Реактивная сборка

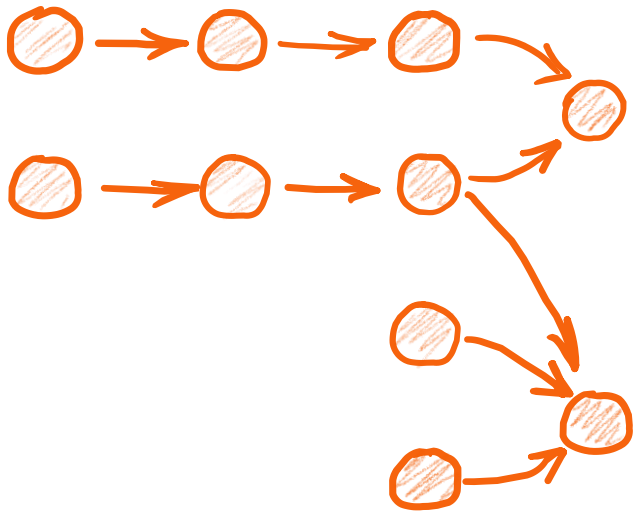
Огромного проекта

Станислав Сидристый

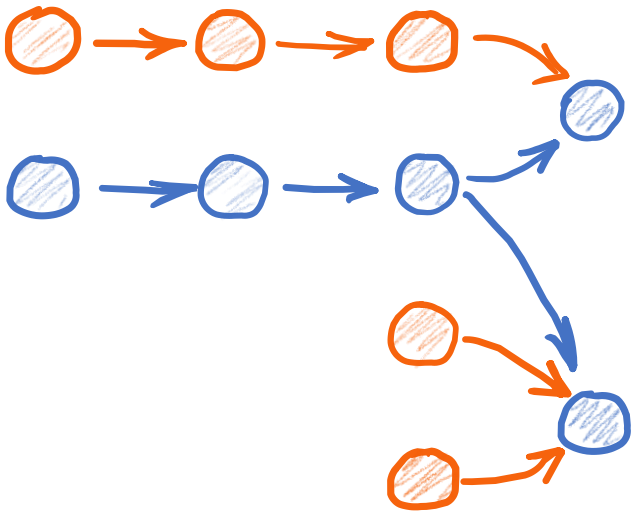
- Системный архитектор в Центре Речевых Технологий
- Книга: https://github.com/sidristij/dotnetbook
- telegram: @sidristij
- sunex.development@gmail.com

Что дано?

- монорепозиторий
- более 100 файлов решений
- более 1000 проектов
- выпуск идёт под общим срезом
- отдельные билды под каждый сервис



- есть общие nuget-пакеты
- есть общие проекты (в рамках группы сервисов)



- есть общие nuget-пакеты
- есть общие проекты (в рамках группы сервисов)

Первичная проблема

- трудно отслеживать что было изменено (nuget+сервисы)
- трудно отслеживать что нужно пересобрать (nuget+сервисы)
- версии сервисов должны расти только если они были изменены
- dev/release версии

Так что же делать?

Можно попробовать сделать обычным .NET путём

- прошлый билд
- *изменения*
- текущий билд
  - msbuild сам всё понимает по датам изменения файлов

Проблемы

- git не сохраняет даты изменения файлов

Решение

```
find . -exec touch -m -d ‘-1 day’ {} +
/// достать откуда-то bin/Release прошлого билда
git diff --name-only PREV-SHA HEAD | xargs -d'\n' touch -m -d
dotnet build -c Release
```

Проблемы

- msbuild нужен не только bin/Release, но и obj/Release
- их объёмы будут под 20Gb на полную сборку
- из-за больших объемов билд будет не супер-быстрым

Можно попробовать сверять bin/Release

Решение

```
Dockerfile билда:
FROM registry/<service>:<version> AS prev-version
FROM dotnetsdk as build
...
+ COPY --from=prev-version app <sources-path>/last
dotnet build -c Release
diff /last /bin/Release/net6.0
```

Проблемы

- из-за многопоточной сборки dll отличаются
- если отключить многопоточную сборку, будет очень долго

