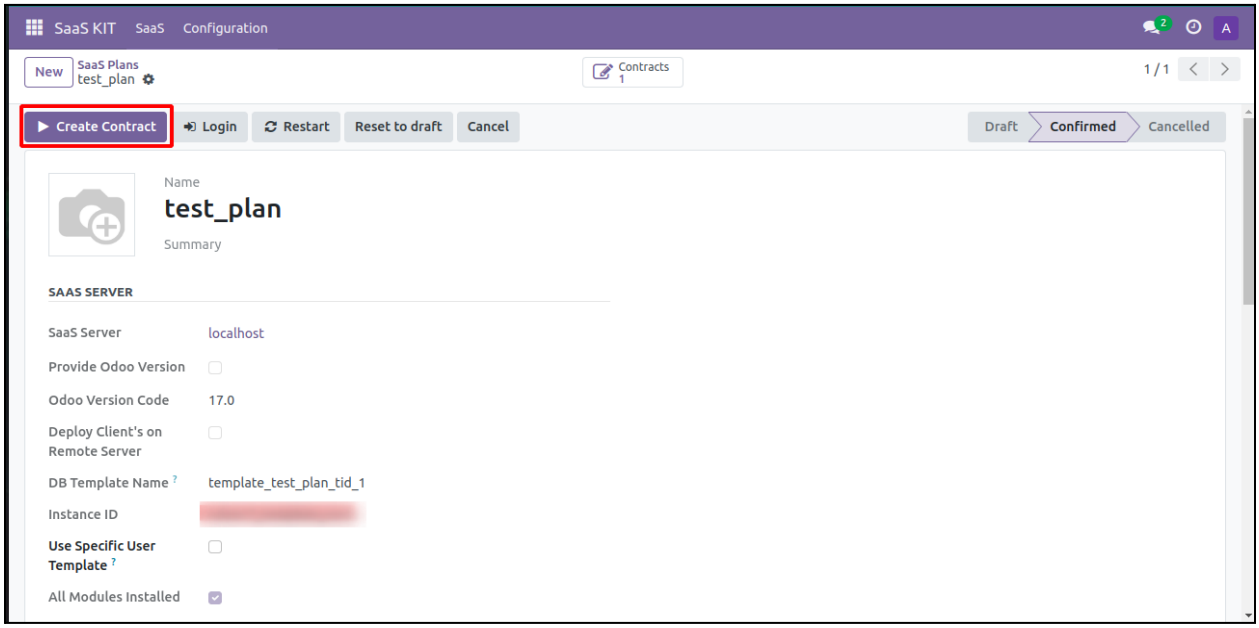


7.) After successful creation of db template. Click on **Create product.**

Note: **When creating the product, select the product type as 'service' and link this product to a SAAS plan here.**

