



Cloud Native apps

Что такое Cloud Native?



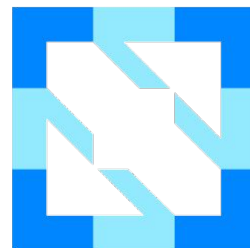
- **Cloud (computing):** Это система, которая предоставляет ИТ-ресурсы по запросу (инфраструктуру, хранилища, базы данных и все виды прикладных сервисов) через Интернет.
- **Cloud Native:** Это модель архитектуры для разработки приложений с использованием основных принципов облачных вычислений, таких как масштабируемость, гибкость и динамичность. Это трансформация мышления и культуры в способах создания, развертывания и сопровождения приложений.

Cloud-native позволяет значительно сократить time to market на 40-60%, и это достигается за счет снижения эксплуатационных расходов примерно на 30-50%. DOI:

<https://doi.org/10.30574/wjaets.2025.15.2.0630>



Cloud Native Computing Foundation (CNCF) — это фонд под эгидой Linux Foundation, созданный в 2015 году для продвижения и поддержки облачных технологий и проектов с открытым исходным кодом, таких как Kubernetes и Prometheus, которые лежат в основе создания масштабируемых и динамичных приложений в облачных средах





Что такое Cloud Native приложения?

Cloud Native приложения имеют:

- ❖ **возможность эксплуатации**, т.е. управления жизненным циклом приложения/системы внешними инструментами;
- ❖ **возможность наблюдения**, т.е. наличие механизмов получения текущего состояния и производительности;
- ❖ **эластичность**, позволяющая увеличиваться и уменьшаться в размерах в рамках доступных ресурсов и в зависимости от меняющейся нагрузки;
- ❖ **отказоустойчивость**, гарантирующая быстрое автоматическое восстановление в случае падений;
- ❖ **динамичность (agility)**, включающая в себя быстрые развёртывания (деплой), итерации, изменения конфигурации.

Таким образом, Cloud Native — это не только про запуск приложений в облаке, но и про такое их создание, при котором приложения изначально рассчитаны на работу в облаке.

Agile разработка

Если раньше водопадная модель (waterfall) была нормой, то теперь cloud native действительно обеспечивает **agile разработку**.

По Agile вам необходимо вносить изменения **быстро, часто и надежно!**



Архитектура Cloud Native: основные атрибуты приложений



Объединяет компоненты, которые можно использовать для разработки масштабируемых облачных приложений — их же называют атрибутами.

Containerization



Package an app with all its dependencies in an isolated environment while still using the host OS kernel.

Automation



Reduce human error, make efficient repeatable processes and take fine-grained control over app infrastructure.

Orchestration



Automate lifecycle management for containers in a production environment to efficiently manage tasks.

Microservices



Divide an app into independently created tasks and services in which each one runs a unique process.

Service mesh



Create a dedicated network layer that enables simpler, observable communication between services.

