Шпаргалка

.NET и Visual Studio

Команда / клавиша	Что делает
`dotnet build`	Собрать решение
`dotnet run --project src/DemoApi`	Запустить сервис без Visual Studio
`dotnet test`	Запустить все тесты
`dotnet test --filter "FullyQualifiedName~Projects"`	Только часть тестов
`dotnet add src/DemoApi package <имя>`	Добавить NuGet-пакет
F5 / **Ctrl+F5**	Запуск с отладкой / без
F9	Точка останова
F10 / **F11**	Шаг с обходом / с заходом в метод
Ctrl+E, T	Test Explorer
`DemoApi.http` → **Send Request**	Отправить HTTP-запрос из редактора

HTTP-запросы из PowerShell

```powershell
$api = 'http://localhost:8080'
irm "$api/api/projects"                                                  # GET
irm -Method Post "$api/api/projects" -ContentType application/json -Body '{"name":"Demo"}'
irm -Method Patch "$api/api/tasks/1/status" -ContentType application/json -Body '{"status":"InProgress"}'
curl.exe -s -o NUL -w "%{http_code}`n" "$api/health/ready"               # только код ответа
```

Docker

Команда	Что делает
`docker run -d --name web -p 8088:80 nginx:alpine`	Запустить в фоне, порт хост:контейнер
`docker ps` / `docker ps -a`	Работающие / все контейнеры
`docker logs -f --tail 50 web`	Логи (последние 50 + поток)
`docker exec -it web sh`	Shell внутри контейнера
`docker stop web` / `docker start web`	Остановить (SIGTERM) / запустить
`docker rm -f web`	Удалить (с остановкой)
`docker build -t app:1.0 .`	Собрать образ
`docker image ls` / `docker image history app:1.0`	Образы / слои образа
`docker inspect web --format '{{.RestartCount}}'`	Одно поле из описания контейнера
`docker stats`	CPU и память контейнеров в реальном времени
`docker events --filter container=web`	Поток событий: start, die, oom, health_status
`docker network create` / `docker network connect`	Сеть / подключить контейнер к сети
`docker system df` / `docker system prune`	Место на диске / очистка неиспользуемого

Docker Compose

Команда	Что делает
`docker compose up -d --build --wait`	Собрать, поднять и дождаться healthy
`docker compose up -d --wait db`	Только БД — для разработки в Visual Studio
`docker compose ps`	Состояние сервисов
`docker compose logs -f api`	Логи сервиса
`docker compose exec db psql -U demo -d demo`	Консоль PostgreSQL
`docker compose stop api` / `start api`	Остановить / запустить сервис

`docker compose down`	Удалить контейнеры и сеть (тома остаются)
`docker compose down -v`	**И тома тоже: данные БД пропадут**
`docker compose config`	Итоговый файл с подставленными переменными
`docker compose -f monitoring/compose.yaml up -d`	Другой compose-файл

Git

Команда	Что делает
`git switch -c feat/12-name`	Новая ветка
`git add -A` / `git commit -m "feat: ..."`	Зафиксировать
`git push -u origin feat/12-name`	Отправить ветку
`git fetch` / `git merge origin/main`	Подтянуть main в свою ветку
`git log --oneline --graph -15`	История
`git restore <файл>`	Отменить незакоммиченные правки файла
`git revert <sha>`	Отменить коммит новым коммитом
`git tag v1.0.0` / `git push origin v1.0.0`	Релизный тег

Префиксы коммитов: `feat:` `fix:` `docs:` `ci:` `chore:` `refactor:` `test:`.

GitHub Actions

```yaml
on: { pull_request: { branches: [main] }, push: { branches: [main] } }
jobs:
  test:
    runs-on: ubuntu-latest            # машина GitHub
    steps:
      - uses: actions/checkout@v7     # готовый action
      - run: dotnet test              # команда
  image:
    needs: test                       # после test
    if: github.event_name == 'push'   # только на push, не на PR
    runs-on: ubuntu-latest
    steps:
      - run: echo build image
```

Контексты: `${{ github.sha }}`, `${{ github.ref }}`, `${{ secrets.NAME }}`, `${{ needs.job.outputs.x }}`.

Деплой и откат вручную

```powershell
cd C:\actions-runner; ./run.cmd                    # запустить self-hosted runner
docker image ls ghcr.io/<you>/demo-api             # какие версии есть локально
$env:API_IMAGE = "ghcr.io/<you>/demo-api"; $env:API_TAG = "sha-<нужный>"
docker compose up -d --no-build --wait             # откат на выбранный образ
Remove-Item Env:API_IMAGE, Env:API_TAG
```

Диагностика контейнера: порядок действий

1. `docker compose ps -a` — жив ли, healthy ли, сколько рестартов
2. `docker compose logs api --tail 100` — что писал перед смертью
3. `docker events --since 10m --filter container=demo-api-1` — `die exitCode=?`, `oom`
4. `docker inspect demo-api-1 --format '{{json .State}}'` — состояние и health-лог
5. `docker stats --no-stream` — память и CPU против лимитов

Exit code	Значит
0	Штатное завершение (в т.ч. после `docker stop`)
1	Ошибка приложения (исключение, `Environment.Exit(1)`)
137	SIGKILL: OOM-killer или не уложился в grace period
143	SIGTERM, процесс не обработал сигнал сам

PromQL

Запрос	Что показывает
`up{job="demo-api"}`	1 — Prometheus достаёт метрики, 0 — нет
`sum by (code) (rate(http_requests_received_total[1m]))`	RPS по кодам ответа
`sum by (controller, action) (rate(http_requests_received_total[1m]))`	RPS по методам контроллеров
`sum(rate(http_requests_received_total{code=~"5.."}[1m])) / sum(rate(http_requests_received_total[1m]))`	Доля ошибок сервера
`histogram_quantile(0.95, sum by (le, action) (rate(http_request_duration_seconds_bucket[1m])))`	p95 latency по действиям
`sum(increase(http_requests_received_total{controller="Tasks", action="Create", code="201"}[1m]))`	Создано задач за минуту
`container_memory_working_set_bytes{name="demo-api-1"}`	Память контейнера
`container_memory_working_set_bytes{name="demo-api-1"} / container_spec_memory_limit_bytes{name="demo-api-1"} * 100`	Память в % от лимита
`rate(container_cpu_usage_seconds_total{name="demo-api-1"}[1m])`	CPU, ядер

Порты учебного стенда

Сервис	URL
Сервис задач (API)	http://localhost:8080
Swagger — запросы к API из браузера (Development)	http://localhost:8080/swagger
PostgreSQL	localhost:5432 (или `DB_PORT`)
Prometheus	http://localhost:9090
Grafana	http://localhost:3000 (admin / admin)
cAdvisor	http://localhost:8081
Dozzle (логи)	http://localhost:8888