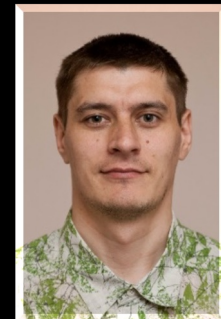


DOT  
NEXT



Анатолий Кулаков

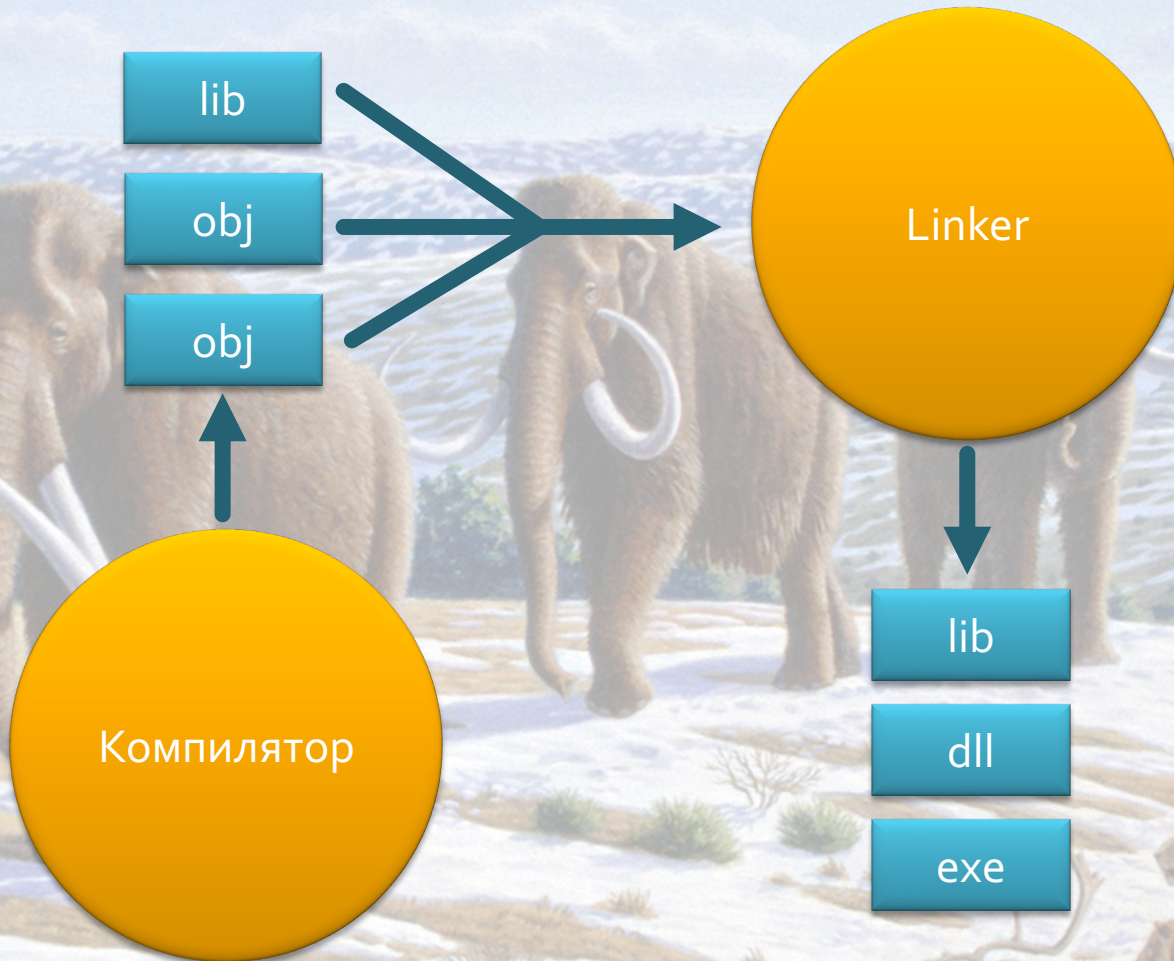
# Build as Code





Разговоры на тему .NET во  
всех его проявлениях,  
новости, статьи,  
библиотеки, конференции,  
личности и прочее  
интересное из мира IT.





# Makefile

```
edit: main.o kbd.o command.o display.o  
    cc -o edit main.o kbd.o command.o display.o
```

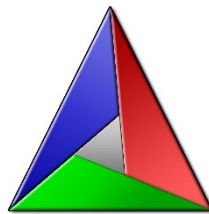
```
main.o: main.c defs.h  
    cc -c main.c
```

```
kbd.o: kbd.c defs.h command.h  
    cc -c kbd.c
```

