Министерство науки и высшего образования Российской Федерации

ФГАОУ ВО «Уральский федеральный университет   
имени первого Президента России Б.Н. Ельцина»

Институт новых материалов и технологий

Кафедра «Теплофизика и информатика в металлургии»

**Автоматизация сборки проекта с помощью Jenkins**

**ОТЧЕТ**

**по практической работе № 4**

**по дисциплине «Основы методологии Development Operation»**

Направление 09.03.02 «Информационные системы и технологии» (уровень бакалавриата)

Образовательная программа  
09.03.02/33.02 «Информационные системы и технологии» (СУОС)

Студент

группы НМТ-413901 Г.Е.Шумин

Преподаватель:

профессор, д.т.н. В.В.Лавров

Екатеринбург

2025

СОДЕРЖАНИЕ

[**Практическая работа «Автоматизация сборки проекта с помощью Jenkins»** 3](#_Toc195760314)

[1.1 Цель работы 3](#_Toc195760315)

[1.2 Ход проведения работы 3](#_Toc195760316)

[1.2.1 Разработка тестового приложения Visual Studio .NET Core 3](#_Toc195760317)

[1.2.2 Создание контейнера с использованием Docker 3](#_Toc195760318)

[1.2.3 Создание файла docker-compose.yml 3](#_Toc195760319)

[1.2.4 Создание файла Jenkinsfile 3](#_Toc195760320)

[1.2.5 Размещение приложения в системе контроля версий GitHub 5](#_Toc195760321)

[1.2.6 Настройка автоматической сборки проекта в Jenkins и интеграции с GitHub 5](#_Toc195760322)

[1.2.7 Выполнение автоматической сборки проекта в Jenkins и анализ результатов 8](#_Toc195760323)

[1.3 Выводы 9](#_Toc195760324)

[Приложение А Листинг программного кода приложения «Калькулятор» (контроллер) 10](#_Toc195760325)

[Приложение Б Листинг программного кода приложения «Калькулятор» (appsettings.json) 11](#_Toc195760326)

[Приложение В Листинг программного кода приложения «Калькулятор» (appsettings.Development.json) 12](#_Toc195760327)

[Приложение Г Листинг программного кода приложения «Калькулятор» (Program.cs) 13](#_Toc195760328)

[Приложение Д Листинг файла Dockerfile 14](#_Toc195760329)

[Приложение Е Листинг файла docker-compose.yml 15](#_Toc195760330)

[Приложение Ж Листинг файла Jenkinsfile 16](#_Toc195760331)

# **Практическая работа «Автоматизация сборки проекта с помощью Jenkins»**

1.1 Цель работы

Цель данной работы – научиться пользоваться процессом непрерывной интеграции программного обеспечения на Jenkins.

1.2 Ход проведения работы

### 1.2.1 Разработка тестового приложения Visual Studio .NET Core

Вернем приложение “15\_Calculator” к моменту, когда была завершена практическая работа №1, для этого просто загрузим соответствующий коммит из ветки master.

### 1.2.2 Создание контейнера с использованием Docker

Создание контейнера в данной работе осуществляется автоматически через docker-compose.yml, настройки которого будут выполнены во время сборки через Jenkins.

### 1.2.3 Создание файла docker-compose.yml

Создадим docker-compose.yml со следующим содержанием (Рисунок 1):



Рисунок 1 – содержание файла docker-compose.yml

### 1.2.4 Создание файла Jenkinsfile

Создадим Jenkinsfile со следующим содержанием (Рисунок 2):

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

Рисунок 2 – содержание файла Jenkinsfile

Данные инструкции будут выполнены при запуске сборки в Jenkins.

