



Data Warehouse Analyst

otus.ru



Проверить, идет ли запись

Меня хорошо видно && слышно?



Тема вебинара

Инструментарий разработки



Бурковский Никита

Data Engineer

- 2 года опыта работы backend-разработчиком (Django, Flask, FastApi)
- Действующий инженер данных в команде DWH (Airflow, dbt, различные СУБД)
- Преподаватель с **2-летним** стажем



Правила вебинара



Активно
участвуем



Задаем вопрос
в чат или ГОЛОСОМ



Вопросы вижу в чате,
могу ответить не сразу

Маршрут вебинара

1. IDE, Terminal, git

2. Yandex.Cloud, Terraform

3. Docker, Github Codespaces, Actions

4. Практическое задание

Цели вебинара

После занятия вы сможете

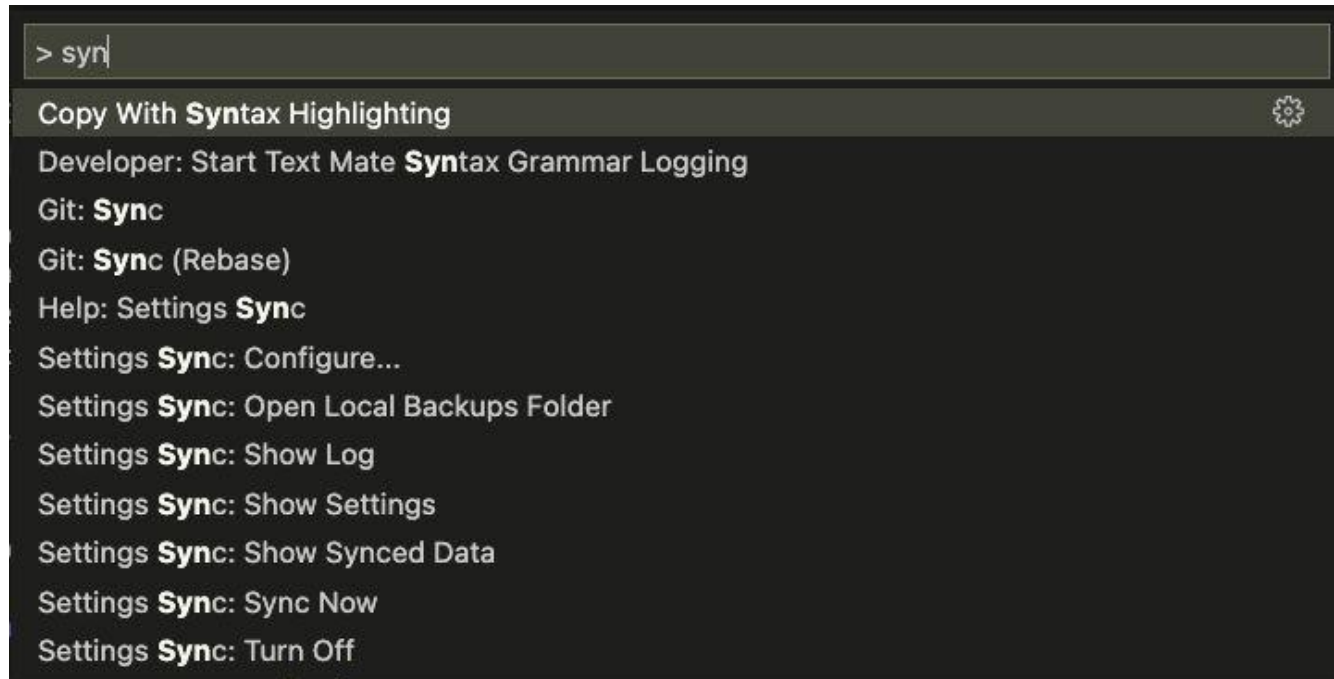
1. Организовать удобную современную среду разработки
2. Иметь навыки работы с облачными провайдерами и инфраструктурой
3. Продемонстрировать полученные знания на практическом примере

Integrated Development Environment



















Какой IDE пользуетесь? С чем имели опыт?



