Подробно о NiceOS V

1. Введение

NiceOS V — это специальная редакция дистрибутива **Nice OS**, предназначенная для использования в виртуальных машинах, контейнерах и облачных инфраструктурах. Она создавалась с нуля под нужды автоматизированных развёртываний, CI/CD-пайплайнов, шаблонов облачных образов (например, OVA, QCOW2, AMI) и виртуализированных тестовых сред.

В отличие от универсальных дистрибутивов общего назначения, **NiceOS V** изначально минималистична, не содержит графических компонентов, не тянет за собой среды рабочего стола и поставляется в виде лёгких и быстрых для установки ISO- или OVA-образов. Её главная задача — обеспечить максимально чистую, быструю и воспроизводимую установку в автоматизированной среде.

Философия проектирования

NiceOS V следует принципу: «ничего лишнего — только то, что необходимо для стабильной виртуальной среды». Это означает:

- Нет графики, только консоль (tty, systemd).
- Нет лишних демонов — только sshd, systemd-udevd, networking.
- Все настройки — через файлы, CLI и конфигурацию Kickstart/JSON.
- Образы можно предсобрать заранее, добавив нужные пакеты, скрипты и SSH-ключи.
- Прямая поддержка кастомных ядровых параметров, метаданных от гипервизора, и загрузки конфигураций с удалённых серверов.

2. Зачем нужна отдельная VM-сборка?

Развёртывание операционных систем в виртуальных средах требует совершенно иного подхода по сравнению с классическими дистрибутивами. Там, где пользовательский интерфейс, мультимедиа и графическая оболочка необходимы для

настольных решений, в виртуализированной инфраструктуре всё это становится балластом. Специальная сборка **NiceOS V** разработана с учётом именно этих условий.

Фундаментальные отличия от настольных и серверных редакций

Редакции для десктопов и серверов обычно содержат полный стек пользовательских и системных компонентов, даже если пользователь этого не требует. Это может включать графическую подсистему, библиотеки для GUI-приложений, дисплей-менеджеры и сопутствующие фоновые службы. **NiceOS V** лишена всего этого по замыслу. Вот ключевые отличия:

- **Нет графических зависимостей.** В дистрибутиве отсутствуют Qt, GTK, X11, Wayland и любые связанные с ними пакеты. Это не просто сокращает размер образа — это исключает возможность их случайной загрузки или эксплуатации.
- **Минималистичная загрузочная цепочка.** Используются упрощённые версии загрузчиков (например, без поддержки меню выбора ядра или графических splash-экранов), что ускоряет запуск и упрощает интеграцию с виртуальными BIOS и UEFI.
- **Нулевая поддержка окружений рабочего стола.** Не предусмотрены ни GDM, ни SDDM, ни LightDM. Отсутствуют оконные менеджеры, сессии и всё, что связано с пользовательским GUI.
- **Меньше сервисов по умолчанию.** Стартовая система включает только строго необходимые службы: сетевую инициализацию, SSH, управление журналами и базовую защиту. Всё остальное пользователь настраивает сам через postinstall-скрипты или Ansible.
- **Полное отсутствие мультимедиа-подсистем.** Нет PulseAudio, ALSA, PipeWire, видеодрайверов и даже шрифт-рендеринга. Никаких консольных интерфейсов типа `dialog`, `whiptail` — только голая TTY.