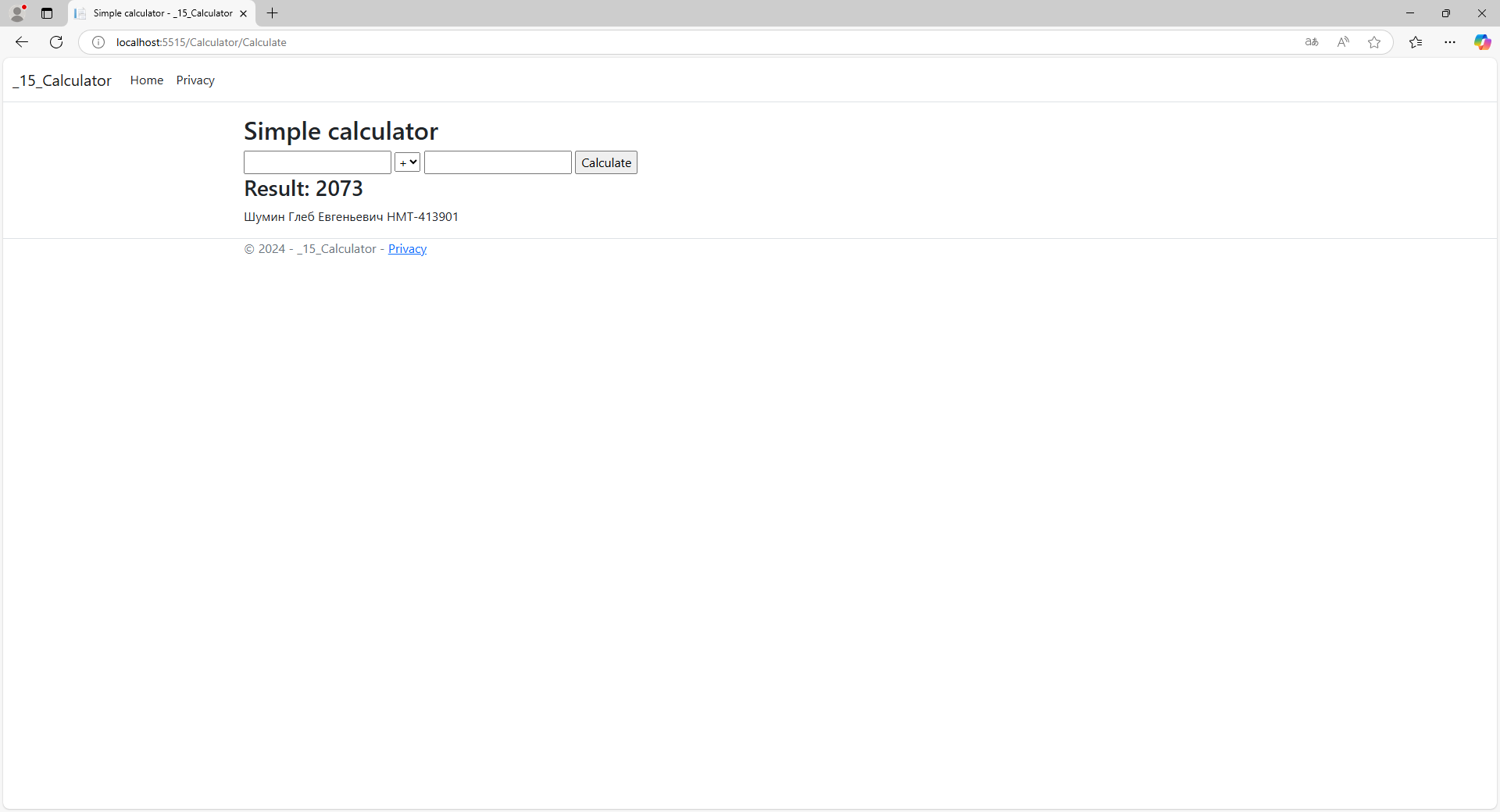


Рисунок 3 – локальная работа приложения

### 1.2.5 Размещение приложения в системе контроля версий GitHub

Разместим наше приложение в отдельной ветке “jenkins”, репозитория “15\_Calculator”, доступного по ссылке:

https://github.com/gshumin2013/15\_Calculator

### 1.2.6 Настройка автоматической сборки проекта в Jenkins и интеграции с GitHub

Перейдем по адресу 93.88.178.186:8080 и создадим новый item с названием “15\_Calculator”, который будет копировать настройки item “Test”.