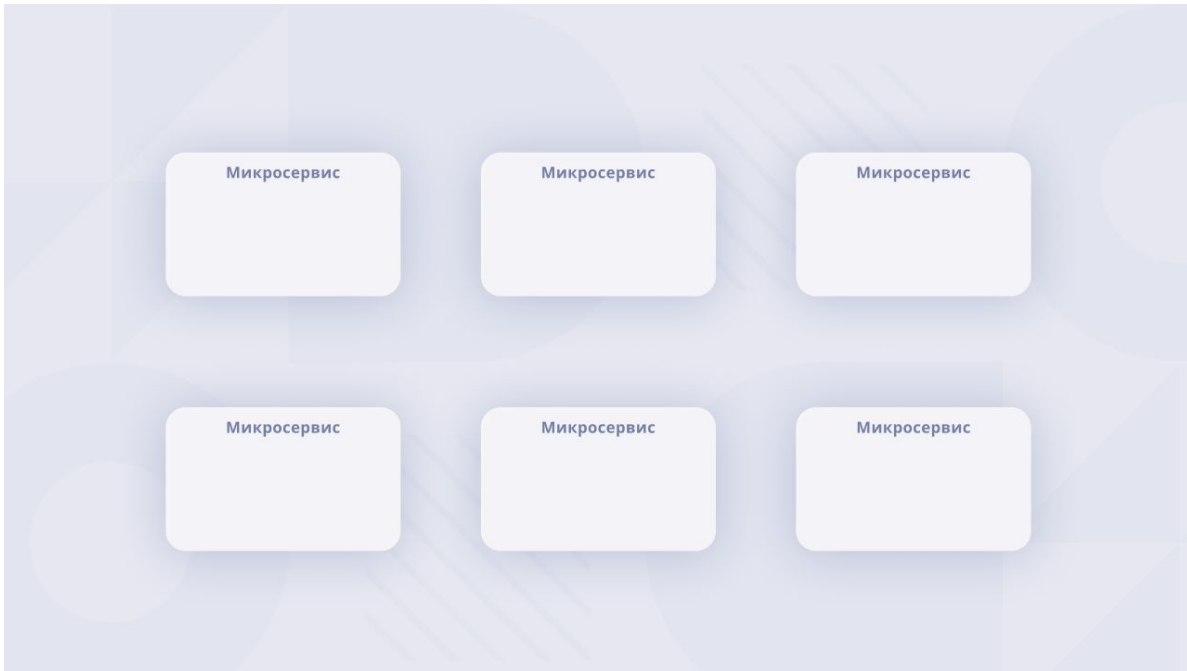
Микросервисы



Cloud Native-приложение обычно разделено на **микросервисы** — небольшие независимые части, каждая из которых выполняет отдельную функцию.

Например, в приложении по доставке еды могут быть микросервисы, отвечающие за меню, оформление и обработку заказов, оплату, доставку и отзывы клиентов.

Некоторые преимущества:
проще вносить изменения,
можно разбить на команды,
повышение
отказоустойчивости.



API

Для взаимодействия микросервисов требуется **механизм API**, где прописано, какие данные и в каком формате микросервис может отдавать и принимать.

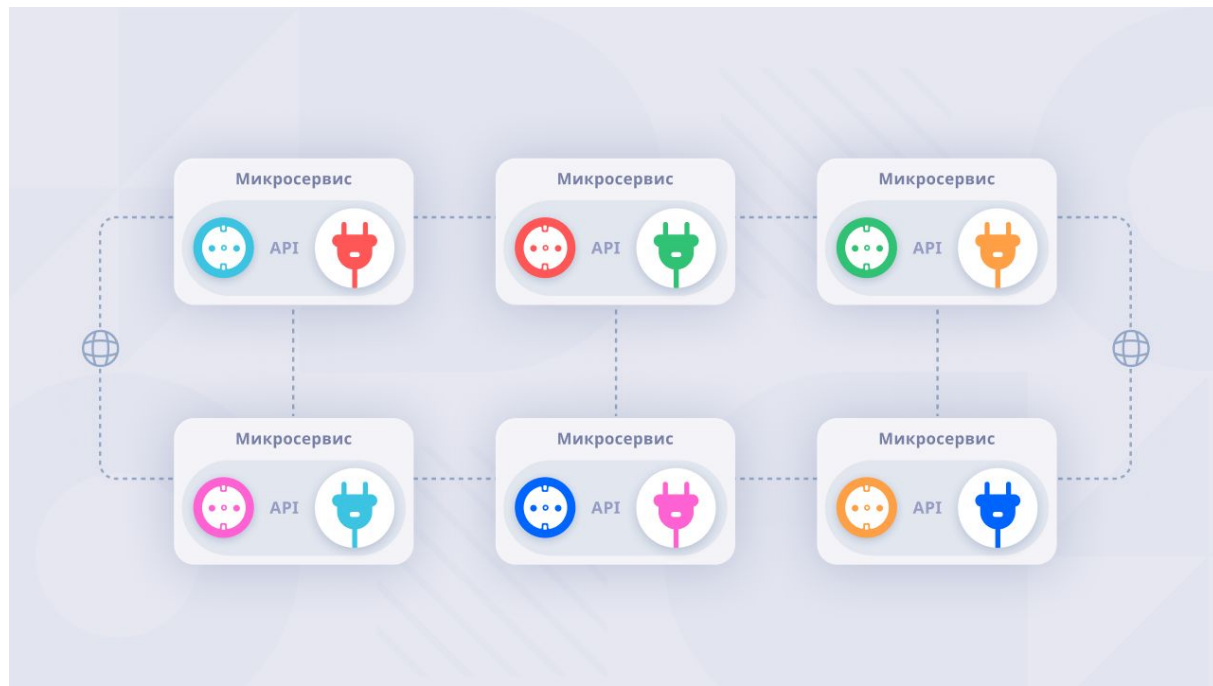
Например, микросервис оплаты может получать стоимость заказа и платёжные данные клиента, а отдавать подтверждение оплаты.