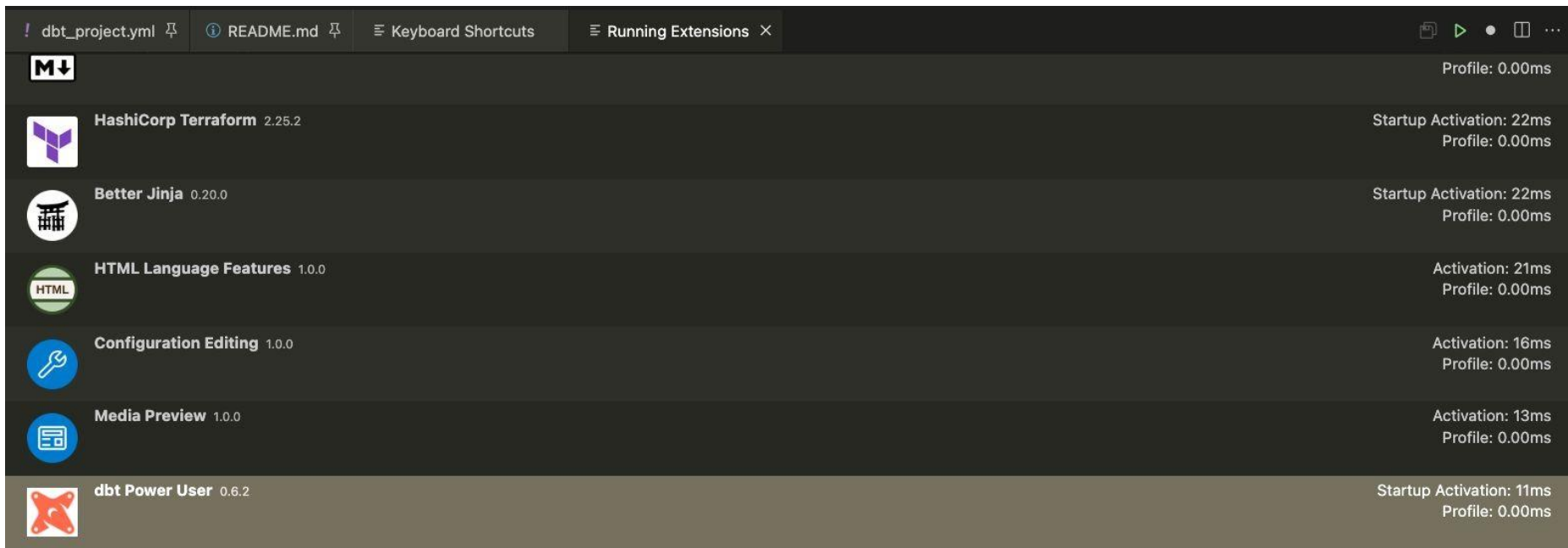
VS Code – Command palette (cmd + shift + p)










VS Code – Keyboard shortcuts

Type to search in keybindings (↑↓ for history)				
Command	Keybinding	When	Source	
Add Cursor Above	  ↑	editorTextFocus	Default	
Add Cursor Below	  ↓	editorTextFocus	Default	
Add Cursors to Line Ends	  ⏎	editorTextFocus	Default	
Add Line Comment	 K  C	editorTextFocus && !editorReadOnly	Default	
Add Selection To Next Find Match	 D	editorFocus	Default	
Auto Fix...	  .	editorTextFocus && !editorReadOnly && supportedCodeAction =~ /(\/\s .../	Default	
Calls: Show Call Hierarchy	  H	editorHasCallHierarchyProvider	Default	
Cancel Selection Anchor	Escape	editorTextFocus && selectionAnchorSet	Default	
Change All Occurrences	 F2	editorTextFocus && !editorReadOnly	Default	
Change Language Mode	 K M	!notebookEditorFocused	Default	
Close Exception Widget	Escape	exceptionWidgetVisible	Default	
Close Window	  W	-	Default	
Close Window	 W	!editorIsOpen && !multipleEditorGroups	Default	

Extensions (dbt, Terraform, Docker)



The screenshot shows the 'Running Extensions' panel in VS Code. The top bar includes tabs for 'dbt_project.yml', 'README.md', 'Keyboard Shortcuts', and 'Running Extensions'. The panel lists several extensions with their icons, names, versions, and performance metrics (Startup Activation and Profile).

Extension Icon	Extension Name	Version	Startup Activation	Profile
				0.00ms
	HashiCorp Terraform	2.25.2	22ms	0.00ms
	Better Jinja	0.20.0	22ms	0.00ms
	HTML Language Features	1.0.0	21ms	0.00ms
	Configuration Editing	1.0.0	16ms	0.00ms
	Media Preview	1.0.0	13ms	0.00ms
	dbt Power User	0.6.2	11ms	0.00ms

Markdown preview (↑🔗V)

! dbt_project.yml ⓘ ⓘ README.md ⓘ Preview README.md × ⌵ Keyboard Shortcuts ⌵ Running Extensions

1. Deploy Clickhouse

1. Get familiar with Managed Clickhouse Management Console

yccloud default Managed Service for ClickHouse / Clusters / Create cluster

Version 22.8 LTS

Resources

Platform Intel Ice Lake

Type memory-optimized standard **cpu-optimized** burstable

Host class	VCPU	VCPU fraction	Memory
c3-c2-m4	2 cores	100%	4 GB
c3-c4-m8	4 cores	100%	8 GB
c3-c8-m16	8 cores	100%	16 GB
c3-c12-m24	12 cores	100%	24 GB
c3-c16-m32	16 cores	100%	32 GB
c3-c24-m48	24 cores	100%	48 GB
c3-c32-m64	32 cores	100%	64 GB
c3-c40-m80	40 cores	100%	80 GB
c3-c48-m96	48 cores	100%	96 GB
c3-c64-m128	64 cores	100%	128 GB
c3-c80-m160	80 cores	100%	160 GB
c3-c96-m192	96 cores	100%	192 GB