Ключевые преимущества для виртуальных машин

Такой подход обеспечивает преимущества, критичные для эксплуатации в автоматизированных и облачных сценариях:

- **Существенное уменьшение размера дистрибутива.** Благодаря строгой минимизации компонентов, ISO-образ занимает значительно меньше места, чем классические серверные аналоги. Это особенно важно при тиражировании

шаблонов и экономии хранилища.
- **Оптимизация под автоматизированное развёртывание.** Благодаря интеграции с инструментами командной строки и JSON-конфигурациями, установка NiceOS V полностью контролируется извне. Это делает её идеальным кандидатом для CI/CD, развёртывания в облаке и виртуальных тестовых сред.
- **Повышенный уровень безопасности.** Отсутствие GUI, графических библиотек и оконных систем не только снижает уязвимость, но и делает систему менее подверженной эксплуатации изолированных компонентов через локальные уязвимости. Меньше кода — меньше потенциальных точек входа.
- **Быстрота запуска и предсказуемость поведения.** Система загружается практически мгновенно, так как отсутствует фаза запуска графики, демонстрации splash-экранов и инициализации многочисленных подсистем. Это повышает стабильность в гипервизорах и снижает нагрузку на хостовую систему.
- **Упрощённое обслуживание и сопровождение.** Меньше пакетов означает меньше обновлений, меньше конфликтов зависимостей, быстрее тестирование и выше вероятность успешного автоматического обновления в шаблонных средах.

Точечная настройка под гипервизоры и облачные среды

В отличие от универсальных решений, NiceOS V предусматривает точечную оптимизацию под гипервизоры: отключение udev-правил, настройка автоматического получения MAC-адресов, минимизация логирования в `/var/log`, предустановленные метки и шаблоны cloud-init. Также обеспечена корректная работа с виртуальными драйверами, такими как virtio, hv_storvsc, vmxnet и др.

Это делает дистрибутив особенно подходящим для развёртывания на:

- Proxmox VE и VirtualBox — как локальная тестовая среда.
- VMware ESXi — как платформа для корпоративных решений.
- KVM/QEMU — для облаков с OpenStack, OpenNebula, Virt-Manager.
- AWS, Yandex Cloud, Google Cloud — через экспорт в AMI/QCOW2/OVA.

За счёт всего вышеперечисленного, **NiceOS V** — это не просто облегчённая версия, а самостоятельный продукт, ориентированный на виртуализированную инфраструктуру. Его архитектура — это результат точного отбора необходимых компонентов и системных механизмов для бесперебойной работы в условиях

масштабирования и автоматизации.

3. Архитектура установщика Nice OS

Сердцем специальной VM-сборки **NiceOS V** является собственный консольный установщик, разработанный с прицелом на автоматизацию и полное отсутствие графической оболочки. Он предоставляет простой, расширяемый и управляемый через параметры командной строки способ развёртывания операционной системы в виртуальных средах и контейнерах.

CLI как единственная точка входа

Установщик полностью консольный — для запуска и управления используется только командная строка. Такой подход не только упрощает автоматизацию, но и делает возможным использование дистрибутива в headless-средах, где доступен только терминал (например, через VNC, serial console или cloud-init).

Вызов установщика выполняется через утилиту `niceos-installer`, которая по сути является точкой входа в скрипт `niceos_installer/main.py`. Этот компонент отвечает за предварительную обработку аргументов и выбор сценария установки.

Компонент `main.py`: управление установкой

Файл `niceos_installer/main.py` реализует основной CLI-интерфейс установщика. Его задача — принять аргументы пользователя, провести их валидацию и передать управление соответствующему установочному механизму.

- **Проверка параметров:** валидируется тип образа (`iso`, `qcow2`, `raw`), путь к рабочей директории, релизная версия системы (например, `5.2`).
- **Выбор инсталлятора:** в зависимости от формата образа может быть вызван ISO-инсталлятор или универсальный (generic) режим, который работает без предварительно собранного ISO-файла.
- **Интеграция с CLI-инструментами:** все опции могут быть переданы в виде аргументов командной строки, что делает установщик идеальным для использования в скриптах и CI-средах.

Компонент `installer.py`: ядро логики установки

Установочный движок реализован в файле `niceos_installer/installer.py`. Это основной исполнительный модуль, который отвечает за пошаговую установку системы с момента запуска до полного завершения. Он обрабатывает конфигурацию, выполняет подготовку и осуществляет все ключевые действия по установке.