Можно попробовать вручную

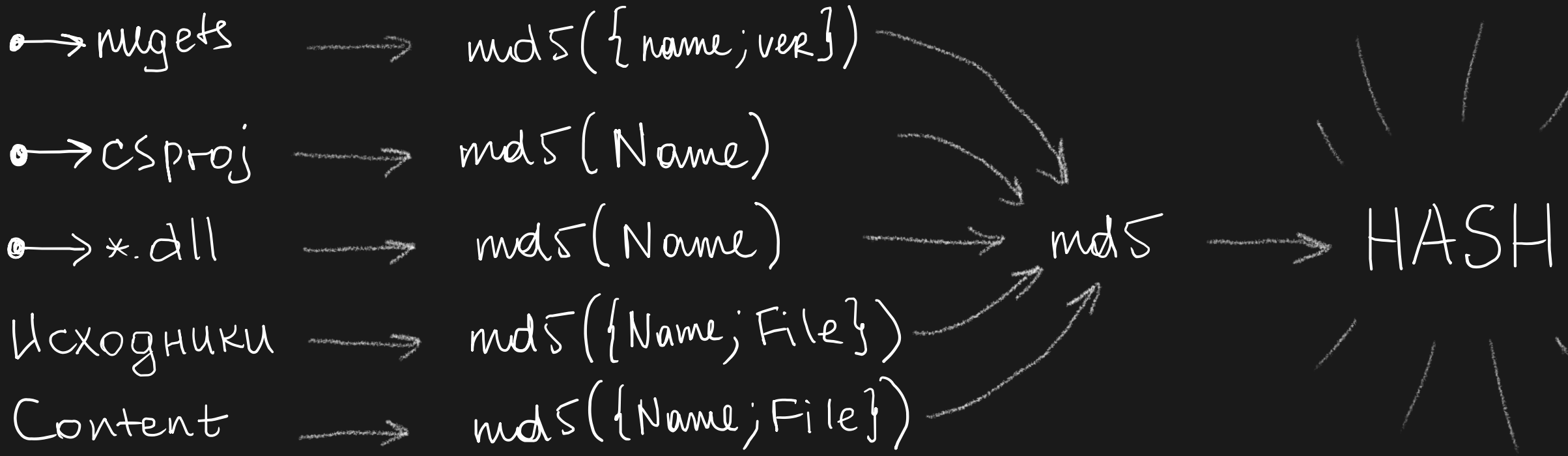
Гипотеза

- Состояние любого проекта зависит от:
  - исходников проекта;
  - зависимостей проекта:
    - других проектов;
    - nuget пакетов;
    - *.dll, *.so;
    - Contents.

