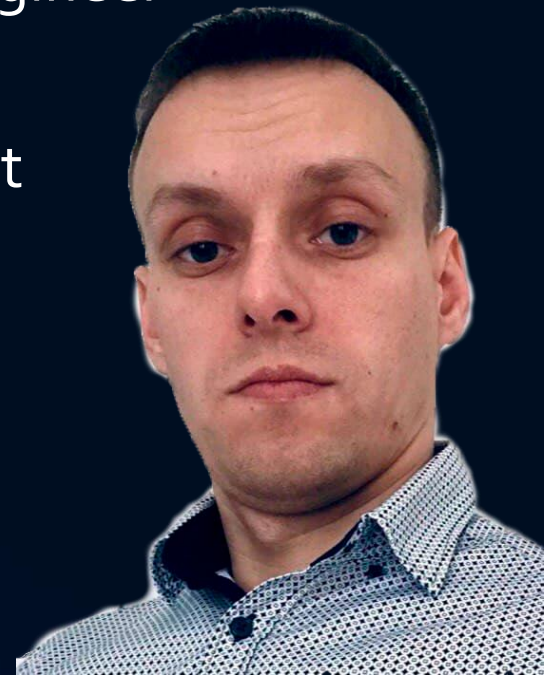


# Final project: CI/CD pipeline Spring Petclinic

VALERY YURCHENKO,  
EPAM DEVOPS EXTERNAL COURSE

# Let me to tell some facts about myself:

- I am 36
- I am from Ops: System Administrator / Support Engineer (Windows, GNU/Linux) 15+ years / Head of IT Department at Matrapac Ukraine.
- My goal is to upgrade my skills and start to work as System Engineer with EPAM.
- I like System Engineering and System thinking and believe that system practices make our world better place.



## What is my motivation?

- *This is my first CI/CD project, who teaches me, how it works.*
- In process of creating of this project I get many knowledge (subjectively) in tools, what I used.

## Relevance:

- My method is relevant, because it allows in a short time and with the least effort on the part of a software developer to organize the assembly of the artifact from the source code and the delivery of the artifact to the testing and staging environment.