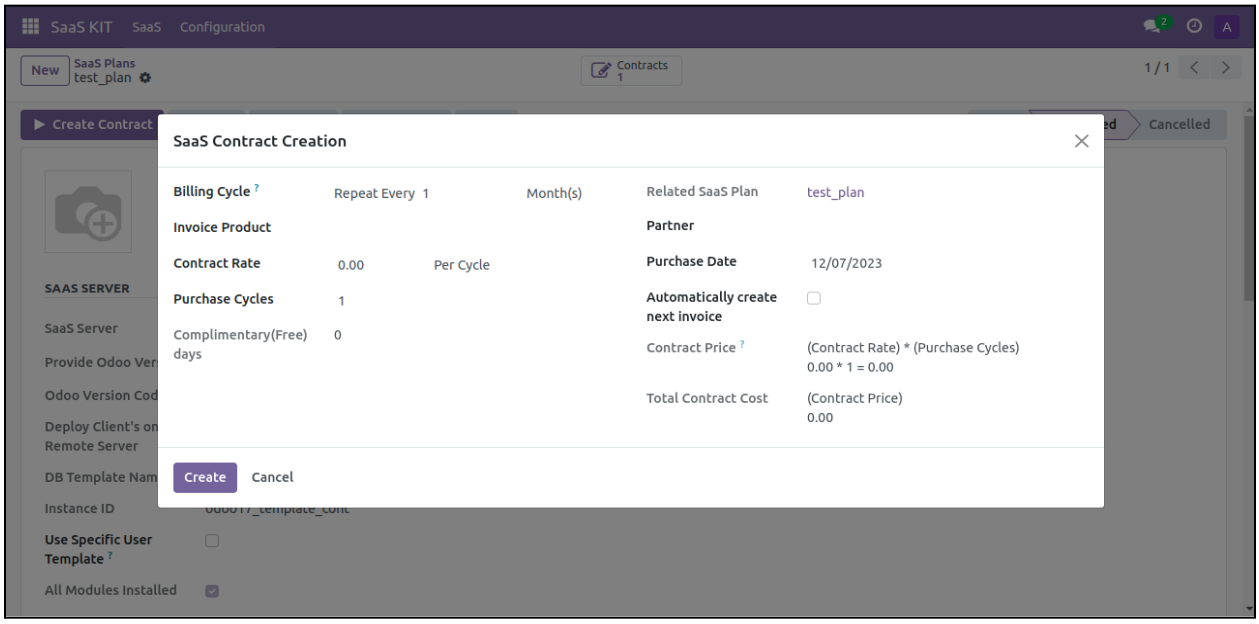


8.) After successful creation of db template. Click on **Create Contract.**

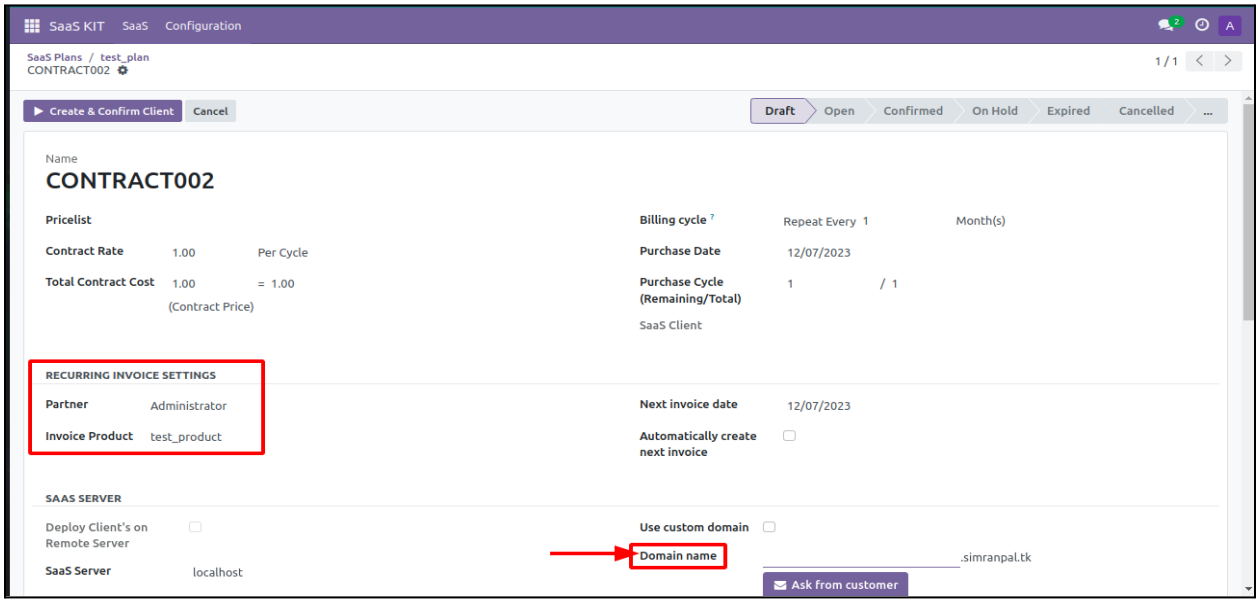


9.) On the pop up that comes up, mention the details required in the wizard for selling the plan.

10.) Select **Partner Name** to whom you want to sell this plan. Enter **contract price** for the plan then click on Create to create the contract of the plan for the selected partner, a contract will be created for the customer.