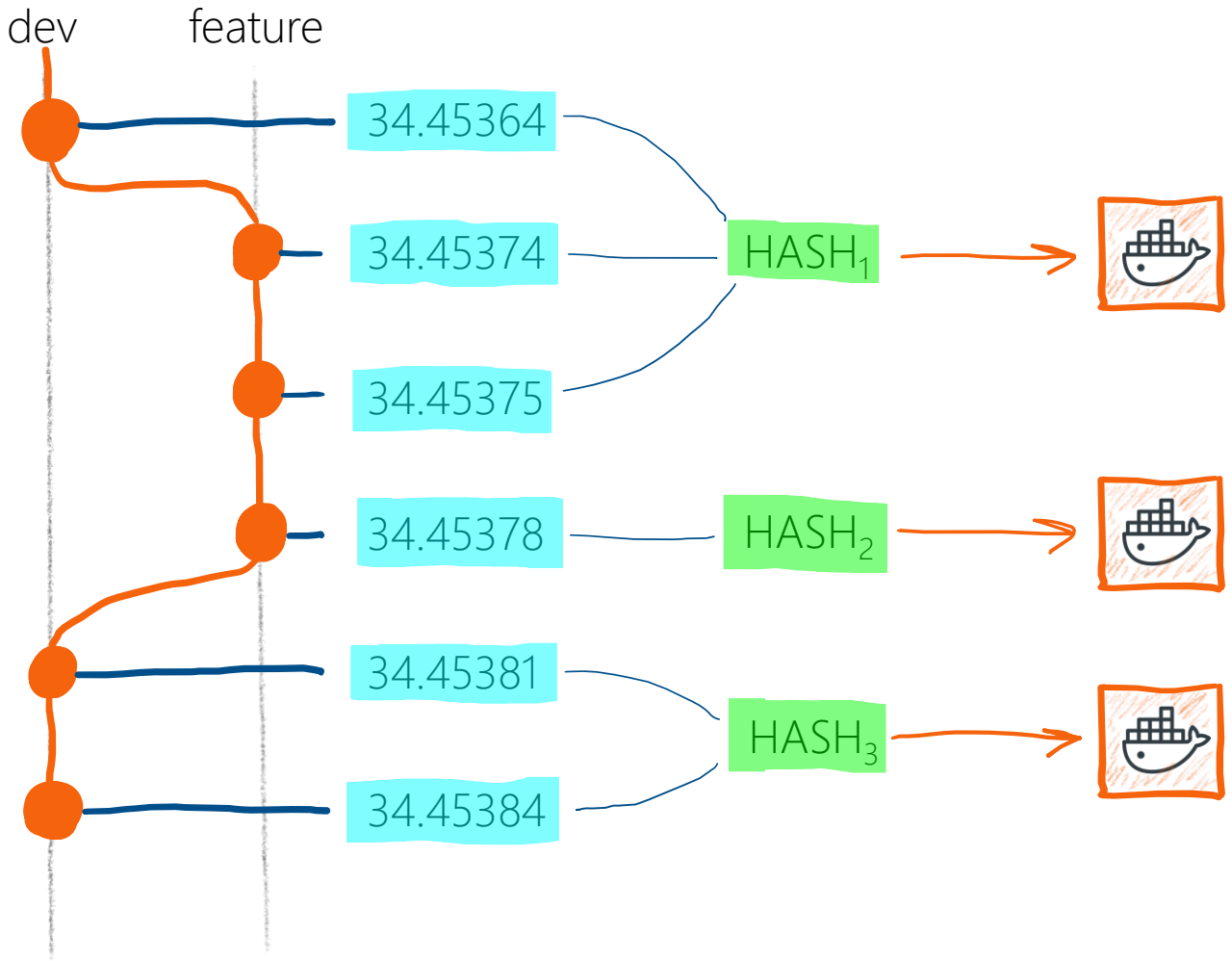
Гипотеза

Чтобы рассчитать «версию» проекта надо:

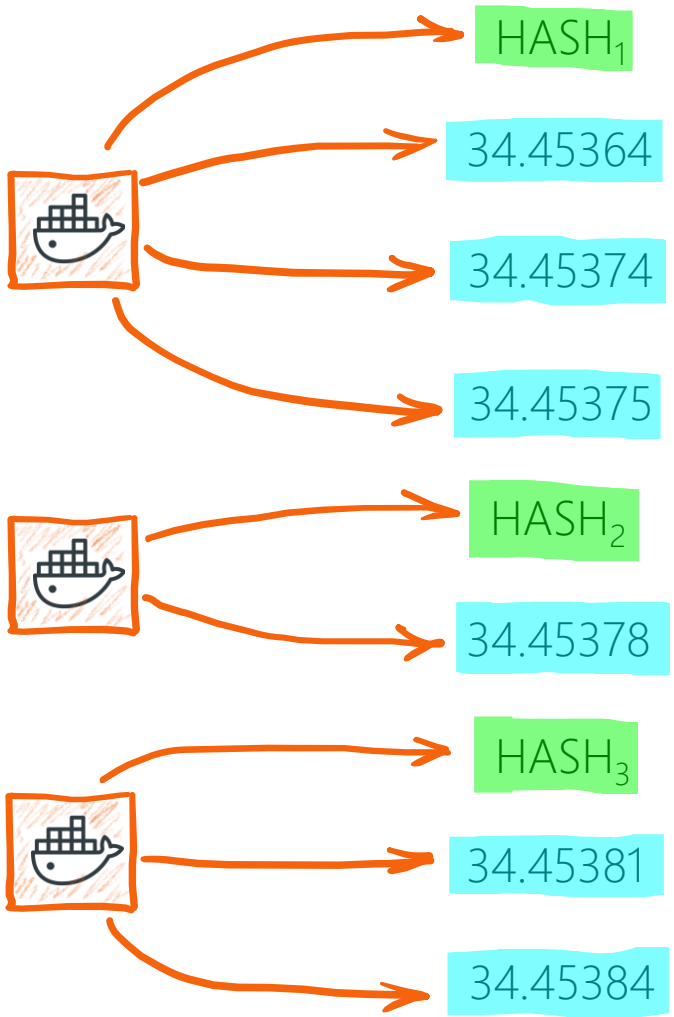
- рассчитать число, которое определит версию исходников и contents проекта
- рассчитать число, которое определит «версии» всех зависимостей


nugets
csproj
*.dll
Исходники
Content
md5({name;ver})
md5(Name)
md5(Name)
md5({Name;File})
md5({Name;File})
md5
HASH

Версии


dev
feature
34.45364
34.45374
34.45375
34.45378
34.45381
34.45384
HASH1
HASH2
HASH3