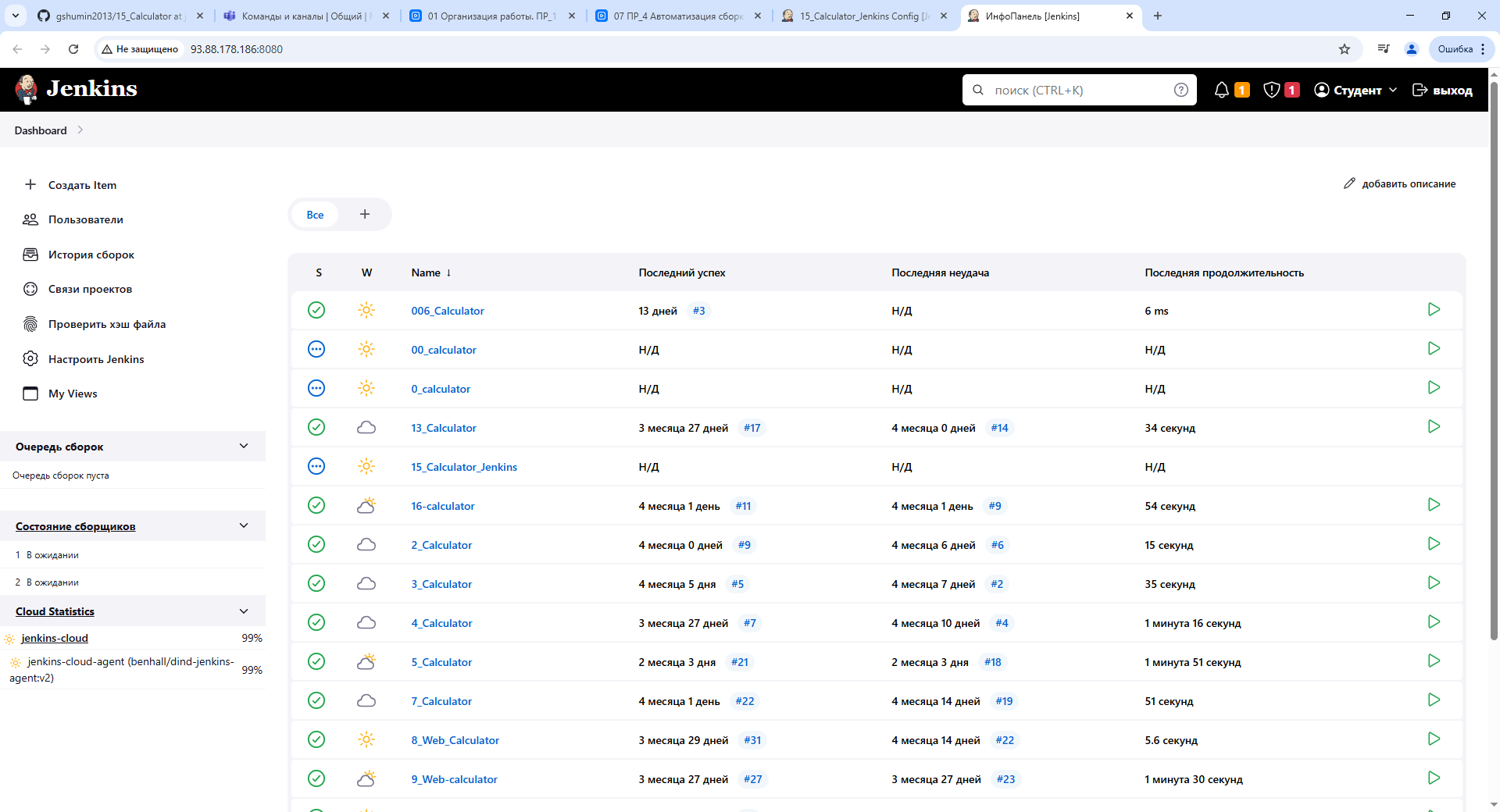


Рисунок 4 – Список существующих Item