Сетка сервисов (service mesh)

Если API — это возможность связи между микросервисами, то **service mesh** — сама связь и наблюдение за ней.

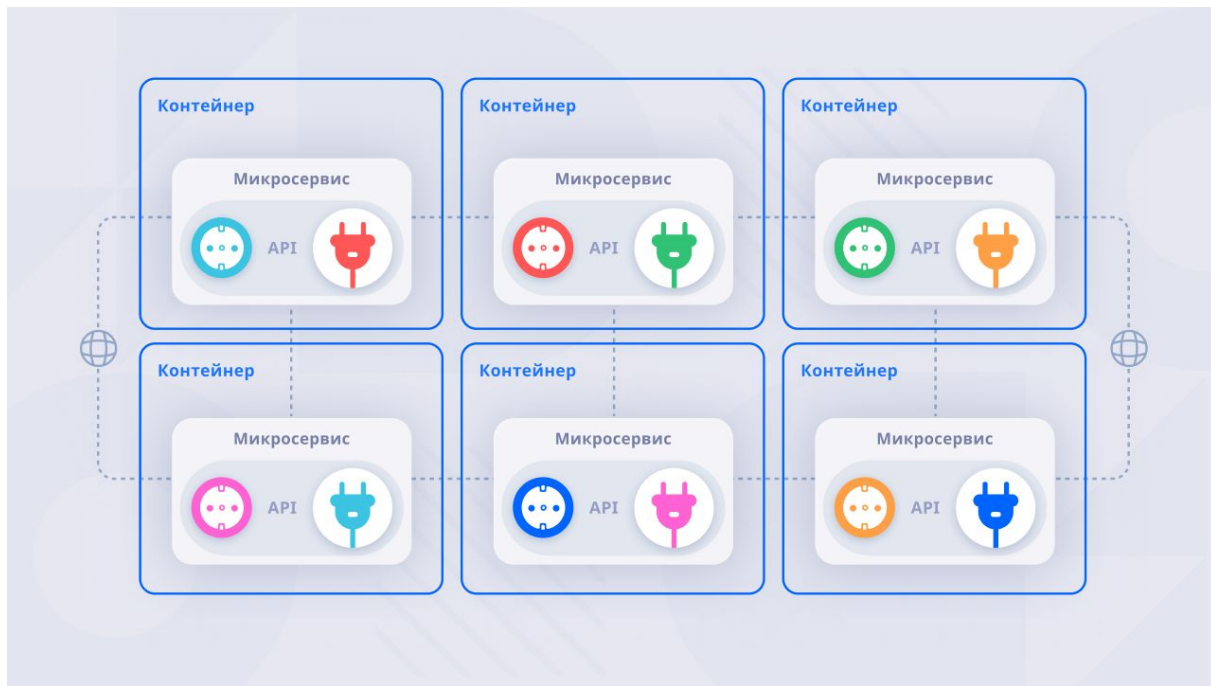
В том числе — обнаружение сервисов, проверка их состояния, маршрутизация, балансировка нагрузки, шифрование данных.



Контейнеры

Микросервисы упаковываются в **контейнеры** — изолированную среду, в которую кроме кода самого микросервиса помещается всё, что необходимо ему для работы, например, файлы библиотек и фреймворков.

В крупных приложениях контейнеров много, поэтому ими необходимо управлять. Для этого используются **системы оркестрации** контейнеров, например Kubernetes.





Неизменяемая IT-инфраструктура

Это не часть архитектуры Cloud Native-приложения, а скорее её особенность.