Основные этапы установки:

1. **Обработка конфигурации Kickstart/JSON.** Считываются пользовательские параметры, включая разметку диска, список пакетов, сетевые настройки, SSH-ключи, hostname и скрипты.
2. **Подготовка дисков.** Создание таблицы разделов (GPT/MBR), разметка LVM, файловых систем и точек монтирования в соответствии с конфигурацией.
3. **Установка базовой системы.** Установка необходимых RPM-пакетов в целевую файловую систему, настройка chroot-среды, копирование системных файлов и конфигураций.
4. **Настройка загрузчика.** Установка GRUB или EFI-загрузчика, генерация `grub.cfg`, настройка ядра и initramfs для корректной загрузки в VM-среде.
5. **Постустановочные задачи.** Выполняется настройка дополнительных компонентов: открытие доступа по SSH, импорт Docker-образов, применение Ansible-плейбуков, настройка локали, паролей и hostname.

Все эти действия происходят последовательно, с возможностью вывода логов на консоль или в лог-файл. В случае ошибок установка прерывается с кодом завершения, что делает возможным использование установщика в автоматизированных пайплайнах.

Модульность и расширяемость

Один из ключевых принципов установщика Nice OS — модульность. Все дополнительные функции (настройка сети, SSH, генерация machine-id и т.д.) реализованы в виде Python-модулей в каталоге `niceos_installer/modules/`. Это позволяет разработчикам легко добавлять новые функции или изменять существующие шаги без необходимости менять основную логику установщика.

Интеграция с Docker и сборкой образов

Установщик также интегрируется со скриптом `create-image-util`, который позволяет запускать установку внутри Docker-контейнера и собирать ISO-, OVA- или QCOW2-образы автоматически. Таким образом, можно настроить процесс сборки дистрибутива так, чтобы он включал все нужные пакеты, конфигурации и даже внешние зависимости (например, Docker-образы или Ansible-скрипты).

4. Конфигурация установки: Kickstart

В основе автоматической установки **NiceOS V** лежит декларативная конфигурация в формате JSON, часто называемая **Kickstart-файлом**. Такой подход позволяет задать параметры установки заранее и разворачивать систему без участия пользователя — быстро, последовательно и без ошибок.

Конфигурация может быть встроена непосредственно в ISO-образ, передана через URL (например, при загрузке через PXE), либо подключена через виртуальный диск. Благодаря этому, установщик NiceOS может работать в полностью автоматическом режиме, что идеально подходит для массового тиражирования образов.

Поддерживаемые возможности конфигурации

JSON-файл, описывающий установку, может содержать десятки параметров, сгруппированных по смыслу: от базовой разметки дисков до сетевых настроек, списка RPM-пакетов и пользовательских скриптов. Некоторые из ключевых возможностей:

- **Разметка диска:** создание таблиц разделов (MBR или GPT), указание точек монтирования, размеров томов, форматов файловых систем (ext4, xfs и др.).
- **Выбор пакетов:** установка минимального набора (например, coreutils, bash, systemd) или кастомного списка, включая Docker, Python, Ansible и прочее.
- **Сетевые настройки:** можно задать статический IP, DNS, шлюз, hostname, имя интерфейса, MTU и даже активацию DHCP при старте.
- **Пользовательские скрипты:** запуск shell- или Python-скриптов до и после установки пакетов (например, для инициализации системы, добавления SSH-ключей, настройки пользователей).

Ключевые параметры JSON-конфигурации

Вот некоторые из наиболее часто используемых параметров, которые могут быть включены в `sample_ks.cfg`:

- `"root_ssh_key"` — публичный SSH-ключ, который будет автоматически добавлен в `/root/.ssh/authorized_keys` для прямого доступа после установки. Очень удобно для headless-развёртываний и CI-серверов.
- `"hostname"` — задаёт имя хоста (может быть как фиксированным, так и генерируемым).
- `"postinstall_scripts"` — список скриптов (или ссылок на них), которые выполняются в установленной системе. Используется для любой постнастройки: установка пакетов из внешних источников, настройка фаервола, развертывание служб.
- `"ansible_playbook"` — путь к локальному YAML-файлу или URL с плейбуком Ansible, который будет применён после базовой установки. Позволяет встроить DevOps-логику прямо в установку.
- `"packages"` — список RPM-пакетов, которые должны быть установлены, можно включать группы или конкретные имена.