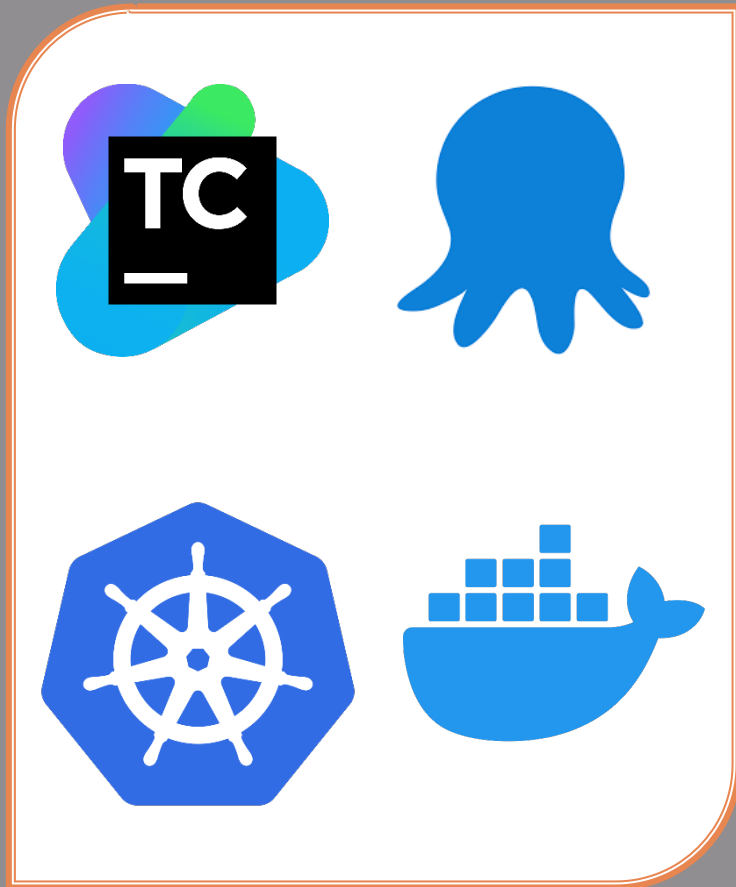
```
command.o: command.c defs.h command.h  
    cc -c command.c
```

```
display.o: display.c defs.h  
    cc -c display.c
```

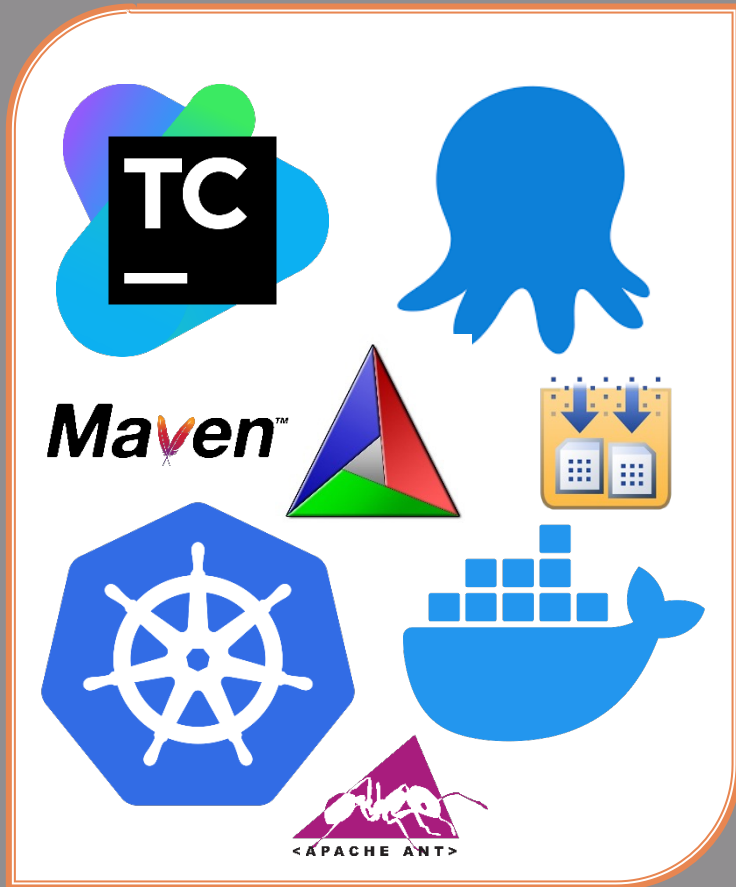
```
clean:  
    rm edit main.o kbd.o command.o display.o
```



# Сисадмин



# Девопс



# Нет такой профессии

## DevOps — это набор инструментов и практик

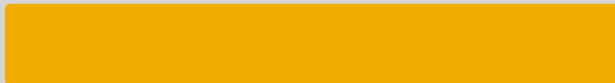
которые помогают автоматизировать и интегрировать процессы между командой разработчиков и командой, ответственной за инфраструктуру, чтобы они могли быстрее и надежнее собирать, тестировать и выпускать релизы.