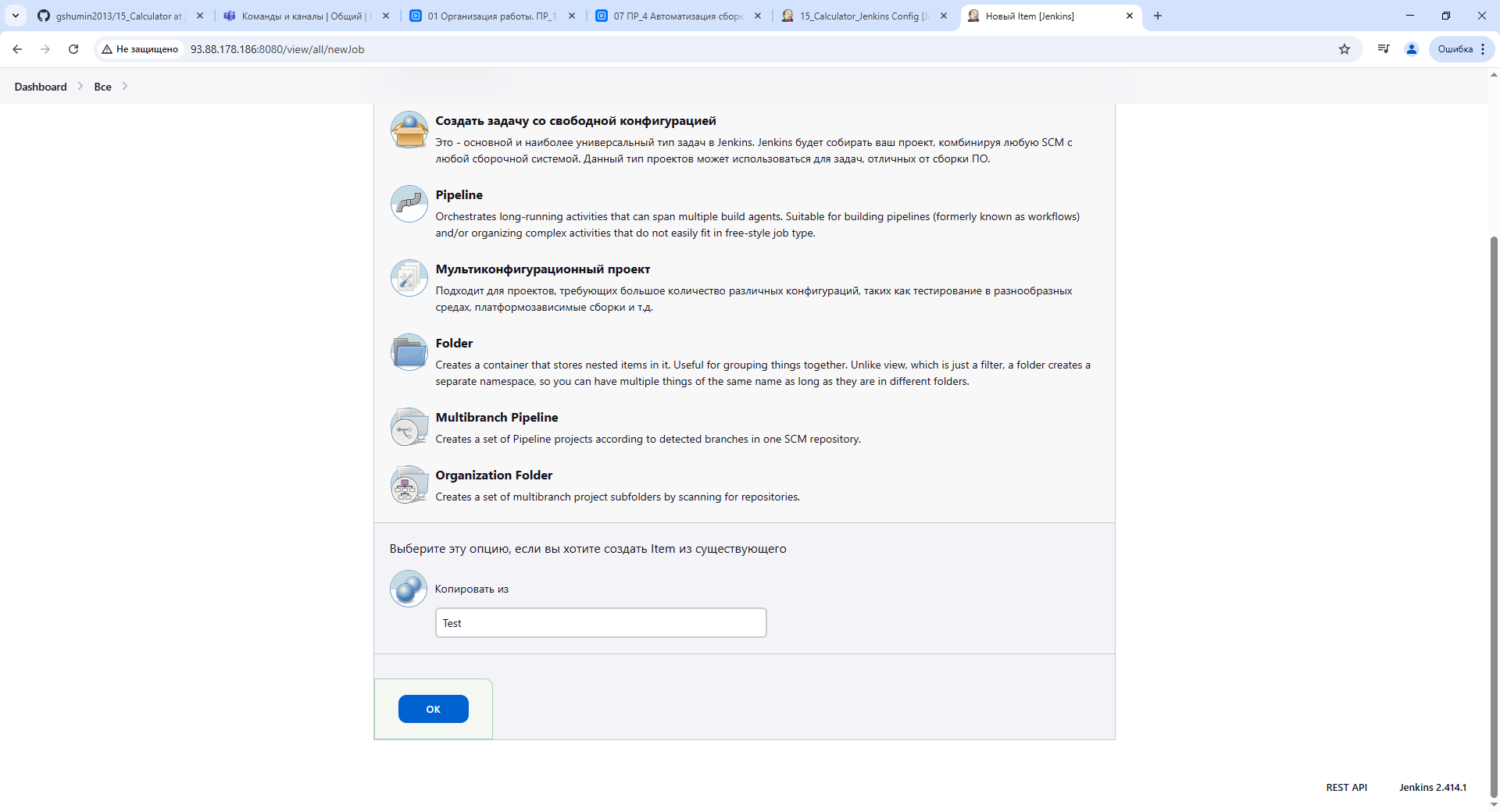
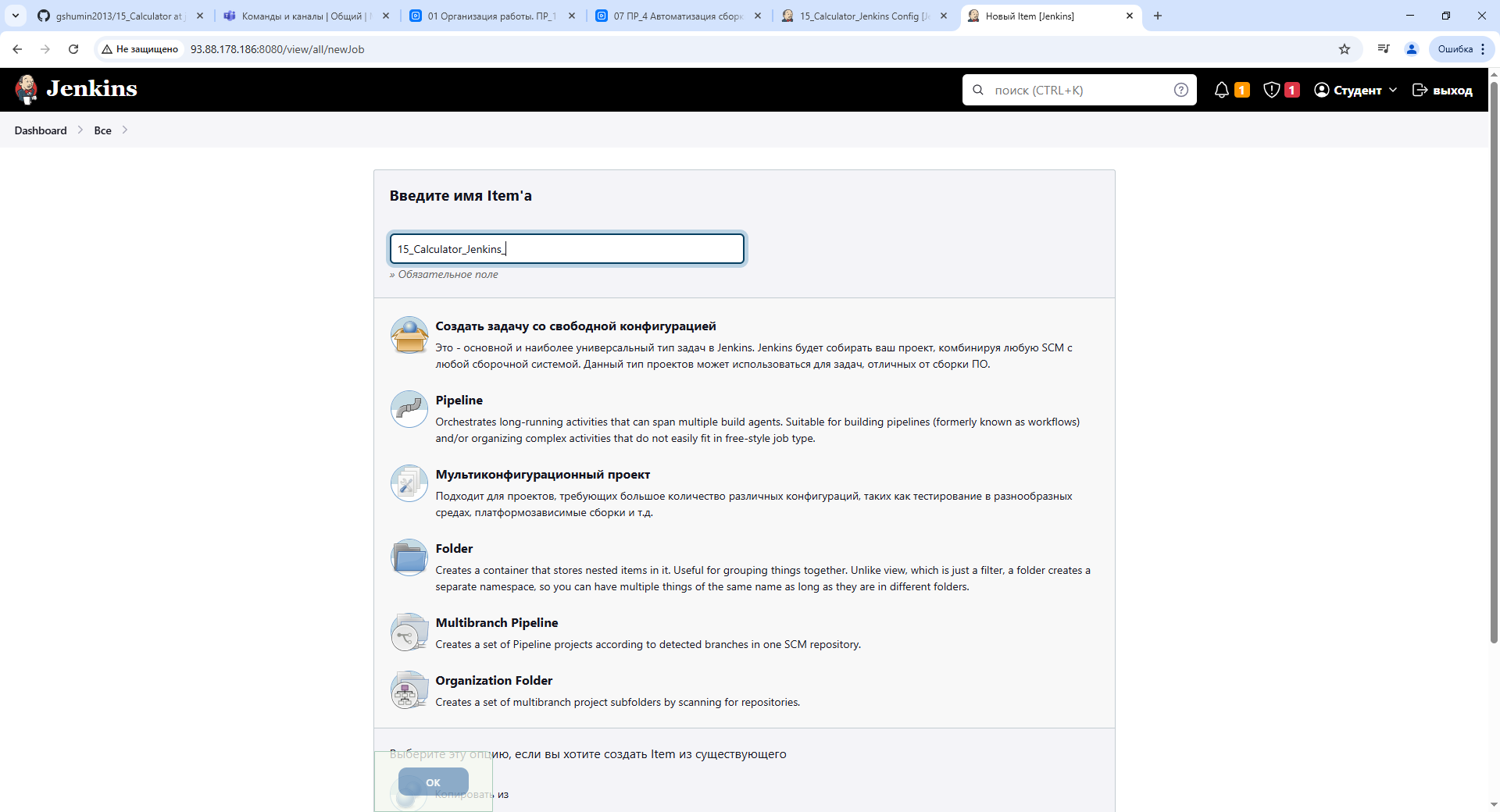