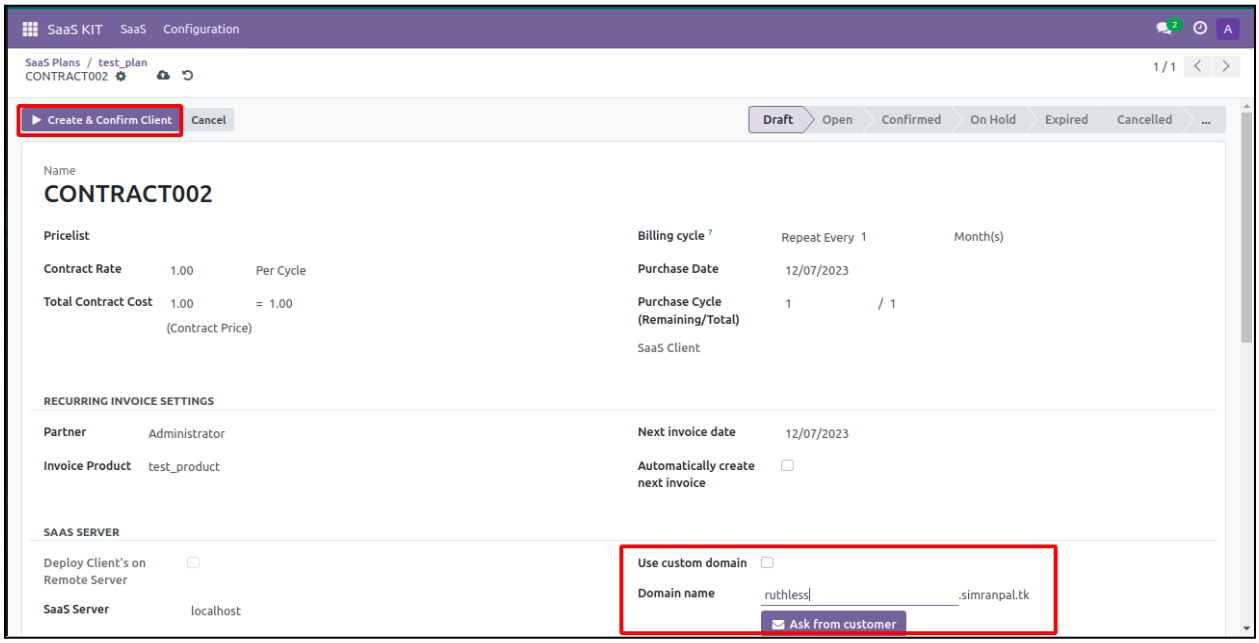


11.) Now, to create an Odoo Saas instance for the particular customer, enter the domain name on which customer's Odoo instance would run.



12.) For entering the domain name, you can either manually enter a unique domain name in **Domain Name** which is not used by any other customer till now.
or you can click on the Ask **from Customer** button which will send a mail to the respective customer in which they will get a link to select a domain name of their choice.

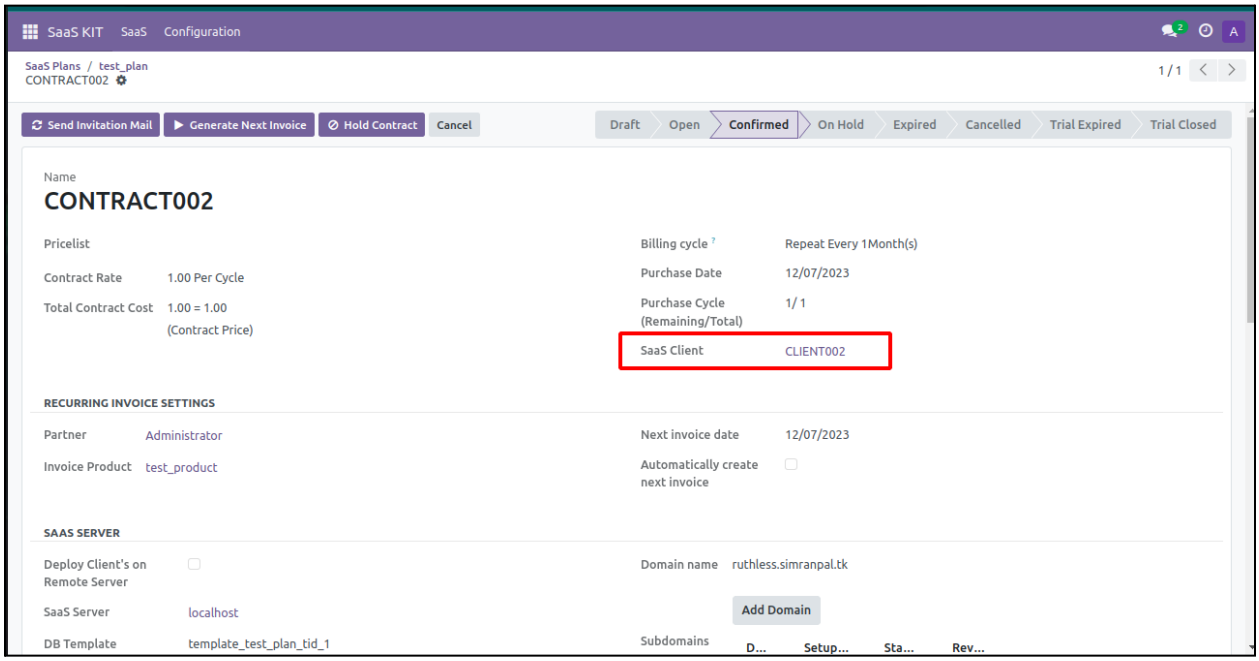
If the customer entered a valid and a unique domain name then that domain will assign to his instance after auto creation of Instance.



