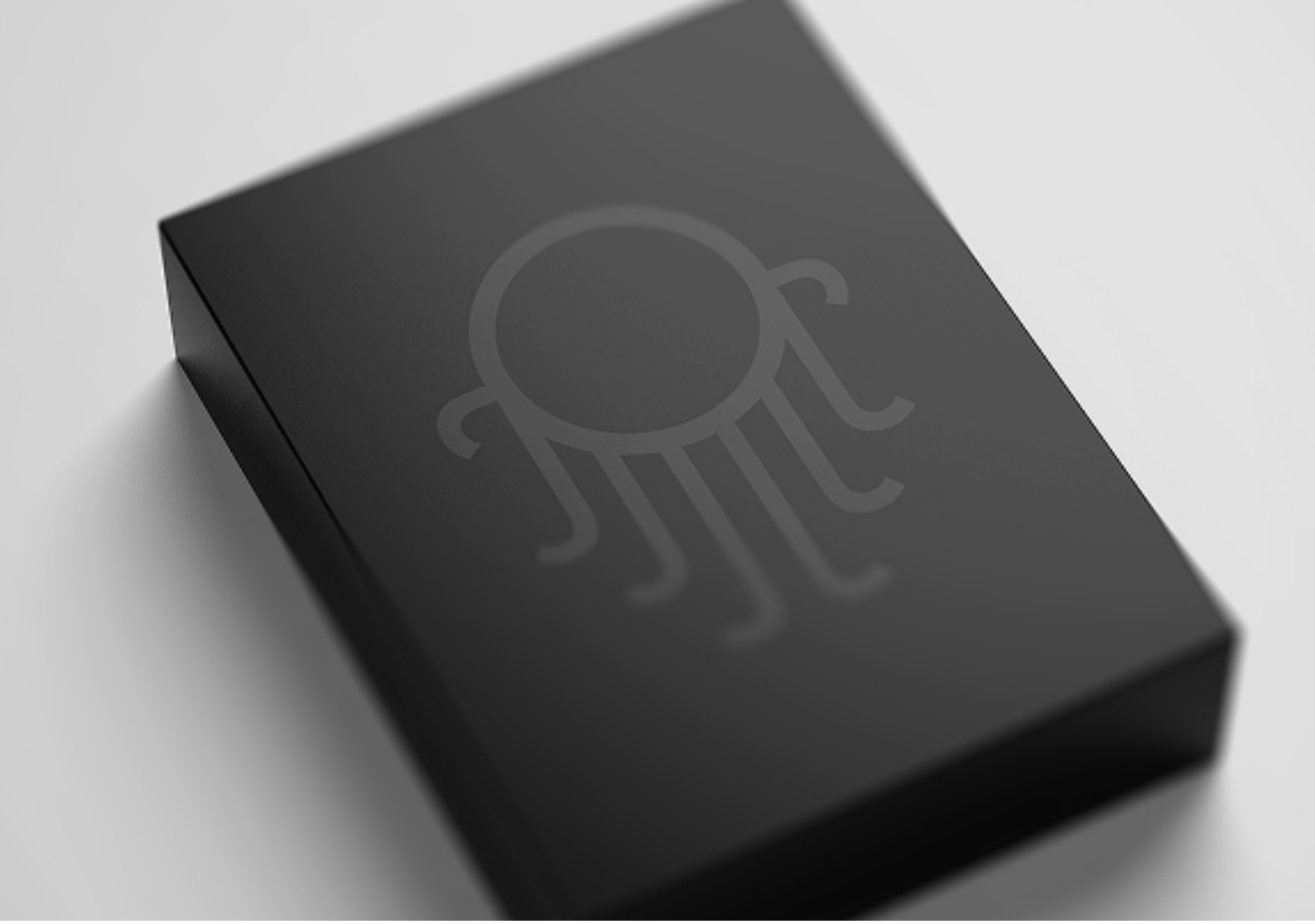

OCTOPUS

Развертывание и запуск Октопус в Red OS

USETECH

Развертывание и запуск Октопус в Red OS

Важно: При установке новой ОС необходимо создать пользователя `octopus/octopus` на этапе установки системы.

Если этот этап пропущен, выполните следующие действия:

Создание пользователя Октопус

Выполняется один раз от пользователя `root`.