Рисунок 5 – Создание нового Item

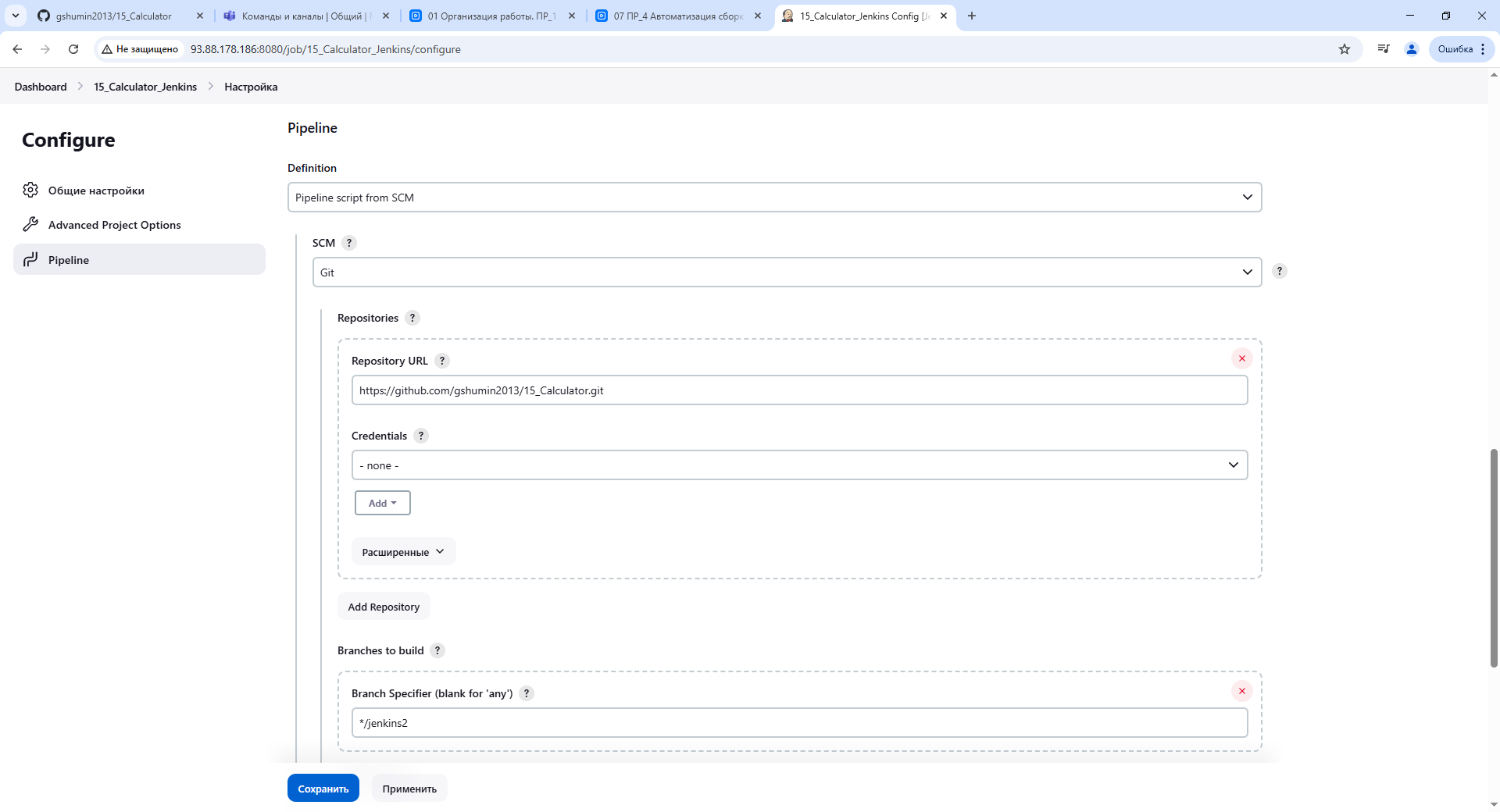
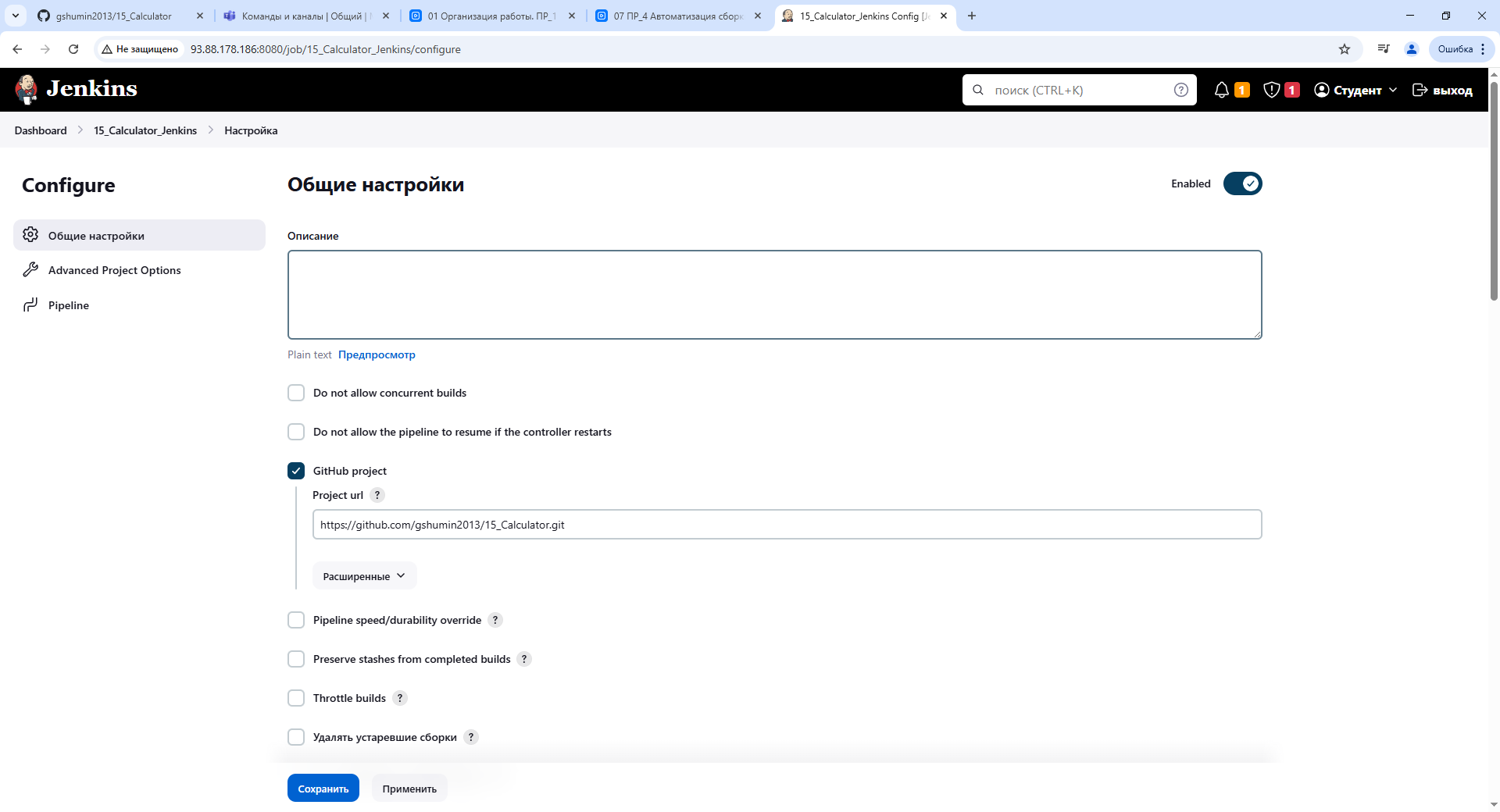