Пример конфигурационного файла и его разбор

Ниже представлен фрагмент примера из `sample_ks/sample_ks.cfg` с пояснениями:

```
{
  "version": "5.2",
  "hostname": "vm-niceos-node01",
  "root_ssh_key": "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQC...",
  "disk": {
    "type": "gpt",
    "layout": [
      { "mount": "/", "size": "20G", "fs": "ext4" },
      { "mount": "/var", "size": "10G", "fs": "xfs" },
      { "mount": "swap", "size": "2G" }
    ]
  },
  "network": {
    "interface": "eth0",
    "dhcp": true
  },
  "packages": [
    "bash", "coreutils", "openssh-server", "docker", "iptables"
  ],
  "postinstall_scripts": [
    "echo 'PermitRootLogin yes' >> /etc/ssh/sshd_config",
```

```
    "systemctl enable sshd",
    "systemctl enable docker"
  ],
  "ansible_playbook": "/root/bootstrap.yml"
}
```

Разбор:

- `"version"` — определяет релиз, с которым устанавливается система (например, `5.2`).
- `"hostname"` — система после установки получит имя `vm-niceos-node01`.
- `"disk.layout"` — определяет схему разметки диска: основной раздел `/` (20 ГБ), отдельный том под `/var` (10 ГБ, с использованием XFS), а также swap-раздел.
- `"packages"` — в систему будут установлены только необходимые пакеты, включая Docker и openssh-server.
- `"postinstall_scripts"` — сразу после установки пакетов выполняются команды, которые включают SSH-доступ для root, а также активируют Docker и sshd.
- `"ansible_playbook"` — в финальной стадии будет запущен плейбук Ansible, например, для настройки ролей, пользователей, монтирования, деплоймента приложений.

Гибкость и расширяемость

Благодаря JSON-формату, конфигурации легко генерировать программно. Можно иметь набор шаблонов (для разных целей — CI, разработка, прокси, шлюз), которые автоматически подставляются в ISO на этапе сборки. NiceOS-инсталлятор обрабатывает любые допустимые конфигурации, и в случае ошибки — возвращает читаемую диагностику.

Скрипты, указанные в `postinstall_scripts`, выполняются в **chroot-среде** целевой системы, то есть они уже работают как в "настоящей" установленной ОС. Это позволяет выполнять любые настройки сразу, без необходимости дополнительных перезагрузок или внешнего вмешательства.

Автоматизация массового развёртывания

Использование kickstart-конфигураций делает **NiceOS V** идеальным решением для:

- предсборки шаблонов для KVM/VMware/VirtualBox;
- инициализации облачных образов с преднастроенным окружением;
- генерации готовых к CI/CD машин;

- построения гибких базовых образов (base image) с универсальным стеком;
- однородного развёртывания в корпоративных средах.

Заключение раздела

Поддержка JSON-конфигураций делает установку NiceOS V не просто быстрой, а предсказуемо повторяемой. От разметки диска до запуска Ansible — весь процесс описывается декларативно и может быть сохранён в системе контроля версий. Такой подход обеспечивает чистоту, контроль и масштабируемость при развёртывании виртуальных машин.

5. Создание VM-образов: create-image-util

Для формирования готовых к использованию виртуальных машин в **NiceOS V** используется мощный и гибкий инструмент — `create-image-util`. Это shell-скрипт, предназначенный для запуска сборки образов внутри Docker-контейнера, поддерживающий как локальные, так и удалённые репозитории, разные архитектуры и широкий спектр выходных форматов: **RAW, OVA, OVF, VMDK, AMI, Azure, RPI**.