Важно: Пользователь Октопус обязательно должен быть с фиксированным UID 1000

Проверьте, не занят ли UID 1000, выполнив команду:

```
getent passwd 1000
```

Если UID 1000 свободен - создаём пользователя:

```
useradd -m -u 1000 -s /bin/bash octopus
echo "octopus:octopus" | chpasswd
usermod -aG wheel octopus
```

Если UID 1000 занят - удаляем старого пользователя:

```
userdel -r <старый_пользователь>
```

Базовая настройка системы

1. Выполните команду для настройки хоста и времени:

```
sudo hostnamectl set-hostname red-octopus
sudo timedatectl set-timezone Europe/Moscow # или свою таймзону
sudo dnf -y update
```

2. Проверьте настройки:

```
hostnamectl
timedatectl
```

3. Установите базовые инструменты:

```
sudo dnf install -y curl vim unzip tar net-tools git lsof firewalld
```

Настройка сетевого экрана

1. Запустите Firewalld:

```
sudo systemctl enable --now firewalld
sudo firewall-cmd --state
```

2. Откройте сервисы Октопус:

```
sudo firewall-cmd --permanent --add-port=80/tcp     # UI
sudo firewall-cmd --permanent --add-port=443/tcp    # UI
sudo firewall-cmd --permanent --add-port=8081/tcp   # Keycloak
```

3. Откройте служебные сервисы:

```
sudo firewall-cmd --permanent --add-port=22/tcp      # SSH
```

4. Настройте маскарадинг (для NAT/форвардинга):

```
sudo firewall-cmd --permanent --add-masquerade
sudo firewall-cmd --reload
sudo firewall-cmd --list-all
```

Настройка сетевого форвардинга

В Red OS по умолчанию активирована строгая сетевая политика, которая блокирует пересылку пакетов между сетевыми пространствами имен (network namespaces). Такое поведение обусловлено принципом безопасной конфигурации (secure-by-default), однако оно препятствует взаимодействию контейнеров Октопус, даже если они находятся в пределах одной Docker-сети. При запуске Docker создает правила межсетевого экрана (iptables), разрешающие прохождение трафика между контейнерами, включая правило ACCEPT all -- anywhere anywhere ctstate RELATED, ESTABLISHED. Однако, поскольку форвардинг изначально заблокирован в ядре системы, установка этого правила оказывается невозможной, что приводит к изоляции сети контейнеров друг от друга.

Для обеспечения нормальной работы контейнеров Октопус необходимо активировать IP-форвардинг на уровне ядра операционной системы.

1. Включите IP forward:

```
echo "net.ipv4.ip_forward = 1" | sudo tee /etc/sysctl.d/99-octopus.conf
sudo sysctl --system
```

2. Проверьте, выполнив команду:

```
sysctl net.ipv4.ip_forward
```

Должен вернуться ответ: `net.ipv4.ip_forward = 1`

3. Сбросьте правила FORWARD для Docker:

```
sudo iptables -P FORWARD ACCEPT
sudo iptables -F FORWARD
```

Установка Docker CE (ветка CentOS 8)

1. Добавьте репозиторий Docker:

```
sudo tee /etc/yum.repos.d/docker-ce.repo >/dev/null <<'EOF'
[docker-ce-stable]
name=Docker CE Stable - $basearch
baseurl=https://download.docker.com/linux/centos/8/$basearch/stable
enabled=1
gpgcheck=1
gpgkey=https://download.docker.com/linux/centos/gpg
EOF
```

2. Обновите кэш:

```
sudo dnf clean all
sudo dnf makecache
```

Настройка SELinux для работы Docker

Red OS по умолчанию использует SELinux в режиме enforcing, что обеспечивает повышенный уровень безопасности, но может блокировать операции Docker с локальными томами. Это связано с политиками безопасности, которые ограничивают монтирование файловых систем контейнерами.

Для нормальной работы Октопус необходимо перевести систему в режим `permissive`, при котором нарушения логируются, но не блокируются. Это безопасный вариант для стендов и внутренних серверов.