# Парадокс

Разработчики  
создают

Сисадмины  
разгребают

## Разработчики



- Пишут код
- Пишут тесты
- Определяют артефакты

## Сисадмины



- Собирают код
- Запускают тесты
- Строят артефакты



# Плачут Сисадмины

- Зависимости (SDK и библиотеки)
- Структура проекта
- Артефакты
- Настройка и диагностика CI pipeline
- Узкое горлышко

# Плачут Разработчики

- Устаревающий инструментарий сборки
- Магические артефакты
- Неизвестное рабочее окружение
- Операционная система

# Продуктовая разработка

- Commit в master
- Понимание продукта
- Жизненный цикл
- Качество работы

# Плачет Бизнес

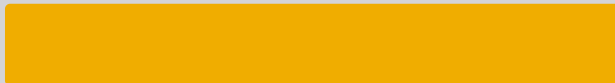


# Парадокс

Разработчики  
создают

Сисадмины  
разгребают

## Разработчики



- Пишут **код**
- Пишут **тесты**
- Определяют **артефакты**

## Сисадмины



- Собирают **код**
- Запускают **тесты**
- Строят **артефакты**

Разработчики  
отвечают за сборку

## Разработчики

Пишут код

Пишут тесты

Определяют артефакты

Собирают код

Запускают тесты

Строят артефакты



Jump to...

Matrix / Trunk

## Integration Tests ★

<Active branches> 10 info items

Overview More

Builds Trends

### DSL Converters Tests ★

#11881	master	Tests passed: 14, ignor...	Trinity:1	teamcity-linu... 0-506	an hour ago	10m:06s	Run ...
--------	--------	----------------------------	-----------	------------------------	-------------	---------	---------

### DSL UI Tests ★

#10811	master	Tests passed: 111	Trinity:1	teamcity-linu... 0-498	40 minutes ago	35m:14s	Run ...
--------	--------	-------------------	-----------	------------------------	----------------	---------	---------

### IntegrationBuild (HSQLDB Incremental) ★

#28192	master	Tests passed: 746, igno...	Oracle:1	teamcity-win... 42-87	2h:15m left		Run ...
#28191	master	Tests passed: 6430, ign...	Architect:1	teamcity-win... 0-176	1h:30m left		Run ...
#28190	master	Tests failed: 1 (1 new), ...	Trinity:1	teamcity-win... 0-170	44m:47s left		Run ...
#28188	master	Tests failed: 3 (3 new), ...	Oracle:1	teamcity-win... 42-89	16m:26s left		Run ...
#28187	master	Tests failed: 1 (1 new), ...	Architect:1	teamcity-win... 0-172	4m:24s left		Run ...
#28185	master	Tests failed: 3 (3 new), ...	Architect:2	teamcity-win... 0-174	a few seconds ago	2h:53m	Run ...
#28151	user_id_sequence	Tests failed: 29 (27 ne...	Neo:1	teamcity-win... 0-175	4 days ago	2h:41m	Run ...
#28137	whiterabbit/master	Tests failed: 1 (1 new), ...	Smith:3	teamcity-win... 0-165	4 days ago	3h:21m	Run ...

### IntegrationBuild (Security) ★

#21053	master	Tests passed: 331	Trinity:1	teamcity-win... 42-87	an hour ago	33m:38s	Run ...
#21022	user_id_sequence	Tests passed: 331	Neo:1	teamcity-linu... 0-503	4 days ago	23m:46s	Run ...

- Matrix
  - Sign-Code-RunnerTemplate
  - Notify slack duty assignee
- Trunk
  - Compile
  - BuildDist +16
  - BuildDist (tar.gz) +16
  - BuildServer merge +16
  - BuildServer scheduled merge +100
  - MasterBuild (BuildDist + All Tests)
  - All Tests
  - Open API Diff
  - Publish Zion Plugin to repository
  - Publish nightly +100
  - Publish Open API Release
  - Sample plugin build +4
  - BuildDist (docker) +100
  - Publish Integration Tests Artifacts +4
  - Tag Sources +100
- Integration Tests
  - DSL Converters Tests
  - DSL UI Tests
  - IntegrationBuild (HSQLDB Incre... 5
  - IntegrationBuild (Security)
  - IntegrationBuild (MS SQL) +13
  - IntegrationBuild (MySQL/... 1
  - IntegrationBuild (MySQL/Docke... 3
  - IntegrationBuild (MariaDB/Docker) ↓2
  - IntegrationBuild (Oracle/Docker) ↓2



## Build

General Settings

Version Control Settings 1

**Build Steps 2**

Triggers 1

Failure Conditions

Build Features

Dependencies

Parameters

Agent Requirements

Suggestions 1

« Hide unconfigured

Last edited moments ago  
by Matthias Koch (view history)

2 build steps added.

### Build Steps

In this section you can configure the sequence of build steps to be executed. Each build step is represented by a build runner and provides integration with a specific build or test tool. ?

+ Add build step

Reorder build steps

Auto-detect build steps

Build Step	Parameters	Description	
1. .NET	build MyFirstProject.sln	Execute: If all previous steps finished successfully	Edit [Menu]
2. .NET	test Library.Tests/Library.Tests.csproj	Execute: If all previous steps finished successfully	Edit [Menu]



## Build Step

[+ Add build s](#)

Runner type:

Powershell

Powershell runner

Step name:

Select icon set from Team City config

You can specify build step name to distinguish it from other steps.

Powershell run mode

x86

Working directory: [?](#)

Optional, specify if differs from the checkout directory.