## Goals:

- Automation
- Reducing the entry threshold
- Savings
- Security

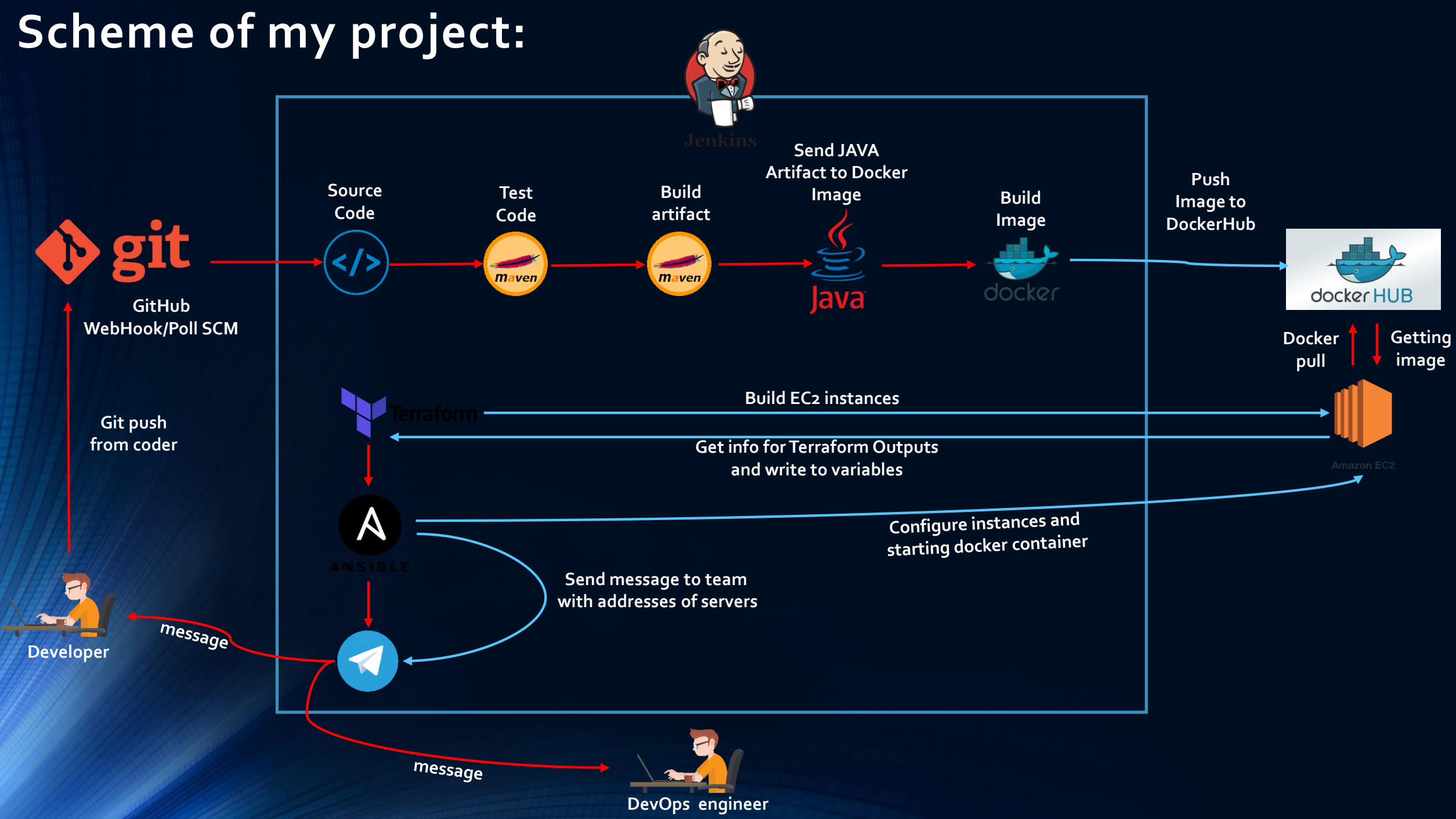
# Selected technology stack:

- GNU/Linux
- GitHub
- Jenkins
- Ansible
- Docker
- Terraform
- AWS
- Telegram



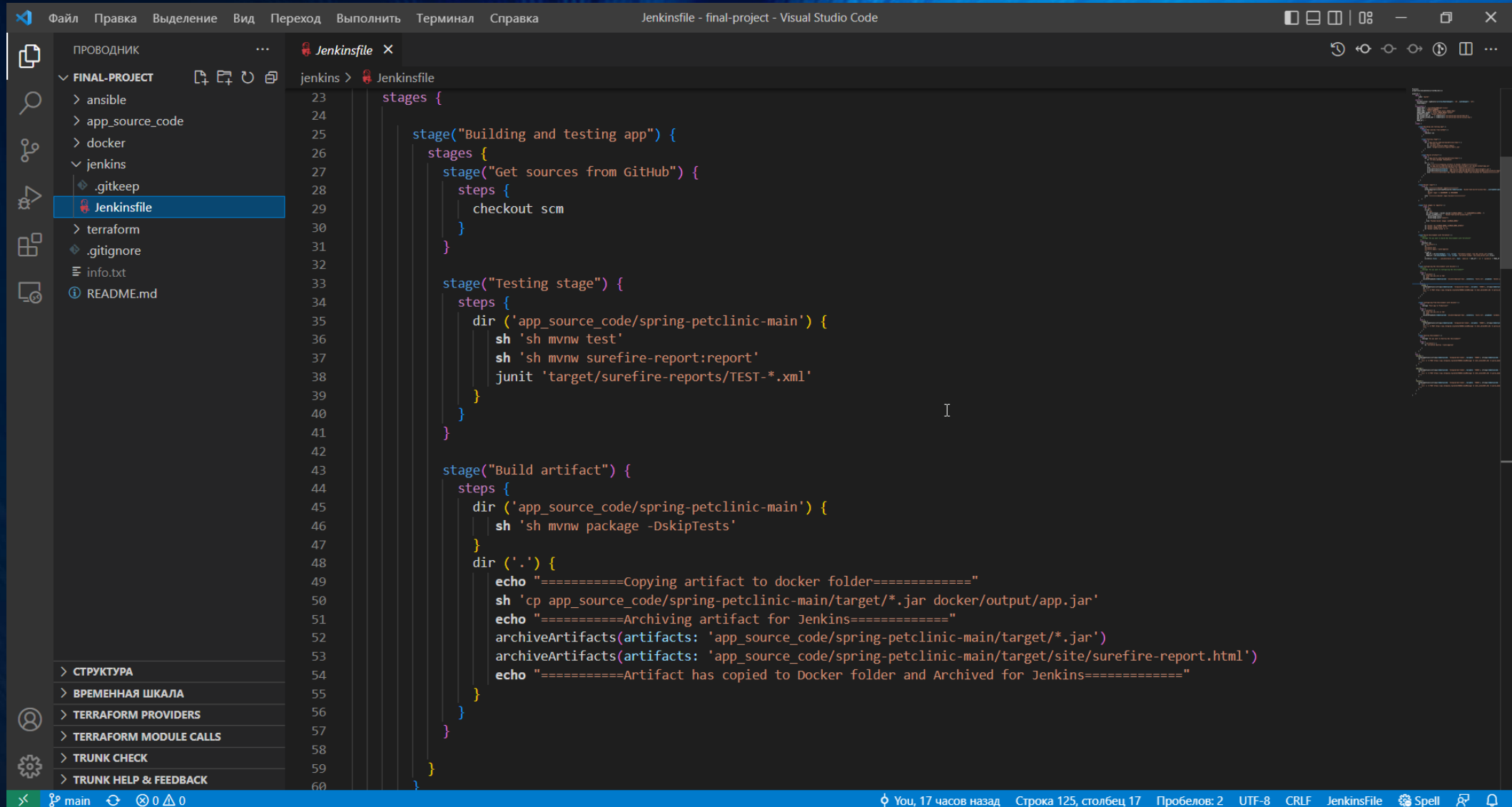
# Scheme of my project:

```
graph LR
    Developer[Developer] -- "Git push from coder" --> GitHub[GitHub]
    GitHub -- "GitHub WebHook/Poll SCM" --> Jenkins[Jenkins]
    Jenkins -- "Source Code" --> Test[Test Code]
    Test -- "Test Code" --> Build[Build artifact]
    Build -- "Build artifact" --> DockerImage[Send JAVA Artifact to Docker Image]
    DockerImage -- "Send JAVA Artifact to Docker Image" --> BuildImage[Build Image]
    BuildImage -- "Build Image" --> DockerHub[docker HUB]
    DockerHub -- "Push Image to DockerHub" --> DockerHub
    DockerHub -- "Docker pull" --> AmazonEC2[Amazon EC2]
    AmazonEC2 -- "Getting image" --> DockerHub
    AmazonEC2 -- "Build EC2 instances" --> Terraform[Terraform]
    Terraform -- "Get info for Terraform Outputs and write to variables" --> Ansible[ANSIBLE]
    Ansible -- "Configure instances and starting docker container" --> AmazonEC2
    Ansible -- "Send message to team with addresses of servers" --> Telegram[Telegram]
    Telegram -- "message" --> Developer
    Telegram -- "message" --> DevOps[DevOps engineer]
```