3 960,50 P per month

Pricing

ClickHouse: Intel Ice Lake, 100% vCPU
2 476,80 P

ClickHouse: Intel Ice Lake, RAM
1 353,60 P

Fast network storage — ClickHouse
130,10 P

Save up to 22%

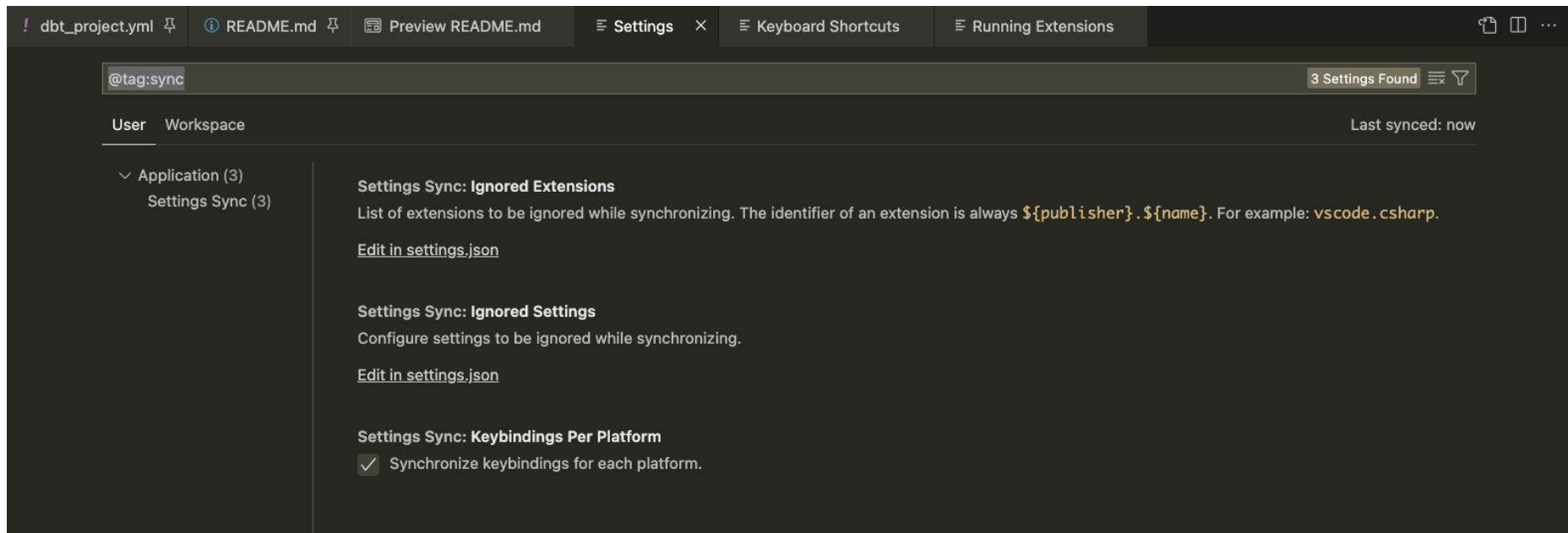
Commit to managed database resources for 6 months or 1 year and pay less.