Версии

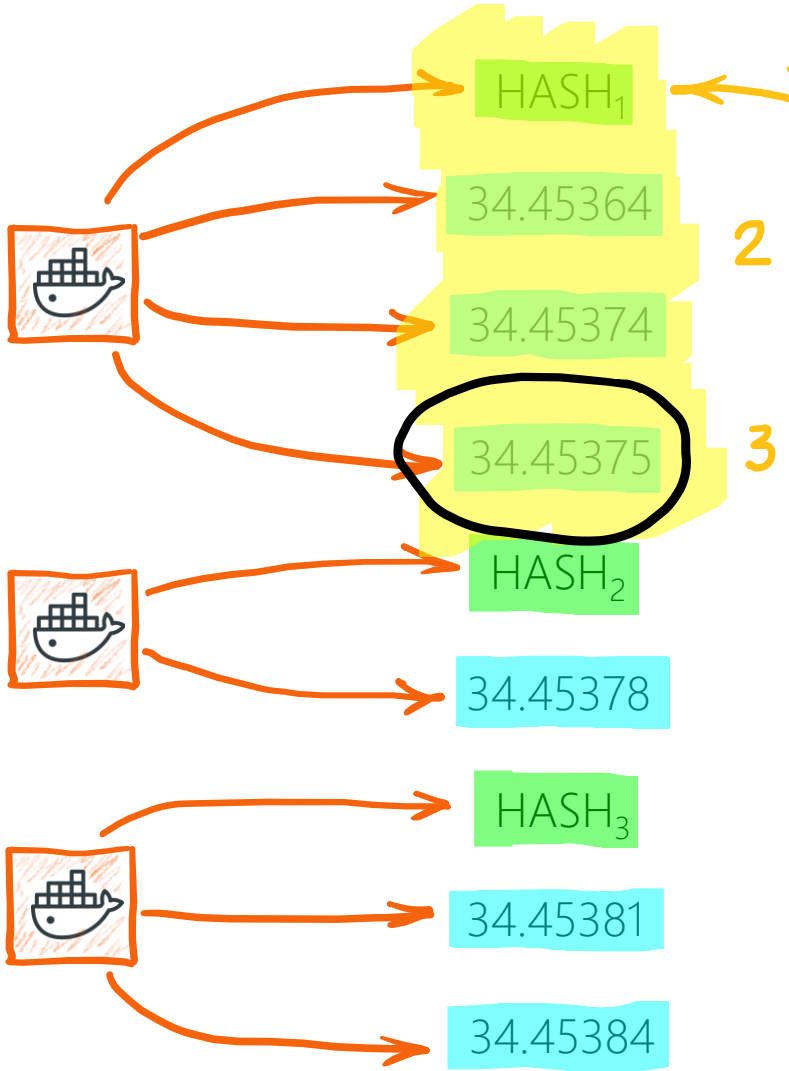


1. Рассчитать HASH-версию сервиса

2. Забрать полный список тэгов

3. Забрать максимальную версию

Версии



1. Рассчитать HASH-версию сервиса

2. Забрать полный список тэгов

3. Забрать максимальную версию

Как считать hash-версию?