Script:

Source code

Script source:

Enter Powershell script content:

```
write-host "IconSet value: %IconSet%"
if( "%IconSet%" -eq "Dark"){
  write-host "Dark Icons selected"
  $fileToCreate = "DarkIcons"
  $filesToDelete = @( "YellowIcons", "TransparentBackgroundIcons"
)elseif( "%IconSet%" -eq "TransparentBackground" ){
  write-host "Transparent Background Icons selected"
  $fileToCreate = "TransparentBackgroundIcons"
  $filesToDelete = @( "DarkIcons", "YellowIcons" )
}elseif( "%IconSet%" -eq "Yellow" ){
  write-host "Yellow Icons selected"
  $fileToCreate = "YellowIcons"
```

[Hide](#)

Enter Powershell script. TeamCity references will be replaced in the code

Script execution mode:

Execute .ps1 script with "-File" argument

Specify powershell script execution mode. By default, powershell may not allow executing arbitrary .ps1 files. Select 'Put script into powershell stdin' mode to avoid this issue.

Script arguments:

[Expand](#)

Enter script arguments

Additional command line parameters:

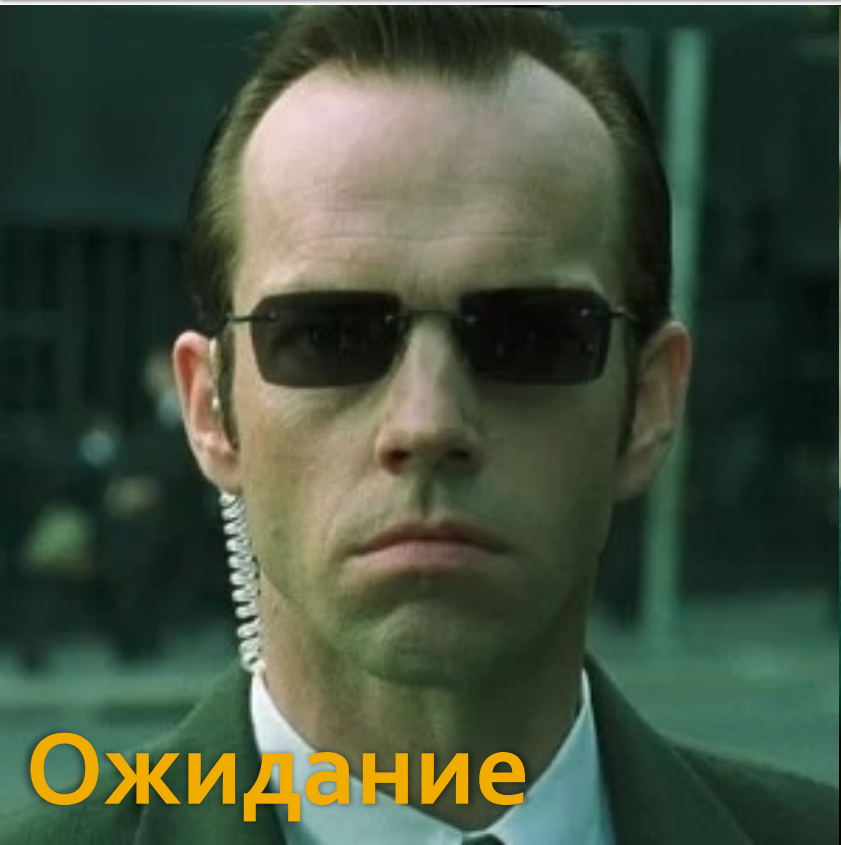
[Expand](#)

Enter additional command line parameters to powershell.exe.