Learn more

2. Install and configure yc CLI: [Getting started with the command-line interface by Yandex Cloud](#)

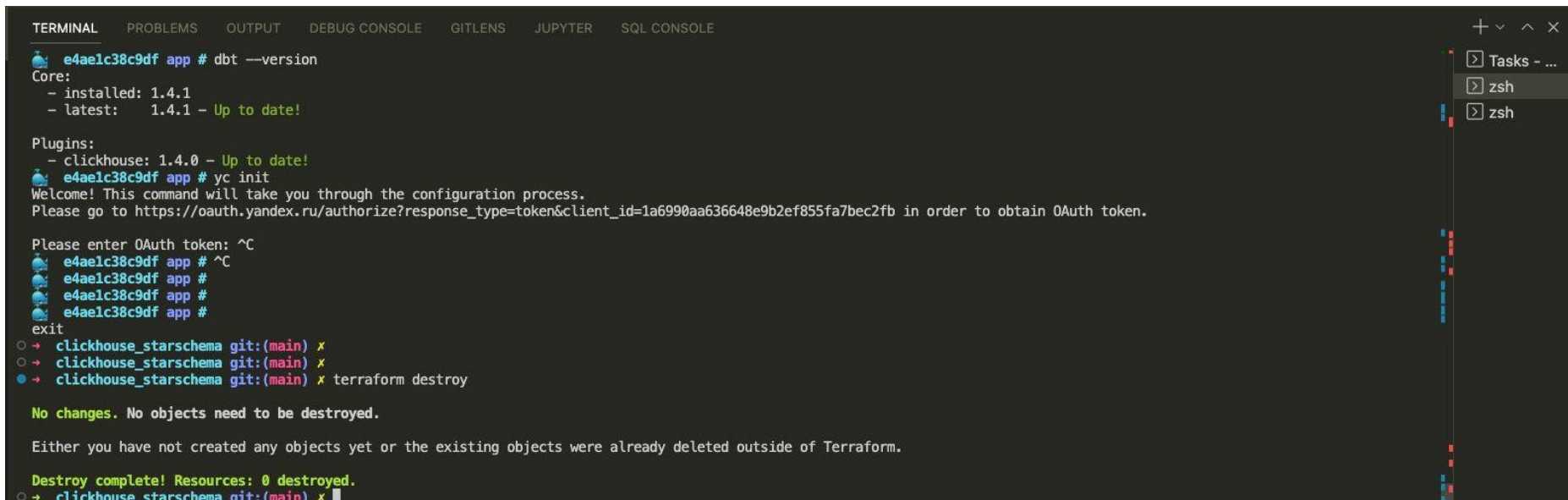
```
yc init
```

Sync settings between devices



Terminal

Built-in terminal (ctrl + `)



```
TERMINAL  PROBLEMS  OUTPUT  DEBUG CONSOLE  GITLENS  JUPYTER  SQL CONSOLE
e4a1c38c9df app # dbt --version
Core:
- installed: 1.4.1
- latest: 1.4.1 - Up to date!

Plugins:
- clickhouse: 1.4.0 - Up to date!
e4a1c38c9df app # yc init
Welcome! This command will take you through the configuration process.
Please go to https://oauth.yandex.ru/authorize?response_type=token&client_id=1a6990aa636648e9b2ef855fa7bec2fb in order to obtain OAuth token.

Please enter OAuth token: ^C
e4a1c38c9df app # ^C
e4a1c38c9df app #
e4a1c38c9df app #
e4a1c38c9df app #
exit
○ → clickhouse_starschema git:(main) x
○ → clickhouse_starschema git:(main) x
● → clickhouse_starschema git:(main) x terraform destroy

No changes. No objects need to be destroyed.

Either you have not created any objects yet or the existing objects were already deleted outside of Terraform.

Destroy complete! Resources: 0 destroyed.
○ → clickhouse_starschema git:(main) x
```

Z shell (zsh)

- [Installing zsh](#)

Shell commands basics

- pwd, ls, cd, mkdir, rm
- hotkeys: editing, history (ctrl+r), interrupt (ctrl+c)
- [Linux tutorial](#)
- [Cheat sheet](#)

Version Control System

Есть ли опыт использования git?



git basics

- Essential git commands: status, log, checkout, add, commit, push, fetch, [rebase](#), merge
- [Atlassian tutorials](#)
- [Collaborating with git](#)
- [Making a Pull Request](#)
- [Comparing Git Workflows](#)

Yandex.Cloud CLI

Command Line Interfaces (CLI)

- [Install yc CLI](#)
- [yc CLI reference](#)
- [Yandex Object Storage CLI \(AWS CLI\)](#)

Terraform

Infrastructure as Code using Terraform

- [Infrastructure as Code principle](#)
- [Terraform Files and Directories](#)
- [Terraform Syntax](#)
- [Terraform Variables and Outputs](#)
- [Terraform essential commands: init, validate, plan, apply, destroy](#)

Docker

Знакомы с контейнерами?



Containerizing apps and environments

- [Docker Get Started](#)
- Essential files: Dockerfile, docker-compose.yml
- Essential commands: [docker](#), [docker-compose](#)

Контейнеры и VM

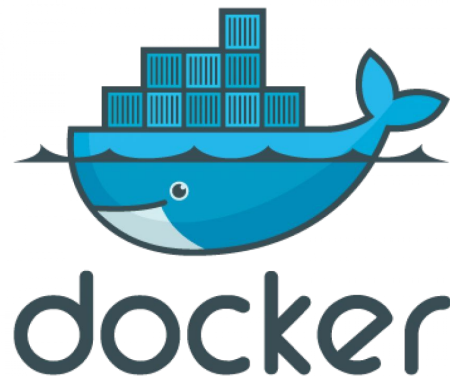
Что нужно знать о докере?

Контейнер - это легкий, автономный и исполняемый программный пакет, включающий в себя все необходимое для запуска программного обеспечения.

Docker - это и название компании (Docker Inc.), и созданное ею программное обеспечение, которое упаковывает программы в контейнеры.

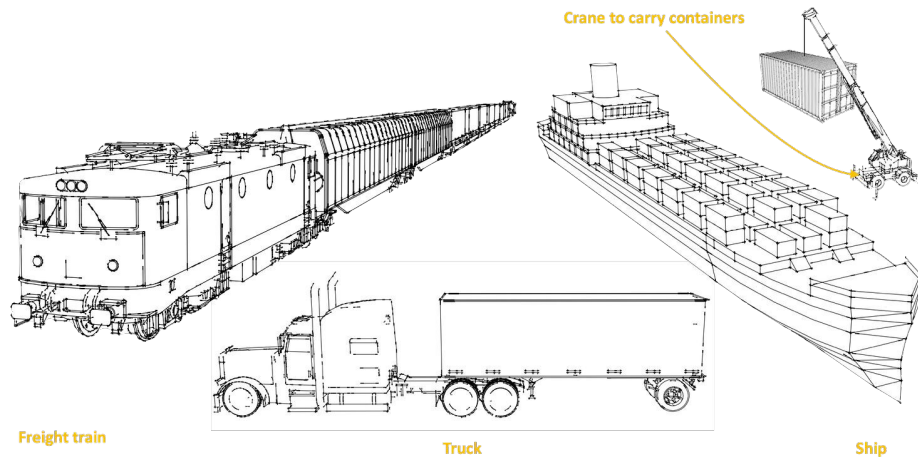
Для понимания работы контейнеров разберемся в следующих темах:

- Контейнеры-доставщики
- Виртуальные машины



Контейнеры-доставщики

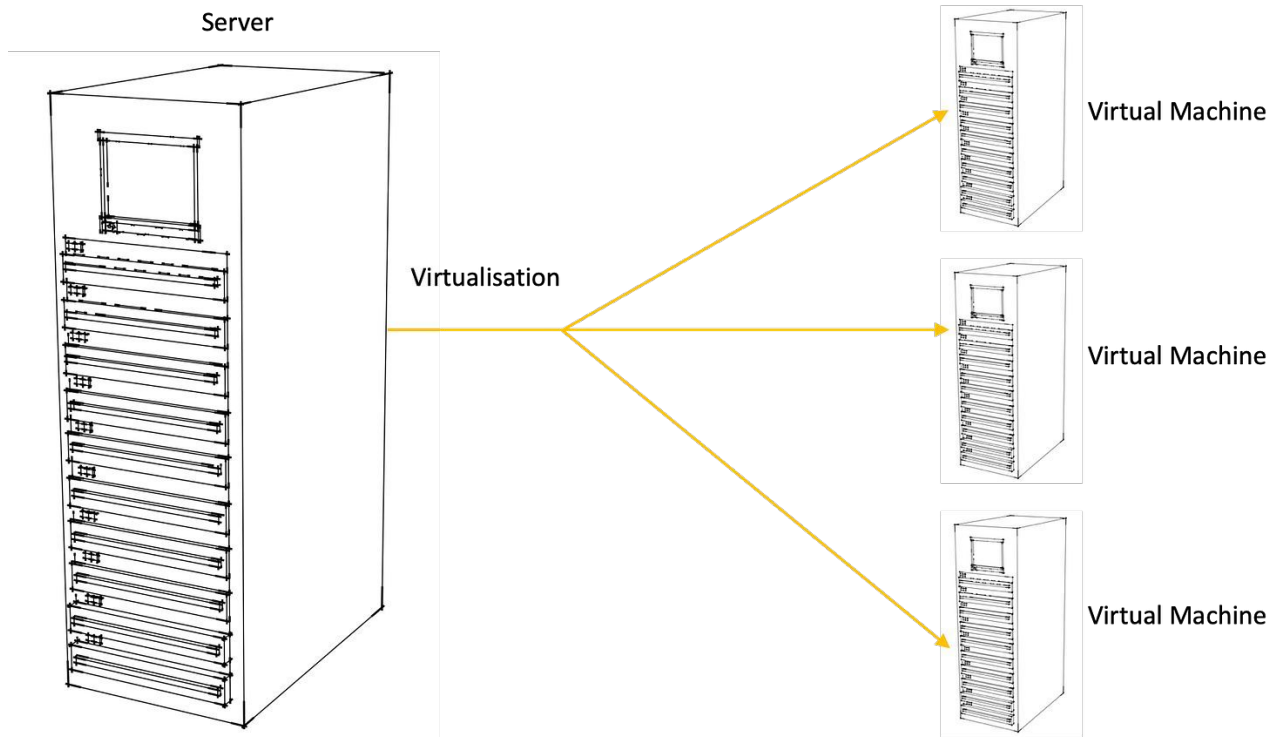
- История контейнеров и докер-контейнеров имеет много общего
- Стандартизация транспортных контейнеров делает их портативными, то есть легко перемещаемыми с места на место. Эта **переносимость** - ключевая особенность контейнеров Docker.



Что такое виртуальные машины?

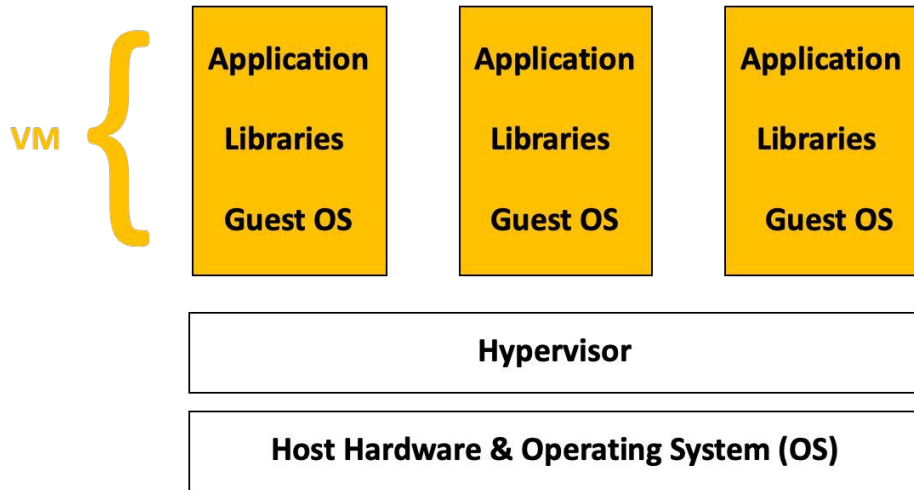
- Виртуальные машины (**ВМ**) создаются с помощью процесса, называемого виртуализацией.
 - **Виртуализация** - это технология, позволяющая создавать на одной физической машине несколько симулированных сред или виртуальных версий.
- В виртуальных машинах **виртуализируется аппаратное обеспечение**.
 - ВМ берет одну единицу оборудования - сервер - и создает виртуальные версии других серверов, работающих под управлением собственных операционных систем.
 - **Физически** это всего лишь одна единица оборудования.
 - **Логически** на одном аппаратном обеспечении может работать несколько виртуальных машин.

Что такое виртуальные машины?



Как работает виртуализация?