1. Переведите SELinux в `permissive` режим:

```
sudo setenforce 0
sudo sed -i 's/^SELINUX=enforcing/SELINUX=permissive/' /etc/selinux/config
getenforce
```

2. Установите Docker в условиях ограничений Red OS:

Ключ `--nobest` необходим для обхода конфликтов версий в репозиториях Red OS 8

```
sudo dnf install -y docker-ce docker-ce-cli containerd.io docker-compose-plugin --nobest
```

3. Запустите Docker:

```
sudo systemctl enable --now docker
docker --version
docker compose version
```

Настройка прав доступа к Docker для пользователя Октопус

1. Добавьте пользователя Octopus в группу docker:

```
sudo getent group docker || sudo groupadd docker
sudo usermod -aG docker octopus
sudo chgrp docker /var/run/docker.sock
sudo chmod 660 /var/run/docker.sock
```

2. Зафиксируйте права через `systemd`:

```
sudo mkdir -p /etc/systemd/system/docker.socket.d
sudo tee /etc/systemd/system/docker.socket.d/10-group.conf >/dev/null <<'EOF'
[Socket]
SocketMode=0660
SocketGroup=docker
EOF
```

3. Перезагрузите службы:

```
sudo systemctl daemon-reload
sudo systemctl restart docker.socket docker
```

4. Проверьте установку:

```
newgrp docker
docker run --rm hello-world
```

Ожидаемый ответ: `Hello from Docker! This message shows that your installation appears to be working correctly.`

5. Проверьте, что Docker выставил свои правила `iptables`:

```
sudo iptables -L FORWARD -n -v | grep ACCEPT
```

Проверка автозапуска

1. Убедитесь, что после перезагрузки сервера сервисы автоматически стартуют самостоятельно:

```
sudo systemctl enable docker
sudo firewall-cmd --runtime-to-permanent >/dev/null 2>&1
sudo reboot
```

2. Перезагрузите:

```
systemctl status --no-pager docker
docker ps
sudo firewall-cmd --list-ports
getenforce
```

Подготовка к установке Октопус

1. Создайте рабочую директорию:

```
sudo mkdir -p /home/octopus/octopus
sudo chown -R octopus:octopus /home/octopus
cd /home/octopus
```

2. Распакуйте дистрибутив:

```
unzip octopus-update-1.30.35.zip
```

После разархивирования получаем структуру каталогов: `octopus.deployment`, `octopus.images`

3. Переименуйте каталог `octopus.deployment` в Октопус:

```
mv octopus.deployment octopus
```

Структура каталогов:

```
/home/octopus/
  ├── octopus/
  │  ├── docker-compose.yml
  │  ├── .env
  │  ├── scripts/
  │  ├── configs/
  │  └── ...
  └── octopus.images/
```

Импорт Docker-образов

Каталог `octopus.images` содержит предсобранные Docker-образы (*.tar), которые следует импортировать, до запуска `docker-compose`.

1. Перейдите в рабочую папку:

```
cd /home/octopus/octopus
```

2. Перейдите в папку с техническими скриптами:

```
cd /home/octopus/octopus/scripts
```

3. Импортируйте оффлайн образы:

```
./import-images.sh /home/octopus/octopus.images
```

Ожидаемый вывод:

```
[octopus@red-octopus scripts]$ ./import-images.sh /home/octopus/octopus.images
Importing image from /home/octopus/octopus.images/clickhouse_clickhouse-server_23.9.3.12-alpine.tar
Loaded image: clickhouse/clickhouse-server:23.9.3.12-alpine
Importing image from /home/octopus/octopus.images/grafana_grafana.tar
Loaded image: grafana/grafana:latest
```

4. Проверьте загруженные образы:

```
docker images | grep -E 'octopus|keycloak|clickhouse|zookeeper|kafka' || true
```

Запуск Октопус

1. Перейдите в рабочую папку:

```
cd /home/octopus/octopus
```

2. Запустите стек:

```
docker compose down -v || true
docker compose up -d --pull never
docker compose ps
```

Ожидание занимает примерно 10 минут от старта.

3. Проверьте состояние:

```
docker ps --format '{{.Names}}\t{{.Status}}'
docker compose logs --tail=60 --no-log-prefix | sed -n '1,200p'
```

4. Пропишите стек в автозагрузку.