Скрипт построен по принципу инфраструктуры как кода: все параметры сборки указываются через флаги командной строки, конфигурационные файлы и переменные окружения. Благодаря этому, процесс создания образа полностью воспроизводим и легко интегрируется в CI/CD.

Этапы работы create-image-util

1. **Инициализация окружения:** определяется целевая архитектура (`x86_64` или `aarch64`), подбирается нужный контейнерный образ, проверяется его наличие или выполняется сборка.
2. **Создание локального репозитория:** если не указан внешний URL, скачиваются RPM-пакеты и создаётся локальный репозиторий по конфигурации kickstart.
3. **Формирование RAW-образа:** в контейнере запускается процесс установки ОС с применением JSON/YAML-конфигурации и создаётся базовый RAW-диск.
4. **Преобразование RAW в целевой формат:** в зависимости от выбранного типа (`--flavor`), из RAW-файла собирается OVA, VMDK, AMI, Azure-образ и т.д.

Ключевые параметры CLI

Вызов `create-image-util` требует набора обязательных и опциональных аргументов:

- `--raw-image` — имя выходного RAW-файла диска (например, `minimal.img`).
- `--config-file` — путь к Kickstart/YAML конфигурации установки.
- `--local-repo-path` — директория с локальным RPM-репозиторием или место для его создания.
- `--poi-path` — путь к исходникам сборочной среды (установщика).
- `--flavor` — формат итогового образа: `ova`, `ami`, `azure`, `rpi` и др.
- `--ova-config`, `--ova-name` — параметры для сборки OVA/OVF-образов.
- `--src-repo-url` — URL к удалённому репозиторию RPM-пакетов (если не используется локальный).
- `--arch` — архитектура целевой платформы: `x86_64` или `aarch64`.
- `--ovf`, `--mf`, `--vmdk-only` — дополнительные опции для форматов семейства OVA.

Примеры использования

Простой OVA-образ:

```
./create-image-util \
  --raw-image minimal.img \
  --config-file minimal_ks.yaml \
  --local-repo-path ./repo/5.2 \
  --poi-path ./niceos-installer \
  --flavor ova \
  --ova-config ova.yaml \
  --ova-name niceos-vm
```

Создание AMI-образа для AWS:

```
./create-image-util \
  --config-file ami_ks.yaml \
  --local-repo-path ./repo/5.2 \
  --poi-path ./niceos-installer \
  --flavor ami
```

Сборка образа с удалённым репозиторием:

```
./create-image-util \
  --raw-image minimal.img \
```

```
--config-file minimal_ks.yaml \
--local-repo-path ./repo/5.2 \
--poi-path ./niceos-installer \
--flavor ova \
--ova-config ova.yaml \
--ova-name niceos-vm \
--src-repo-url=https://repo.niceos.local/5.2/x86_64/
```

Поддерживаемые форматы и окружения

- **OVA / OVF / VMDK** — для развёртывания в VMware, VirtualBox, Proxmox, vSphere.
- **AMI** — экспортируемые образы для AWS EC2 через `ec2 import-image`.
- **Azure** — генерация совместимых дисков для Azure VM.
- **RPI (aarch64)** — создание образов для Raspberry Pi (без GUI).
- **RAW/QCOW2** — для KVM, OpenStack, локальных виртуалок.

Инфраструктурная интеграция

Поскольку весь процесс обёрнут в Docker, его легко запускать из пайплайнов CI/CD: GitLab CI, Jenkins, Drone, GitHub Actions и других. Все файлы конфигурации и ресурсы (OVA-шаблоны, RPM-пакеты, JSON/YAML) могут храниться в Git-репозиториях, обеспечивая полную трассируемость и контроль версий.