Рисунок 6-7 – Настройка Github

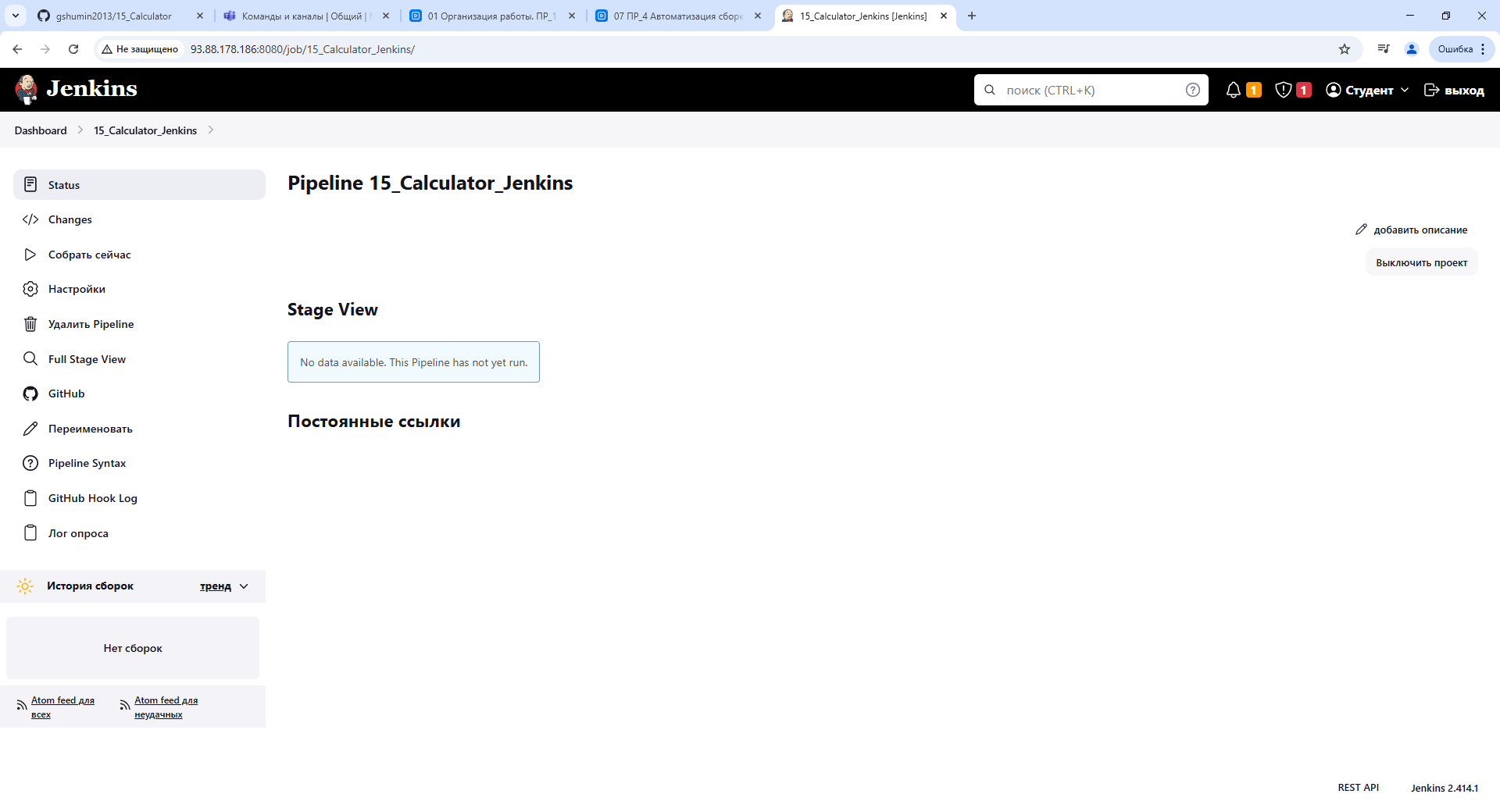


Рисунок 8 – Item был успешно создан

### 1.2.7 Выполнение автоматической сборки проекта в Jenkins и анализ результатов

При попытке запустить сборку сервер очень долго обрабатывал запрос и даже в Putty команды выполнялись с большой задержкой. Вывод на консоль демонстрирует, что проект доходит до стадии запуска, но запуск не происходит и ошибки не выводится. Возможно это вызвано тем, что контейнер с портом 5515 уже существует, однако эта мысль посетила меня слишком поздно.

При каждой попытке собрать контейнер, из GitHub автоматически подтягиваются последние изменения и указывается количество коммитов.