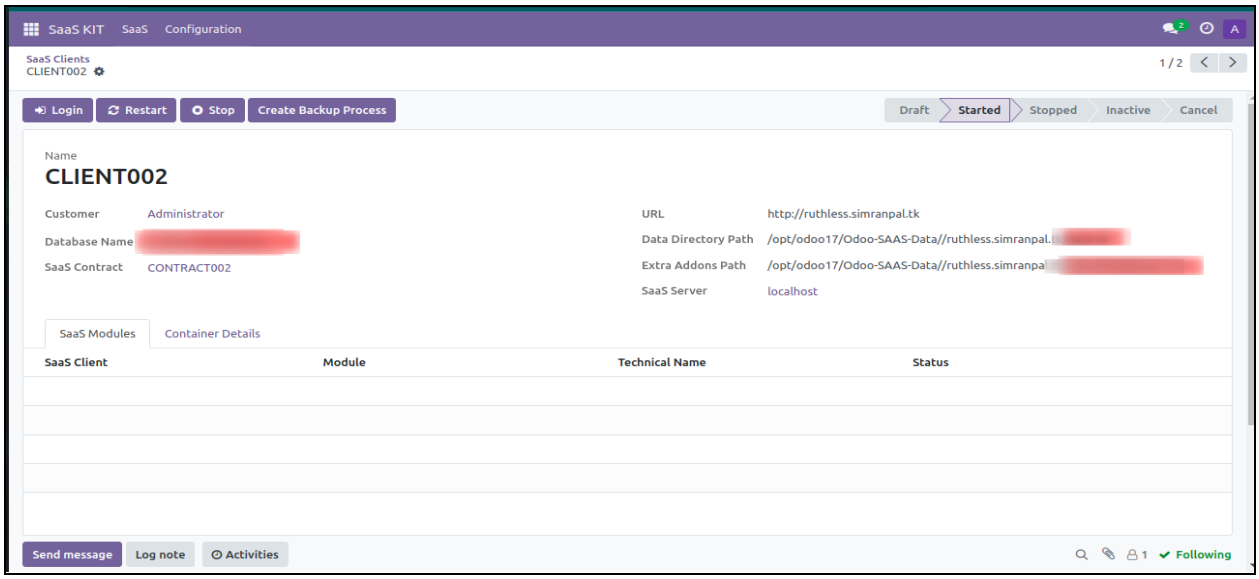
13.) In case of manually entering the domain name, click on **‘Save’** to save the details then click on **‘Create and Confirm Client’** button which will create the odoo instance on selected domain for customers and will create a client record also.

A mail will be sent to the customer having the link to set up its password for odoo and the url to open his odoo.

14.) You can check the details of a client's instance in SaaS Client.



15.) After clicking on the SaaS client, a client record will be shown having all the details of the client.



16.) You can Start , Stop , Restart the customer’s odoo instance with the help of buttons available on the client's record.

17.) You can also login to the customer’s instance by clicking on the Login button on the client's record.