Поддержка архитектур **x86_64** и **aarch64** позволяет создавать образы как для обычных серверов, так и для ARM-устройств, включая Raspberry Pi и облака с ARM-инфраструктурой (например, AWS Graviton).

Заключение раздела

`create-image-util` — это единая точка управления созданием образов NiceOS V. Он формирует мост между декларативной конфигурацией установки и реальным готовым образом, который можно использовать в продакшене, тестировании, автоматизации и облачных развёртываниях. Благодаря полной параметризации и запуску в контейнере он является надёжным элементом любой современной инфраструктуры.

6. Лицензирование и юридическая информация

NiceOS V распространяется на условиях, полностью соответствующих как

российскому законодательству, так и общепринятым мировым практикам открытого программного обеспечения. Все права и обязанности пользователей определены в документе `EULA.txt`, входящем в состав дистрибутива.

Лицензионные условия

- **Конечное пользовательское соглашение (EULA):** предоставляется на русском языке, включает разделы об ограничении ответственности, праве использования в корпоративной и государственной среде, а также ссылку на ООО «НАЙС СОФТ ГРУПП» как правообладателя.
- **Документация:** поставляется на русском языке, содержит юридические примечания, инструкции по развёртыванию, сведения о лицензиях и указания по модификации образов в рамках разрешённого использования.
- **Открытость кода:** все основные модули, включая установщик, конфигураторы, модули postinstall и сборочные скрипты распространяются под лицензиями **GPLv2, GPLv3 или LGPL**, в зависимости от специфики компонента.
- **Безопасность юридического использования:** дистрибутив может применяться как в частных, так и в государственных организациях, включая инфраструктурные проекты, при условии соблюдения условий EULA.

Почему NiceOS V идеальна для виртуальных машин

Эта редакция создавалась именно как базовая система для виртуальных сред, и потому обладает набором качеств, не характерных для традиционных серверных или пользовательских дистрибутивов:

- **Минимальный след:** отсутствуют графические и мультимедийные компоненты, запущены только необходимые сервисы, всё логирование управляется systemd. Размер ISO минимален.
- **Быстрое развёртывание:** установщик работает в headless-режиме и позволяет автоматически подготовить VM за считанные секунды, включая SSH, локаль, hostname и настройки сети.
- **Чёткая автоматизация:** JSON-конфигурации, встроенная поддержка Ansible и shell-скриптов, возможность передачи параметров через kernel или cloud-init делают развёртывание полностью предсказуемым.
- **Простота в сопровождении:** меньше пакетов — меньше уязвимостей, проще обновление, выше стабильность, нет зависимости от окружений рабочего стола или сложных компонентов.

Возможности расширения

Хотя NiceOS V уже готов к использованию в большинстве сценариев, архитектура дистрибутива и установщика позволяет легко адаптировать систему под более сложные требования. Возможные направления:

- **Поддержка новых облаков:** добавление шаблонов и форматов образов для платформ **Yandex.Cloud, Azure, Google Cloud** через расширение create-image-util.
- **Веб-интерфейсы и мониторинг:** интеграция с инструментами управления, такими как **Cockpit, Netdata, Webmin**, для наблюдения и базовой настройки через браузер.
- **Создание кастомных образов на лету:** реализация механизма, позволяющего генерировать образы из заданных JSON-файлов и пакетов без полной пересборки ISO — например, через REST API или веб-консоль.
- **Интеграция с облачными marketplace:** публикация образов в официальных каталоги (AWS Marketplace, Yandex Cloud Image Catalog, Microsoft Azure Image Gallery).

Всё это делает **NiceOS V** универсальной платформой для развёртывания инфраструктуры в облаке, в локальных средах виртуализации, в изолированных средах или на edge-устройствах.

Заключение

Благодаря открытости, модульности и ясной юридической модели распространения, NiceOS V становится не просто ещё одним дистрибутивом, а **базовой платформой для инфраструктур будущего** — безопасных, управляемых и предсказуемых.