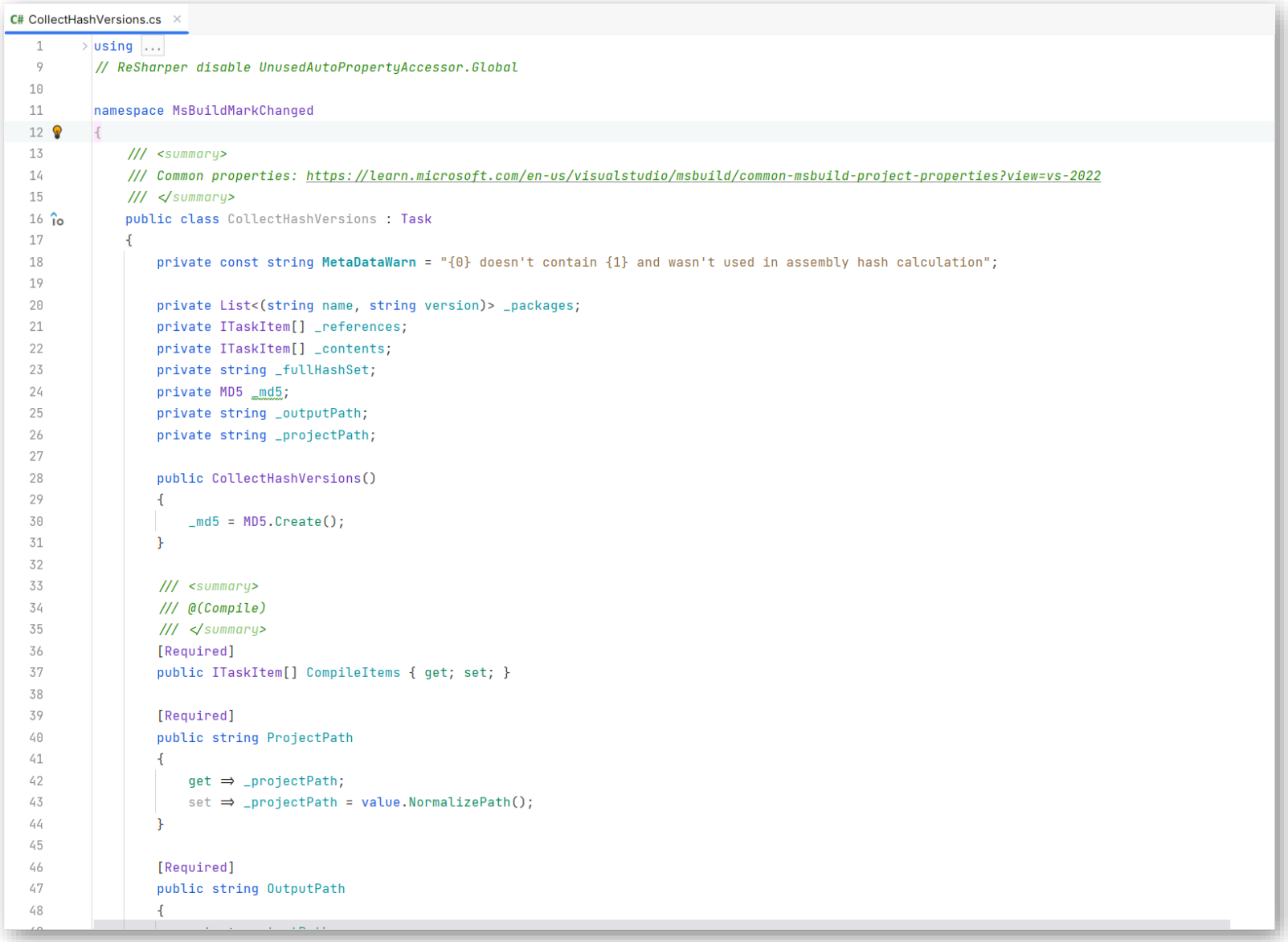
1. Если msbuild запустить на каком-либо проекте, тот сначала соберёт все зависимые проекты, а потом – требуемый;
2. У msbuild есть язык описания шагов сборки.

Что получим

1. Не будет компиляции если она не нужна
2. Меняем версию только если что-то изменилось

План

1. Пишем msbuild Task для подсчёта hash-версии;
2. Пишем ещё один для обхода проектов.


C# CollectHashVersions.cs
using ...
// ReSharper disable UnusedAutoPropertyAccessor.Global
namespace MsBuildMarkChanged
{
/// <summary>
/// Common properties: https://learn.microsoft.com/en-us/visualstudio/msbuild/common-msbuild-project-properties?view=vs-2022
/// </summary>
public class CollectHashVersions : Task
{
private const string MetaDataWarn = "{0} doesn't contain {1} and wasn't used in assembly hash calculation";
private List<(string name, string version)> _packages;
private ITaskItem[] _references;
private ITaskItem[] _contents;
private string _fullHashSet;
private MD5 _md5;
private string _outputPath;
private string _projectPath;
public CollectHashVersions()
{
_md5 = MD5.Create();
}
/// <summary>
/// @(Compile)
/// </summary>
[Required]
public ITaskItem[] CompileItems { get; set; }
[Required]
public string ProjectPath
{
get => _projectPath;
set => _projectPath = value.NormalizePath();
}
[Required]
public string OutputPath
{

Выводы

Выводы

- Вместо линейных 20-25 минут линейные 20 сек – 3 минуты;
- Вместо 400-500 машиноминут 20 секунд – 3 минуты;
- Снижение количества ошибок «забыли собрать nuget»;
- Снижение количества ошибок «забыли собрать сервис»;
- Автоматический деплой среза версий сервисов;
- Версия растёт только если есть изменения;
- Можно удостовериться, что изменения затронули мало сервисов.

QA