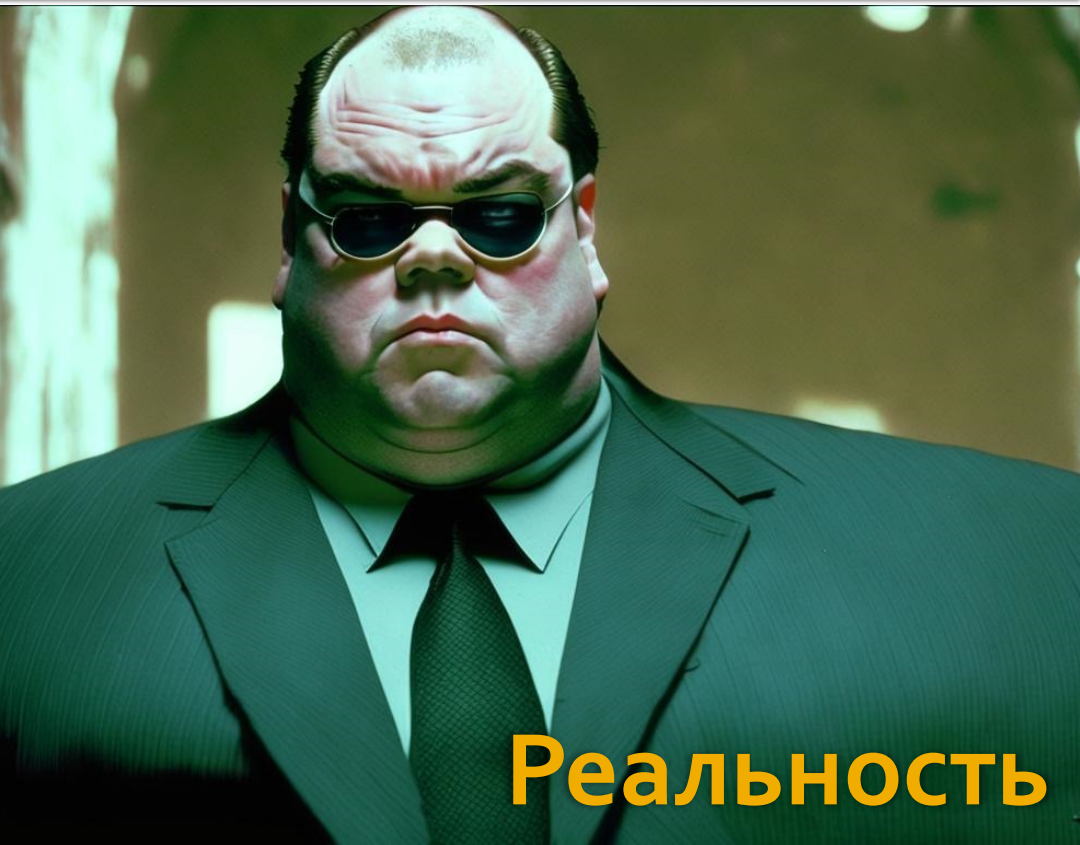
# Где всё?

- Версионирование
- Истории
- Ветвление
- Подсветка кода
- Рефакторинг
- Подсказки
- Code review
- Тестирование
- Отладка
- Переиспользование
- Ограничения и баги
- Vendor lock

# Растолстевшие агенты



Ожидание



Реальность

# Агенты сборки

## УТЯЖЕЛИТЕЛИ

- Все SDK
- Обход багов
- Куча костылей
- Едины для всех

## РИСКИ

- Обновление Агентов
- Обновление SDK
- Новые шаги
- Редактирование шагов

# Магия для сисадминов

- Флаги компиляции
- Переменные окружения
- Используемые версии SDK и зависимостей
- Тестируемые сборки
- Состав новых NuGet пакетов
- Поиск артефактов

# Магия для разработчиков

- Код тестируется не там где работает
- Разработчики не могут сменить SDK
- Отсутствует локальная воспроизводимость

# Повторяемые сборки

- «Интеллектуальный» поиск SDK
- Пути поиска Runtime зависимостей
- Пути поиска пакетов
- Настройки скачивания конфигов
- Операционная система
- Переменные окружения
- Флаги компилятора

# Толстые агенты

- 12 шагов
- Более 300 строк XML
- Более 400 строк PowerShell
- Пропатченные форки стандартных утилит
- Несколько стратегий версионирования
- SQL и базы данных
- Bash, Python, Yaml и RegEx



# Цель

- Упростить процесс сборки
- Уменьшить время выпуска продукта
- Разгрузить команду сисадминов
- Увеличить вовлечённость разработчиков
- Избавиться от vendor lock

# Build as Code

**Всё становится лучше, если оно код**

- Родной для TeamCity способ
- Чужеродный синтаксис
- Не для ручного редактирования
- Vendor lock

```
buildType(Maven( name: "Build", goals: "clean compile"))
parallel { this: CompoundStage
    buildType(Maven( name: "Fast Test", goals: "clean test", runnerArgs: "-Dmaven.test.failure.ignore="
    buildType(Maven( name: "Slow Test", goals: "clean test", runnerArgs: "-Dmaven.test.failure.ignore="
}
buildType(Maven( name: "Package", goals: "clean package", runnerArgs: "-DskipTests"))
}.buildTypes()

bts.forEach { buildType(it) }
bts.last().triggers { this: Triggers
    vcs { this: VcsTrigger

}
}
}
```

```
class Maven(name: String, goals: String, runnerArgs: String? = null) : BuildType({ this: BuildType
    id(name.toExtId())
    this.name = name

    vcs { this: VcsSettings
        root(DslContext.settingsRoot)
    }
}
```

# Все уже там

```
- name: Restore dependencies
run: |
  dotnet restore
  dotnet restore ./src/All/All.csproj

- name: Build
run: dotnet build --no-restore --configuration Debug --nologo -p:Version=${{ env.version }}

- name: Test
run: dotnet test --no-build --configuration Debug --nologo --collect:"XPlat code coverage"

- name: Upload coverage to Codecov
if: matrix.os == 'ubuntu-latest'
uses: codecov/codecov-action@v1

- name: Pack
run: |
  dotnet pack --output ./artifacts --no-build --configuration Debug --nologo -p:PackageVersion=${{ env.version }}
  dotnet pack ./src/All/All.csproj --output ./artifacts --no-build --configuration Debug --nologo -p:PackageVersion=${{ env.version }}

- name: Save
if: matrix.os == 'ubuntu-latest'
uses: actions/upload-artifact@v2
with:
  name: Artifacts-${{ env.version }}
  path: |
    ./artifacts
    !**/*.snupkg
```