Неизменяемость подразумевает, что если приложению для работы требуется больше ресурсов, то серверы, на которых оно развёрнуто, не меняются под новые потребности, а просто удаляются, и приложение автоматически «переселяется» на новые, более мощные серверы.

Такой подход позволяет перейти от ручного развёртывания к **автоматизации** и сократить число человеческих ошибок.

CLOUD NATIVE Landscape



Cloud Native Landscape v2.0

See the interactive landscape at landscape.cncf.io

Database & Data Warehouse Streaming Source Code Management Application Definition Continuous Integration / Continuous Delivery (CI/CD)

App Definition & Development

Orchestration & Management

Scheduling & Orchestration Coordination & Service Discovery Service Management

Cloud Native Storage Container Runtime Cloud Native Network

Runtime

Host Management / Tooling Infrastructure Automation Container Registries Secure Images Key Management

Provisioning

Public Private

Cloud

This landscape is intended as a map through the previously uncharted terrain of cloud native technologies. There are many routes to deploying a cloud native application, with CNCF Projects representing a particularly well-traveled path.

github.com/cncf/landscape

CLOUD NATIVE Landscape CLOUD NATIVE COMPUTING FOUNDATION Redpoint Amplify

Platforms

Observability & Analysis

Monitoring Logging Tracing

Serverless

Kubernetes Certified Service Provider

Special

Grouped logos are not open source

Разделы:

Разработка приложений
Оркестровка и управление
Окружение для исполнения
Инструментарий
Платформы
Наблюдение и анализ
Облака



<https://landscape.cncf.io/>

Отличие от on-premise-приложений



	Cloud Native приложения	On-premise-приложения
Архитектура и разработка	Позволяют легко вносить изменения в отдельные компоненты и добавлять новые по мере необходимости.	Нередко разрабатываются как монолитные структуры, где изменения в одной части могут повлиять на весь проект, что усложняет развитие продукта
Автоматизация	Характерна высокая степень автоматизации процессов. Важную роль здесь играют практики DevOps, например, непрерывная интеграция и доставка (CI/CD).	Автоматизация тоже не чужда, но процесс обновления может быть более сложным и рискованным.

Отличие от on-premise-приложений



	Cloud Native приложения	On-premise-приложения
Масштабируемость	Это одно из ключевых преимуществ. Облака позволяют быстро и легко увеличить ресурсы для работы приложения.	В случае с физическими серверами так не получится — придётся обновлять существующие серверы или покупать новые, а также искать место в дата-центрах, что может быть непросто.
Управление расходами	При использовании облаков компании обычно платят за фактически использованные ресурсы. Это обеспечивает гибкость управления бюджетом.	Преобладают капитальные затраты на приобретение оборудования и его обслуживание. При этом часть серверов может простаивать в моменты, когда нагрузки не такие высокие.

Отличие от on-premise-приложений



	Cloud Native приложения	On-premise-приложения
Обслуживание и обновление инфраструктуры	В случае использования публичных облаков, ответственность за обслуживание и обновление «железа» лежит на облачном провайдере. Это позволяет компаниям сосредоточиться на улучшении своего продукта, а не поддержке серверов.	Ответственность за все аспекты надёжности инфраструктуры ложится на сам бизнес, что потребует наличия в штате отдельных специалистов. Но при этом on-premise обеспечивает полный контроль за данными и инфраструктурой, без которого не могут работать компании с особо высокими требованиями к безопасности.

Путь к Cloud Native



В 2015 году на заре становления Cloud Native появилась классификация приложений по степени готовности к облаку.

Cloud Ready. Приложению не требуется постоянный доступ к диску, оно контейнеризировано, а управление с внешними сервисами осуществляется за счет конфигов. Есть возможность подключить сторонние службы.

Cloud Friendly. Приложение [12 факторов](#). Здесь появляется горизонтальное масштабирование за счет ресурсов облачной платформы.

Cloud Resilient. Решения, построенные так, что увеличение нагрузки или отказ отдельных компонентов не приведет к недоступности сервиса.

Cloud Native. Микросервисная архитектура, при проектировании используется принцип API-first.