Убедитесь, что все корректно:

```
sudo systemctl is-active docker
sudo usermod -aG docker octopus
```

Настройка автозагрузки

1. Создайте systemd-сервис:

```
sudo tee /etc/systemd/system/octopus-stack.service >/dev/null <<'EOF'
[Unit]
Description=Octopus stack (Docker Compose)
After=docker.service network-online.target NetworkManager-wait-online.service
Wants=docker.service network-online.target
```

```
ConditionPathExists=/home/octopus/octopus/docker-compose.yml
[Service]
Type=oneshot
User=octopus
Group=octopus
WorkingDirectory=/home/octopus/octopus
Environment=PWD=/home/octopus/octopus
Environment=COMPOSE_PROJECT_NAME=octopus
ExecStartPre=/bin/bash -lc 'test -S /var/run/docker.sock'
ExecStartPre=/usr/bin/docker --version
ExecStartPre=/usr/bin/docker info
ExecStartPre=/usr/bin/docker compose version
ExecStartPre=/bin/bash -lc '/usr/bin/docker compose -p octopus config >/dev/null'
ExecStart=/usr/bin/docker compose -p octopus up -d --pull never
ExecStop=/usr/bin/docker compose -p octopus down --remove-orphans
RemainAfterExit=yes
TimeoutStartSec=900
TimeoutStopSec=120
Restart=on-failure
RestartSec=10
StandardOutput=journal
StandardError=journal
[Install]
WantedBy=multi-user.target
EOF
```

2. Активируйте изменения:

```
sudo systemctl daemon-reload
sudo systemctl enable --now octopus-stack
systemctl status --no-pager octopus-stack
```

3. Перезагрузите систему:

```
sudo reboot
```

Ожидайте около 10 минут и проверьте веб-интерфейс.

4. Выполните проверку статуса и работу контейнеров:

```
systemctl status --no-pager octopus-stack
docker ps --format '{{.Names}}\t{{.Status}}' | sort
```

Ожидаемый вывод:

```
[octopus@red-octopus vmware-tools-distrib]$ systemctl status --no-pager octopus-stack
docker ps --format '{{.Names}}\t{{.Status}}' | sort
● octopus-stack.service - Octopus stack (Docker Compose)
     Loaded: loaded (/etc/systemd/system/octopus-stack.service; enabled; preset: disabled)
    Drop-In: /usr/lib/systemd/system/service.d
             └─10-timeout-abort.conf
     Active: active (exited) since Fri 2025-11-07 13:15:14 MSK; 14min ago
   Main PID: 1306 (code=exited, status=0/SUCCESS)
        CPU: 2.825s

ноя 07 13:14:31 red-octopus docker[1322]:  Container agent.hyperv  Started
ноя 07 13:14:32 red-octopus docker[1322]:  Container agent.basis  Started
ноя 07 13:14:33 red-octopus docker[1322]:  Container agent.zabbix  Started
ноя 07 13:14:42 red-octopus docker[1322]:  Container agent.brest  Started
ноя 07 13:14:44 red-octopus docker[1322]:  Container agent.zvirt  Started
ноя 07 13:14:46 red-octopus docker[1322]:  Container agent.vcenter  Started
ноя 07 13:14:46 red-octopus docker[1322]:  Container api  Started
ноя 07 13:14:46 red-octopus docker[1322]:  Container ui  Starting
ноя 07 13:15:14 red-octopus docker[1322]:  Container ui  Started
ноя 07 13:15:14 red-octopus systemd[1]: Finished octopus-stack.service - Octopus stack (Docker Compose).
agent.basis    Up 15 minutes
agent.brest    Up 15 minutes
agent.hyperv    Up 15 minutes
agent.rosplatforma    Up 15 minutes
agent.vcenter    Up 15 minutes
agent.zabbix    Up 15 minutes
agent.zvirt    Up 15 minutes
analysis    Up 16 minutes
api    Up 15 minutes
grafana    Up 16 minutes
keycloak    Up 16 minutes
octopus.clickhouse    Up 16 minutes
octopus-kafka    Up 17 minutes
octopus.postgres    Up 16 minutes
orchestrator    Up 16 minutes
prometheus    Up 16 minutes
pushgateway    Up 16 minutes
repository    Up 16 minutes
rsyslog    Up 17 minutes
topology.worker    Up 16 minutes
ui    Up 14 minutes
zookeeper    Up 16 minutes
```