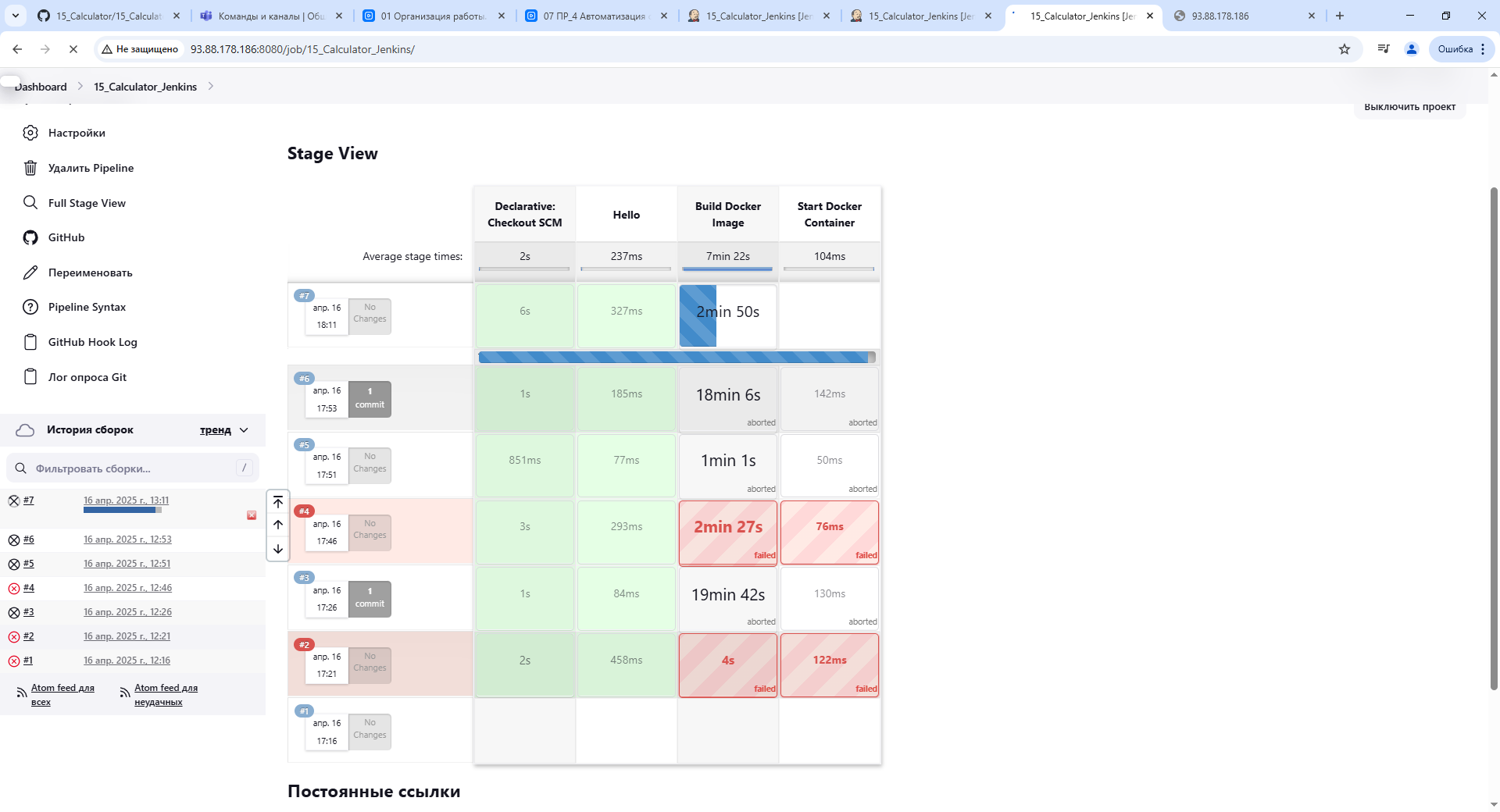


Рисунок 9 – История сборок

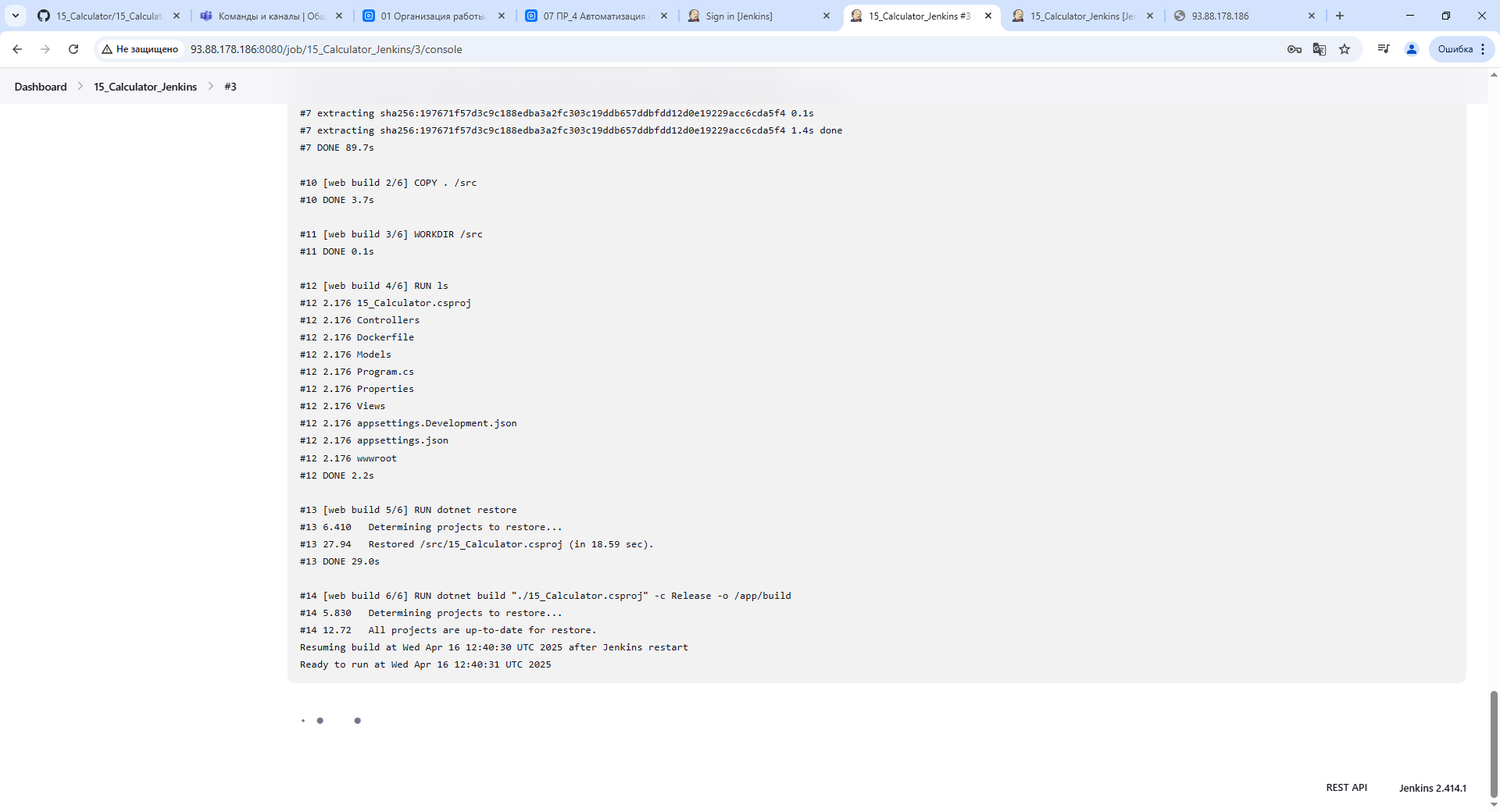
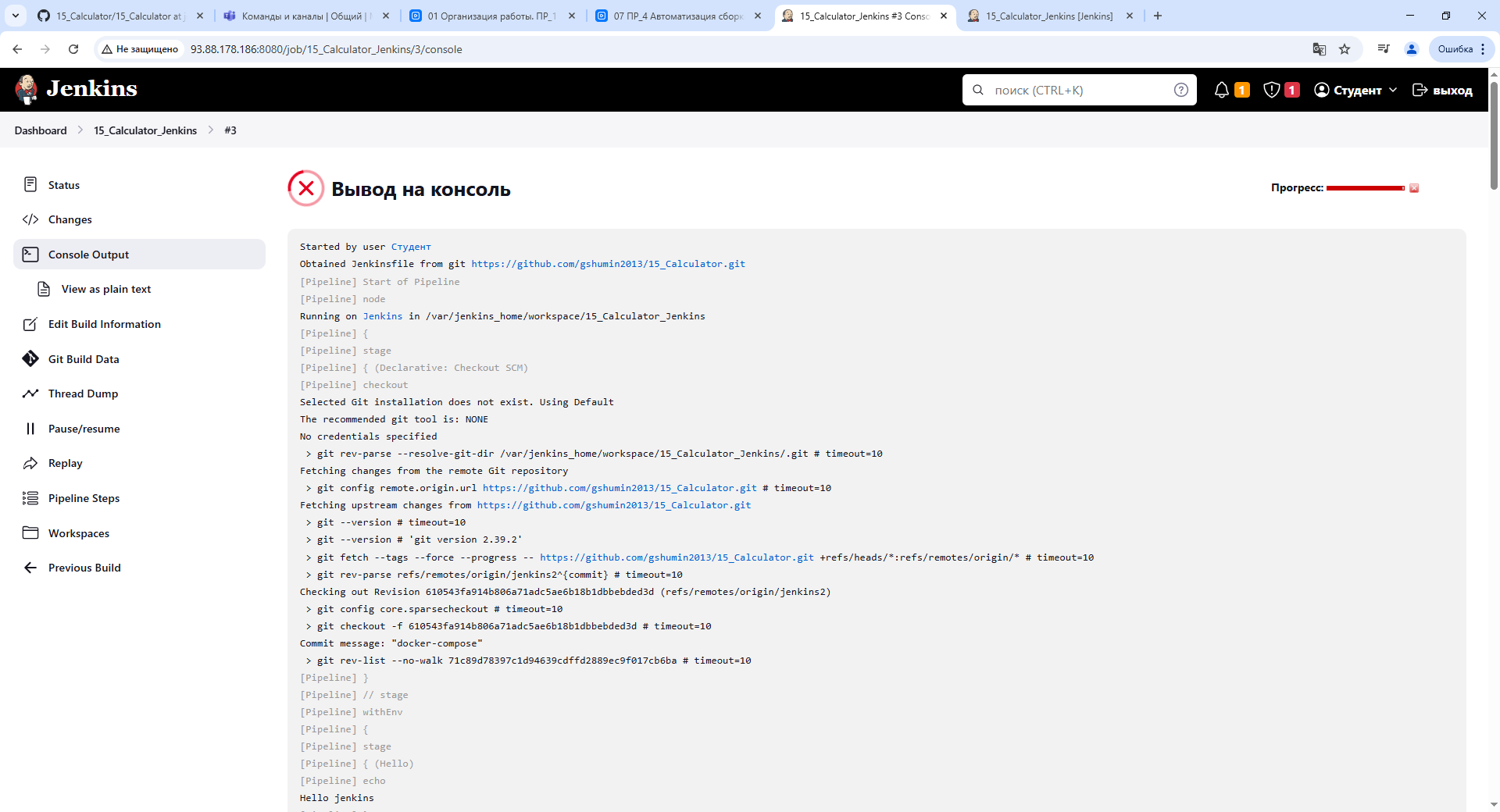