# Vendor lock





Я сделаю свой CI  
с Git'ом и Docker'ом

# CI — это просто!

```
dotnet restore
```

```
dotnet build
```

```
dotnet test
```

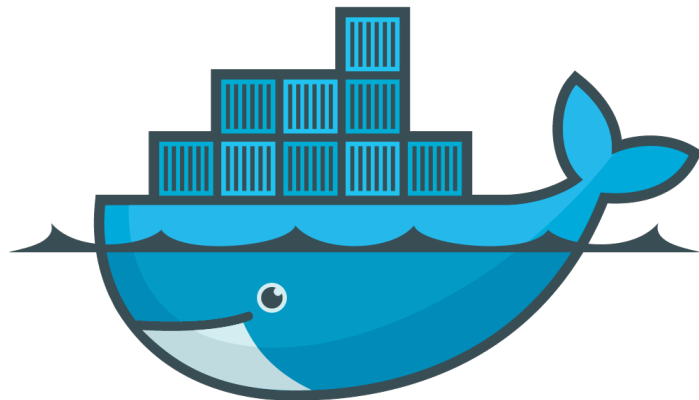
```
dotnet pack
```

```
dotnet publish
```

# Окружение

- Операционная система
- Системные пакеты
- Зависимости
- Конфигурация
- Переменные окружения
- Версии





docker

# Плюсы контейнеризации

- Предсказуемость окружения
- Предсказуемость артефактов
- Тестирование в среде обитания
- Разработчики контролируют SDK и Runtime
- Явный контейнер для E2E тестирования
- Локальный запуск

# Минусы контейнеризации

- Порог входа в технологию
- Непривычный синтаксис
- Ограниченный язык

# Конвейер

- Вычисление версий
- Расчёт путей
- Выгрузка артефактов
- Управление секретами
- Интеграции в системы сборки и оповещения
- Отчёты о тестировании, покрытии, качестве



- NoYaml
- Ошибки в спецификации
- Очень хрупко
- Не тестируемо
  
- Условия, функции, циклы
- Внешние библиотеки
- CI/CD — не декларативная задача

```
- name: Restore dependencies
  run: |
    dotnet restore
    dotnet restore ./src/All/All.csproj

- name: Build
  run: dotnet build --no-restore --configuration Debug --nologo -p:V

- name: Test
  run: dotnet test --no-build --configuration Debug --nologo --colle

- name: Upload coverage to Codecov
  if: matrix.os == 'ubuntu-latest'
  uses: codecov/codecov-action@v1

- name: Pack
  run: |
    dotnet pack --output ./artifacts --no-build --configuration Debug
    dotnet pack ./src/All/All.csproj --output ./artifacts --no-build

- name: Save
  if: matrix.os == 'ubuntu-latest'
  uses: actions/upload-artifact@v2
  with:
    name: Artifacts-${{ env.version }}
    path: |
      ./artifacts
      !**/*.snupkg
```

# PowerShell

<https://microsoft.com/...>



- Системное администрирование
- Управление инфраструктурой
- Низкий порог входа
- Быстрое прототипирование
- Родное .NET окружение

```
$major = 1  
$minor = 2  
$patch = $Counter
```

```
$preRelease = Format-PreReleaseSuffix `   
    -Branch $Branch `   
    -PullRequestNumber $PullRequestNumber `   
    -PullRequestSourceBranch $PullRequestSourceB
```

```
$assemblyVersion = "${major}.${minor}"  
$version = "${major}.${minor}.${patch}${preRelea  
$fileVersion = "${major}.${minor}.${patch}"  
$informationalVersion = "${version}+commit.${Vcs
```

```
$DockerImageNameTag = "${DockerRepositoriesUrl.
```

```
Set-TeamCityBuildNumber -Number $version
```

```
Write-Information "Detailed Version: $informatio  
Write-Information "Docker image name: $DockerIma
```

```
$isPullRequest = [String]::IsNullOrEmpty($P
```



- Мощь PowerShell
- Специально для CI

```
function Clean {
    [DependsOn('Init')]
    [cmdletbinding()]
    param()

    if (Test-Path -Path $outputModVerDir) {
        Remove-Item -Path $outputModVerDir -Recurse -Force > $null
    }
}

function Build {
    [DependsOn('Clean')]
    [cmdletbinding()]
    param()

    if (-not (Test-Path -Path $outputDir)) {
        New-Item -Path $outputDir -ItemType Directory > $null
    }
    New-Item -Path $outputModVerDir -ItemType Directory > $null
    Copy-Item -Path (Join-Path -Path $sut -ChildPath *) -Destination $ou
    Copy-Item -Path (Join-Path -Path $PSScriptRoot -ChildPath 'examples'
```