- В основе лежит **аппаратное обеспечение хоста и ОС**.
 - Это физическая машина, которая используется для создания виртуальных машин.
- Сверху располагается **гипервизор**.
 - Позволяет запускать на одном физическом сервере несколько виртуальных машин со своей ОС.



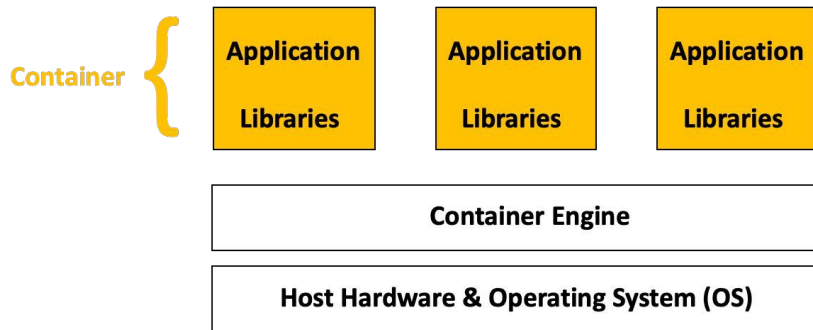
Особенности VM

- **VM потребляют больше ресурсов** из-за необходимости запускать полный экземпляр ОС для каждой VM.
 - Это может привести к увеличению потребления памяти и дискового пространства.
 - Это, в свою очередь, может негативно сказаться на производительности и времени запуска виртуальной машины.
- **Переносимость.** VM обычно менее переносимы из-за различий в базовых средах ОС.
 - Перемещение виртуальных машин между различными гипервизорами или облачными провайдерами может быть более сложным.

Что такое контейнеры?

Контейнер - это легкий, автономный и исполняемый программный пакет.

- Включает в себя все необходимое для запуска программного обеспечения.
- Предназначен для изоляции приложений и их зависимостей, обеспечивая их стабильную работу в различных средах.
- Виртуализирует операционную систему. Это означает, что контейнер использует одну ОС для создания виртуального приложения и его библиотек.
- Работает поверх общей ОС, предоставляемой хост-системой.



Особенности контейнеров

- **Переносимость.**
 - Могут работать на любой системе, поддерживающей контейнерную среду выполнения, например Docker, независимо от базовой операционной системы.
- **Эффективность.**
 - Совместно используют операционную систему хост-системы, что снижает накладные расходы, связанные с запуском виртуальной машины с несколькими операционными системами.
- **Согласованность.**
 - Контейнеры объединяют все необходимые компоненты в единое целое.
- **Изоляция.**
 - Каждый контейнер инкапсулирует приложение и его зависимости, не допуская их взаимодействия друг с другом.
- **Быстрое развертывание.**

Вопросы?



Задаем
вопросы в чат



Ставим “-”,
если вопросов нет

Docker

Что такое Docker?

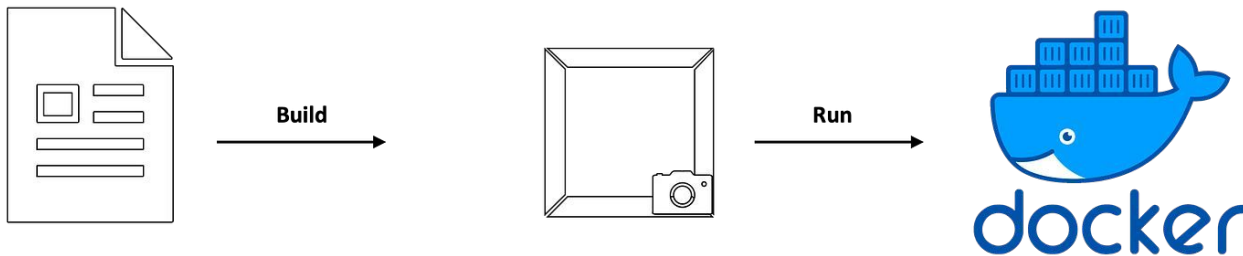
Docker - инструмент для создания и управления контейнерами

- **Dockerfile** содержит набор инструкций для создания образа Docker.
- Образ Docker (**Docker Image**) служит шаблоном для создания контейнеров Docker.
 - Содержит весь необходимый код, среду выполнения, системные инструменты, библиотеки и настройки, необходимые для запуска программного приложения.
- **Dockerfile используется для создания Docker Image, который затем используется в качестве шаблона для создания одного или нескольких контейнеров Docker.**