Рисунок 10-11 – Вывод на консоль попытки сборки.

1.3 Выводы

В данной практической работе было изучена система непрерывной интеграции программного обеспечения Jenkins. Данная система позволяет автоматизировать процесс выгрузки приложения. В моём случае она создаем контейнер и выгружает в него последнюю версию приложения “15\_Calculator”.

Приложение А Листинг программного кода приложения «Калькулятор»   
(контроллер)

using \_15\_Calculator.Models;

using Microsoft.AspNetCore.Components.Forms;

using Microsoft.AspNetCore.Mvc;

namespace Calculator.Controllers

{

public enum Operation { Add, Subtract, Multiply, Divide }

public class CalculatorController : Controller

{

[HttpGet]

public IActionResult Index()

{

return View();

}

[HttpPost]

[ValidateAntiForgeryToken]

public IActionResult Calculate(double num1, double num2, Operation operation)

{

double result = 0;

switch (operation)

{

case Operation.Add:

result = num1 + num2;

break;

case Operation.Subtract:

result = num1 - num2;

break;

case Operation.Multiply:

result = num1 \* num2;

break;

case Operation.Divide:

result = num1 / num2;

break;

}

ViewBag.Result = result;

return View("Index");

}

}

}

Приложение Б Листинг программного кода приложения «Калькулятор» (appsettings.json)

﻿{

"Kestrel": {

"Endpoints": {

"Http": {

"Url": "http://0.0.0.0:5515"

}

}

},

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft": "Warning",

"Microsoft.Hosting.Lifetime": "Information"

}

},

"AllowedHosts": "\*"

}

Приложение В Листинг программного кода приложения «Калькулятор» (appsettings.Development.json)

{

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft.AspNetCore": "Warning"

}

}

}

Приложение Г Листинг программного кода приложения «Калькулятор» (Program.cs)

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllersWithViews();

var app = builder.Build();

// Configure the HTTP request pipeline.

if (!app.Environment.IsDevelopment())

{

app.UseExceptionHandler("/Home/Error");

// The default HSTS value is 30 days. You may want to change this for production scenarios, see https://aka.ms/aspnetcore-hsts.

app.UseHsts();

}

app.UseHttpsRedirection();

app.UseStaticFiles();

app.UseRouting();

app.UseAuthorization();

app.MapControllerRoute(

name: "default",

pattern: "{controller=Calculator}/{action=Index}/{id?}");

app.Run();

Приложение Д Листинг файла Dockerfile

# Используем базовый образ с ASP.NET 6.0

FROM mcr.microsoft.com/dotnet/aspnet:6.0 AS base

# Устанавливаем рабочую директорию внутри контейнера

WORKDIR /app

# Используем SDK

FROM mcr.microsoft.com/dotnet/sdk:6.0 AS build

# Копируем файлы в контейнер

COPY . /src

# Устанавливаем рабочую директорию внутри контейнера

WORKDIR /src

# Устанавливаем зависимости приложения и параметры компиляции

RUN ls

RUN dotnet restore

RUN dotnet build "./15\_Calculator.csproj" -c Release -o /app/build

FROM build AS publish

RUN dotnet publish "./15\_Calculator.csproj" -c Release -o /app/publish

FROM base AS final

WORKDIR /app

COPY --from=publish /app/publish ./

# Определяем команду запуска контейнера

ENTRYPOINT ["dotnet", "15\_Calculator.dll"]

Приложение Е Листинг файла docker-compose.yml

version: '3.7'

services:

web:

container\_name: web-app-15\_calculator\_jenkins

hostname: app

build: 15\_Calculator/

ports:

- "0.0.0.0:5715:5515"

Приложение Ж Листинг файла Jenkinsfile

pipeline {

agent any

stages {

stage('Hello') {

steps {

echo 'Hello jenkins'

}

}

stage('Build Docker Image') {

steps {

sh 'ls -l'

sh 'docker compose build'

}

}

stage('Start Docker Container') {

steps {

sh 'docker compose up -d'

}

}

}

}