- Мощь F#
- Специально для CI

```
Target.create "Clean" (fun _ ->
    !! "src/**/*.bin" |> Shell.cleanDirs

    let fakeRuntimeVersion =
        typeof<Fake.Core.Context.FakeExecutionContext>.Assembly.GetName().Version

    printfn "fake runtime %0" fakeRuntimeVersion

    if fakeRuntimeVersion < new System.Version(5, 10, 0) then
        printfn "deleting obj directories because of https://github.com/fsprojects/"
        !! "src/**/*.obj" |> Shell.cleanDirs
        // Allow paket to do a full-restore (to improve performance)
        Shell.rm ("paket-files" </> "paket.restore.cached")
        callPaket "." "restore"

    Shell.cleanDirs
    [ nugetDncDir
      collectedArtifactsDir ]
```





- Ужасный синтаксис
- Куча магии
- Плохая документация
- Нет поддержки IntelliSense
- Тупиковая концепция
- Cake Frosting
- Самый популярный вариант

```
Task("Clean")
    .Does<BuildParameters>((context, parameters) =>
    {
        CleanDirectories("./src/**/bin/" + parameters.Configuration);
        CleanDirectories("./src/**/obj");
        CleanDirectories(parameters.Paths.Directories.ToClean);
    });

Task("Restore-NuGet-Packages")
    .IsDependentOn("Clean")
    .Does<BuildParameters>((context, parameters) =>
    {
        DotNetRestore("./src/Cake.sln", new DotNetRestoreSettings
        {
            Verbosity = DotNetVerbosity.Minimal,
            Sources = new [] { "https://api.nuget.org/v3/index.json" },
            MSBuildSettings = parameters.MSBuildSettings
        });
    });
```



- Мощь C#
- Специально для CI
- Полная поддержка IDE
- Честный Debug
- Тестирование
- Всё как в .NET приложении

```
Target Clean => _ => _
    .Before<IRestore>()
    .Executes(() =>
    {
        SourceDirectory.GlobDirectories("*/bin", "*/obj").DeleteIfExists();
        OutputDirectory.CreateOrCleanDirectory();
    });

Configure<DotNetBuildSettings> ICompile.CompileSettings => _ => _
    .When(!ScheduledTargets.Contains(((IPublish)this).Publish) &&
    .ClearProperties());

Configure<DotNetPublishSettings> ICompile.PublishSettings => _ => _
    .When(!ScheduledTargets.Contains(((IPublish)this).Publish) &&
    .ClearProperties());

IEnumerable<(Nuke.Common.ProjectModel.Project Project, string Framework)
    from project in new[] { Solution.Nuke_GlobalTool, Solution.Nuke_WebTool }
    from framework in project.GetTargetFrameworks()
    select (project, framework);
```

Победитель

# PowerShell



- Быстрое прототипирование
- Администрирование и разработка
- Командная строка

Пример

`build.ps1`



# Минусы PowerShell

- ~~Ленивые программисты~~
- Непривычный язык

# Nuke

<https://nuke.build/>



- Мощность C#
- Специально для CI
- Полная поддержка IDE
- Честный Debug
- Тестирование
- Всё как в .NET приложении



# Консольное приложение

```
class Build : NukeBuild
{
    public static int Main ()
        => Execute<Build>(x => x.Compile);
    ...
}
```

# Targets

```
Target Restore => _ => _  
    .Executes(() =>  
    {  
        Log.Information("Restore");  
    });
```

```
Target Compile => _ => _  
    .DependsOn(Restore)  
    .Executes(() =>  
    {  
        Log.Information("Compile");  
    });
```



# Команды

```
[Solution(GenerateProjects = true)]  
Solution Solution;
```

```
DotNetBuild(settings => settings  
    .SetConfiguration(Configuration)  
    .SetProjectFile(Solution.WebApplication1)  
);
```

# Переиспользование

```
interface IDockerBuild : INukeBuild
{
    Target Build => _ =>
        .Executes(() => { /* Implementation */ });
}
```

```
interface INuGetBuild : INukeBuild
{
    Target Publish => _ =>
        .DependsOn<IDockerBuild>(x => x.Build)
        .Executes(() => { /* Implementation */ });
}
```

# Переиспользование

```
class Build : NukeBuild, IDockerBuild, INuGetBuild
{
    Target MyTarget => _ => _
        .DependsOn<INuGetBuild>(x => x.Publish)
        .Executes(() =>
        {
            /* Implementation */
        });
}
```

# Поддержка CI/CD

- AppVeyor
- Azure Pipelines
- Bitbucket
- GitHub Actions
- GitLab
- Jenkins
- TeamCity





# Переменные окружения

```
Target Print => _ => _
  .Executes(() =>
  {
    Log.Information(
      "Branch = {Branch}",
      GitHubActions.Ref);

    Log.Information(
      "Commit = {Commit}",
      GitHubActions.Sha);
  });
```

# Артефакты

```
Target Pack => _ => _  
  .Produces(PackagesDirectory / "*.nupkg")  
  .Executes(() => { /* Implementation */ });
```

## Artifacts

Produced during runtime

Name

Size



packages

29.9 MB



test-results

6.54 MB



# Генерация конфигурации

```
[GitHubActions(  
    "continuous",  
    GitHubActionsImage.UbuntuLatest,  
    On = new[] { GitHubActionsTrigger.Push },  
    InvokedTargets = new[] { nameof(Compile) })]  
  
class Build : NukeBuild { /* ... */ }
```