1. Dockerfile

2. Docker Image

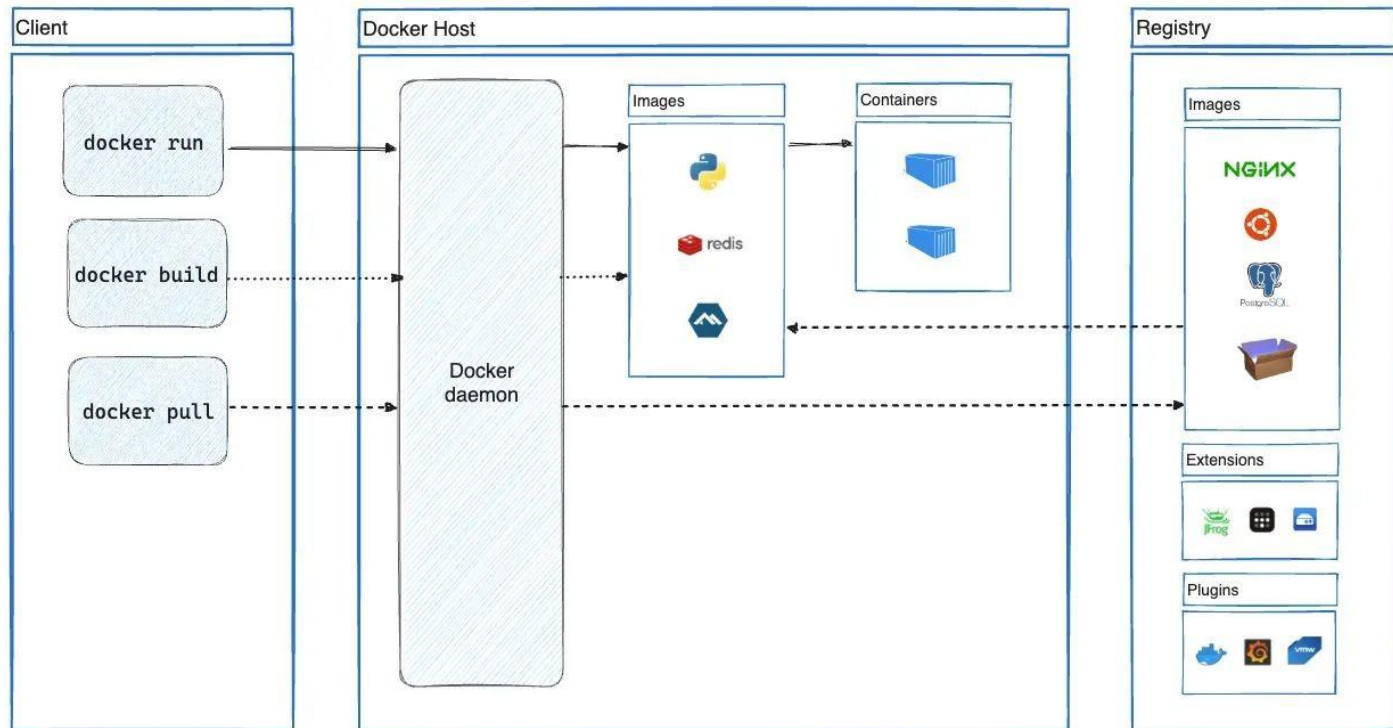
3. Docker Container(s)



Пару слов

- Docker - это компьютерное программное обеспечение, используемое для виртуализации с целью обеспечения работы нескольких операционных систем на одном хосте.
- Docker - это клиент-серверный тип приложения, то есть у нас есть клиенты, которые передают данные на сервер.
- Образы Docker - это "исходный код" для наших контейнеров; мы используем их для сборки
- Dockerfile имеет два типа реестров 1) публичный и 2) приватный реестры
- Контейнеры - это организационные единицы тома Docker. Проще говоря, образ - это шаблон, а контейнер - копия этого шаблона. Можно иметь несколько контейнеров (копий) одного и того же образа.

Архитектура



Главное

Docker Engine

- Демон Docker, называемый **dockerd**, является движком Docker, который представляет собой сервер. Демон docker и клиенты взаимодействуют через бинарные клиенты командной строки, а также через полноценный RESTful API.

Docker Images

- Это "исходный код" наших контейнеров, используем для сборки контейнеров.

Docker Registries

- Docker хранит созданные нами образы в реестрах. Существуют публичные и частные реестры.

Docker Containers

Установка Docker

Проверяем корректность установки:

```
docker run hello-world
```

```
docker ps
```

Базовые команды

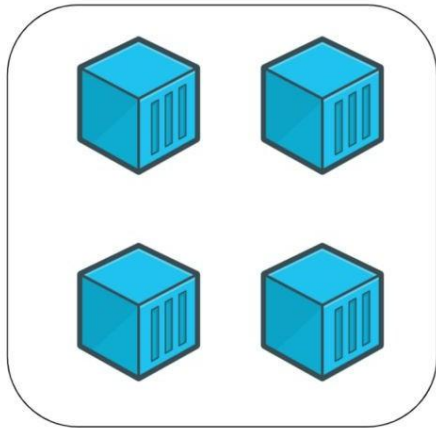
Command	Description
<code>docker info</code>	Information Command
<code>docker pull</code>	Download an image
<code>docker run -i -t image_name /bin/bash</code>	Run image as a container
<code>docker start our_container</code>	Start container
<code>docker stop container_name</code>	Stop container
<code>docker ps</code>	List of all running containers
<code>docker stats</code>	Container information
<code>docker images</code>	List of images downloaded
Docker Cleanup	Kill all running containers.

docker compose

- Docker Compose — это инструментальное средство, входящее в состав Docker. Оно предназначено для решения задач, связанных с развёртыванием проектов.
- Реальные проекты обычно включают в себя целый набор совместно работающих приложений.
- Технология Docker Compose, если описывать её упрощённо, позволяет, с помощью одной команды, запускать множество сервисов.