Frequently Asked Questions

1.) Upload modules

There are two possible situations:-

i) Common Modules i.e modules that you want all SAAS clients to use:-

The modules should be placed in a common location which is reachable to all SAAS Clients & Templates.

E.g the location is "/opt/odoo17/common-addons" in your setup.

ii)SAAS Client specific Modules i.e modules that you want specific clients to use:-

The modules should be placed in location only reachable to the specific SAAS Client.
E.g:
The location is "/opt/odoo17/Odoo-SAAS-Data/{client_container_name}/data-dir/addons/17.0"

Replace {client_container_name} with actual container name (client domain without http://. without braces).

Note:- After adding the module, the permissions/ownership should be set to Odoo user i.e "odoo" in your case.

2. Domain Directory Management

The individual virtual hosts file are kept in "/opt/odoo16/Odoo-SAAS-Data/docker_vhosts" folder. One needs to have basic & sufficient knowledge of Nginx in order to make direct changes in this folder otherwise refrain from doing so.

3. Restart the server

One can start/stop the SAAS client server from the Odoo SAAS kit panel. The suggested method is to perform the operation from the server backend i.e. terminal on the server.

Use below command to restart any client

container:-**docker restart {client_container_name}**

Replace {client_container_name} with actual container name (client domain without http://. without braces).

4. Do I need to test, could I create clients and plans and then delete normally, or is there any way to test before working seriously with clients?

We would like to suggest you create a template & a corresponding client just to test and review the procedure & working.

Plan can be deleted only if no contract has been created for that plan. Similarly, the contract can be deleted only if no client has been created for that contract.

Clients (Customers) cannot be deleted.

So, if a customer has been created, then the respective plan and contract cannot be deleted but you can drop its database and container.

5. Where can I find the log file of the SAAS instance?

You are expected to have a basic understanding of Docker & its commands. You can check the logs of any client instance by running below command on your Server.

Command:-

docker exec -it {client_container_name} bash -c "tail -f /var/log/odoo/odoo-server.log"

Replace {client_container_name} with actual container name (client domain without http://. without braces).

6. How to backup and restore SAAS instance?:- Postgres/RDS backup, source code -- from image, filestore :- local script

As the SAAS kit creates databases on the Postgresql Server or Managed Database Server(e.g. RDS). Thus, the backup depends on that.

Since, your saas has postgresql database hosted locally, you can backup the postgresql server accordingly.

In order to have the backup of Admin's data, a separate add-on, **Odoo Database backup** needs to be installed at the Odoo end. This module will be installed and configured at the Admin's Odoo end.

Now, in order to keep the backup of the data of the client's instances, an extra add-on **' Odoo SAAS Kit Backup'** needs to be installed as taking backing of client's instances is not included in the default functionality of the Odoo SAAS Kit.

The SAAS admin can create a backup process at the Odoo backend in order to take a backup of the data of the client's instances.

Please note that in order to use the **'Odoo SAAS Kit backup'** module, the **'Odoo Database Backup module** needs to be installed first at the Odoo end along with the Odoo SAAS Kit.

7. What is the SAAS master password?:-

Every client that gets created by Odoo SAAS Kit has its own Odoo configuration file e.g. /opt/odoo16/Odoo-SAAS-Data/{client_container_name}/odoo-server.conf. That file contains the master password for that client instance specifically.

Replace {client_container_name} with actual container name (client domain without http://. without braces).

Generally, the saas.conf i.e /opt/odoo16/webkul_addons/odoo_saas_kit/models/lib/saas.conf contains the default master password. And every new client instance inherits its master password from this file.

8. How can I restart the process of a specific client?

You can restart the client instance from the Odoo SAAS Kit panel. Rather, we would recommend you to restart the client specific instance/container from the terminal of the main server. You can follow below command as per the need:-

Command :- docker restart container_name

9. What is the scope of Server priority in the Odoo SAAS Kit?

Priority of server is useful when admin is selecting multiple remote servers while creating saas Plan.

Since the module selects any one server among the entered choices that's why we have added priority to it so that the admin can direct the module to select the highest priority server every time.

If the server on the highest priority is not available or full then the module will automatically select the server having the second-highest priority and so on.

Note: 1 is the highest priority, 2 is the second-highest, and so on.