# Генерация конфигурации

```
name: continuous
```

```
on: [push]
```

```
jobs:
```

```
  ubuntu-latest:
```

```
    name: ubuntu-latest
```

```
    runs-on: ubuntu-latest
```

```
    steps:
```

- uses: actions/checkout@v2
- name: Run './build.cmd Compile'  
 run: ./build.cmd Compile
- uses: actions/upload-artifact@v1  
 with:  
 name: packages  
 path: output/packages

File Edit View Git ReSharper Tools Extensions Search nuk...emo

Solution Explorer

Search Solution Explorer (Ctrl+;)

Solution 'nuke-demo' (3 of 3 projects)

- Build
- Dependencies
- boot
- config
- logs
- Build.cs
- NukeDemo
- NukeDemo.Tests

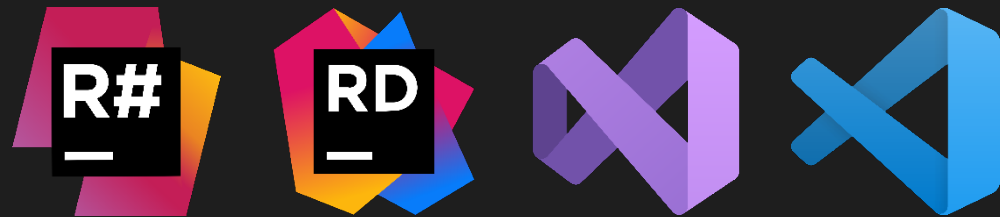
```
class Build : NukeBuild
{
    public static int Main() => Execute<Build>(x => x.Compile);

    Target Compile => _ => _
        .DependsOn(Restore)
        .Executes(() =>
        {
            // ...
        });
}
```

Compile

- To expression lambda
- To named method
- To local function
- Debug
- Run without dependencies
- Debug without dependencies

Ln 11 Col 10 Ch 10 INS 0/0 7 master nuke-demo

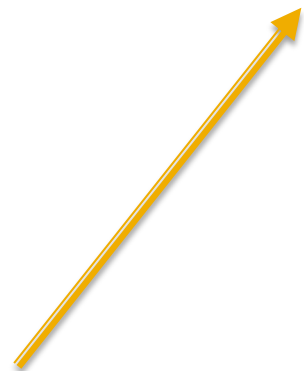
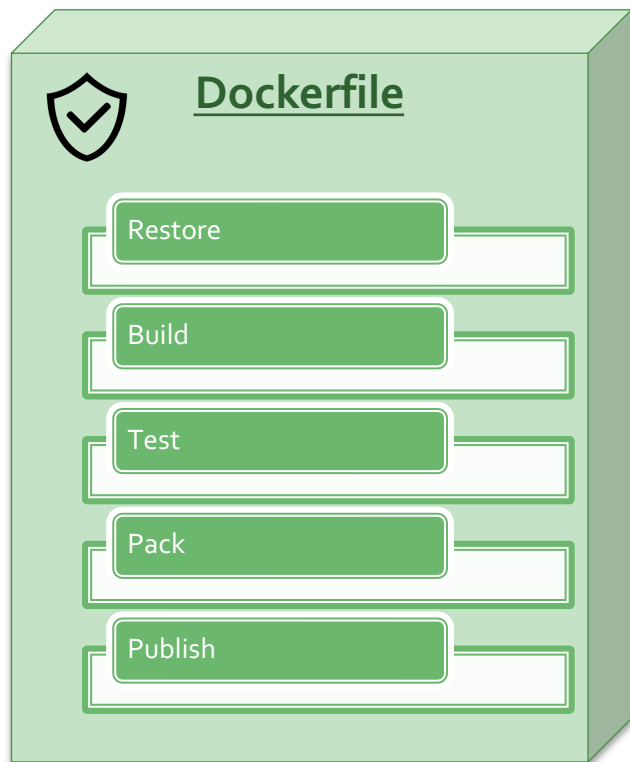


Пример

Nuke



# Процесс сборки



# Плюсы Build as Code

- Полное версионирование всего
- Лучшие инструменты и практики разработки
- Полный контроль над процессом сборки
- Security review и анализаторы
- Изоляция конвейера на уровне веток
- Предсказуемость сборок
- Локальная диагностика
- Максимальное погружение разработчиков
- Никакого vendor lock

# Тонкие агенты

- Десятки шагов превратились в один
- Один базовый шаблон для всех
- Никакого устаревания
- Всегда рабочие билды
- Минимальное внимание DevOps'ов
- Интеграция отчётов качества
- Интеграция артефактов сборки

# Ресурсы

- [docs.docker.com](https://docs.docker.com)
- [nuke.build](https://nuke.build)
- [GitHub.com/KulakovT/Samples](https://GitHub.com/KulakovT/Samples)
- [Kulakov@DotNet.Ru](mailto:Kulakov@DotNet.Ru)