Docker



Docker-Compose

docker compose

Commands:

build	Build or rebuild services
config	Parse, resolve and render compose file in canonical format
cp	Copy files/folders between a service container and the local filesystem
create	Creates containers for a service.
down	Stop and remove containers, networks
events	Receive real time events from containers.
exec	Execute a command in a running container.
images	List images used by the created containers
kill	Force stop service containers.
logs	View output from containers
ls	List running compose projects
pause	Pause services
port	Print the public port for a port binding.
ps	List containers
pull	Pull service images
push	Push service images
restart	Restart service containers
rm	Removes stopped service containers
run	Run a one-off command on a service.
start	Start services
stop	Stop services
top	Display the running processes
unpause	Unpause services
up	Create and start containers
version	Show the Docker Compose version information

Вопросы?



Задаем
вопросы в чат



Ставим “-”,
если вопросов нет

Github Actions

Acting on your code with Github Actions

- [Understanding GitHub Actions](#)
- [Essential features of GitHub Actions](#)
- [Workflow syntax for GitHub Actions](#)
- Action example: [.github/workflows/ci.yml](#)

Acting on your code with Github Actions

The screenshot displays a GitHub Actions workflow run interface. At the top, a green checkmark icon indicates a successful run for the workflow named 'solution' with ID '196d801'. Below this, a sidebar on the left lists the workflow's steps: 'Continuous Integration Tests' (with a sub-step 'on: pull_request') and 'Clickhouse CI (22.8)'. The 'Clickhouse CI (22.8)' step is highlighted with a blue bar. The main area of the interface shows the detailed log for this step, titled 'Clickhouse CI (22.8)' with a subtitle 'succeeded 19 hours ago in 2m 49s'. The log lists 16 steps, each preceded by a right-pointing chevron and a green checkmark icon, indicating they all completed successfully. The steps are: 'Set up job', 'Build kzzzr/mybi-dbt-action@v4', 'Initialize containers', 'Run actions/checkout@v3', 'wait for services to start up', 'dbt version', 'dbt debug', 'dbt deps', 'initialize sources', 'dbt build', 'Setup tmate session', 'Post Run actions/checkout@v3', 'Stop containers', and 'Complete job'.

✓ solution 196d801

Continuous Integration Tests
on: pull_request

✓ Clickhouse CI (22.8)

Clickhouse CI (22.8)
succeeded 19 hours ago in 2m 49s

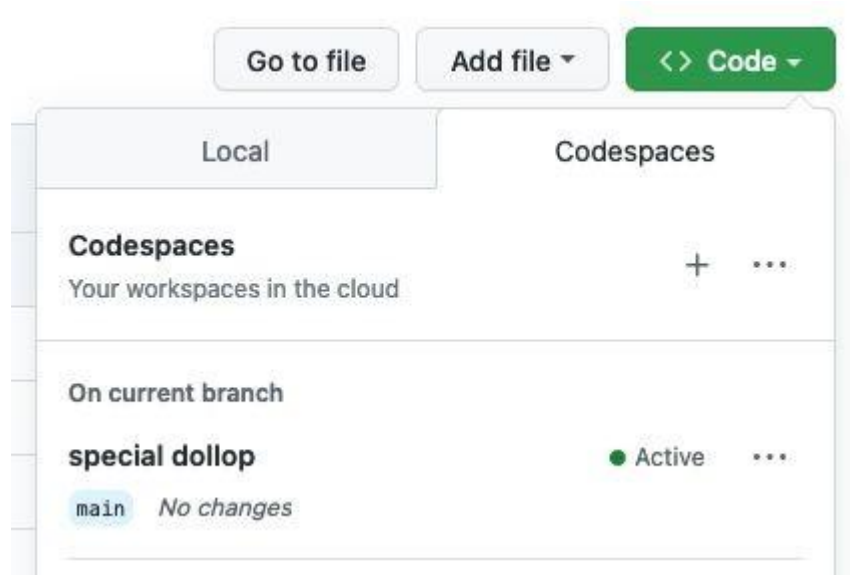
- > ✓ Set up job
- > ✓ Build kzzzr/mybi-dbt-action@v4
- > ✓ Initialize containers
- > ✓ Run actions/checkout@v3
- > ✓ wait for services to start up
- > ✓ dbt version
- > ✓ dbt debug
- > ✓ dbt deps
- > ✓ initialize sources
- > ✓ dbt build
- > ✓ Setup tmate session
- > ✓ Post Run actions/checkout@v3
- > ✓ Stop containers
- > ✓ Complete job

Github Codespaces

Start coding instantly with Codespaces

- A codespace is a development environment that's hosted in the cloud
- [Codespaces overview](#)
- [Adding a dev container configuration](#)

Start coding instantly with Codespaces



Результаты

Цели вебинара

После занятия вы сможете

1. Организовать удобную современную среду разработки
2. Иметь навыки работы с облачными провайдерами и инфраструктурой
3. Продемонстрировать полученные знания на практическом примере

Вопросы?

Опрос о занятии.



Ставим “+”,
если вопросы есть



Ставим “-”,
если вопросов нет



Спасибо за внимание!

Приходите на следующие вебинары



Бурковский Никита

Data Engineer

- 2 года опыта работы backend-разработчиком (Django, Flask, FastApi)
- Действующий инженер данных в команде DWH (Airflow, dbt, различные СУБД)
- Преподаватель с **2-летним** стажем