# Realization:

## Testing and Building:

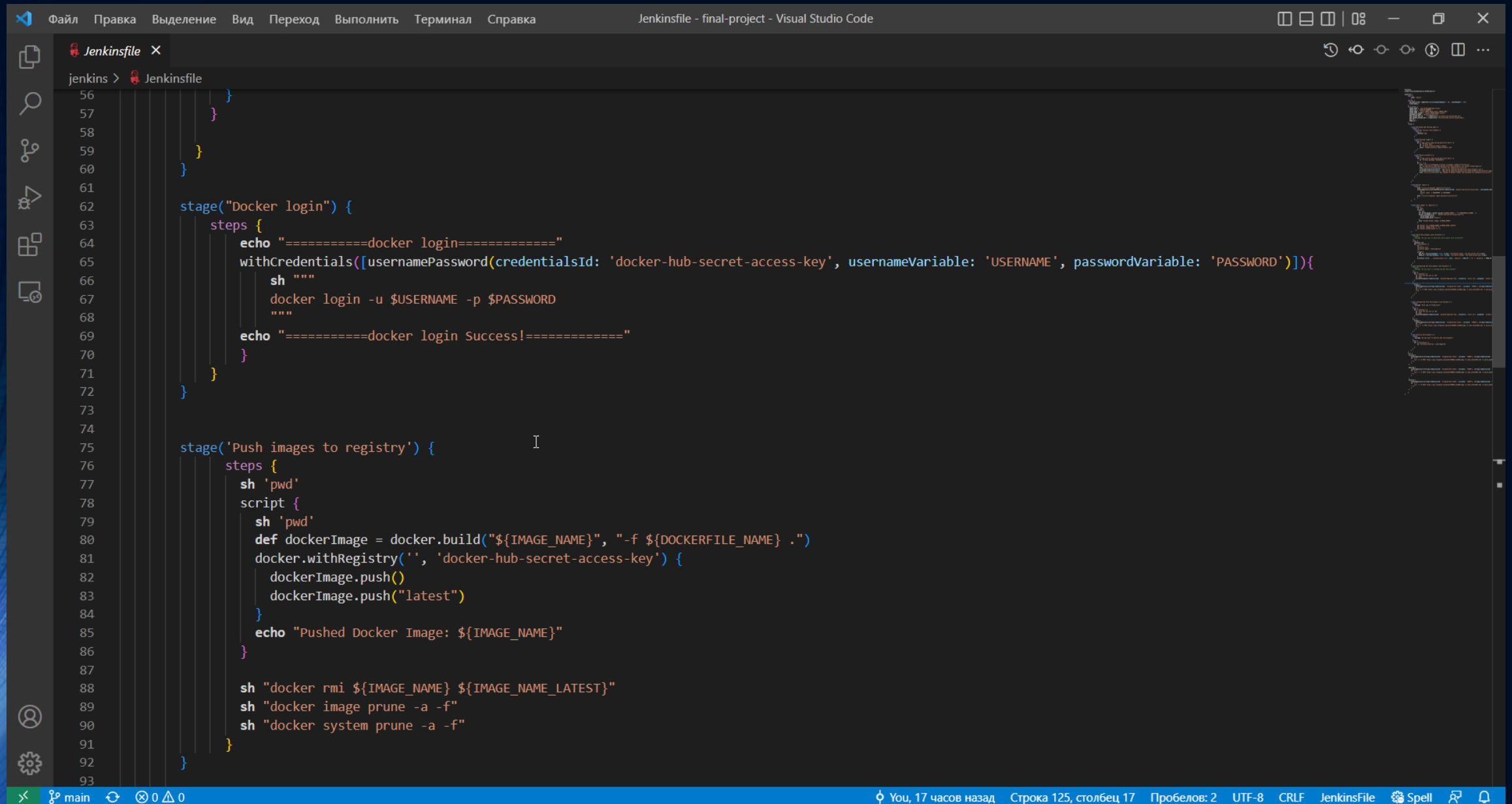


The screenshot shows the Visual Studio Code editor with a Jenkinsfile open. The file explorer on the left shows the project structure, including a 'jenkins' directory containing the 'Jenkinsfile'. The main editor displays the Jenkinsfile content, which is a YAML script defining the CI/CD pipeline. The pipeline consists of three stages: 'Building and testing app', 'Testing stage', and 'Build artifact'. The 'Building and testing app' stage includes a 'Get sources from GitHub' step. The 'Testing stage' includes steps to run tests and generate reports. The 'Build artifact' stage includes steps to package the application, copy artifacts to a Docker folder, and archive them for Jenkins.

```
jenkins > Jenkinsfile
23 stages {
24
25     stage("Building and testing app") {
26         stages {
27             stage("Get sources from GitHub") {
28                 steps {
29                     checkout scm
30                 }
31             }
32         }
33
34         stage("Testing stage") {
35             steps {
36                 dir ('app_source_code/spring-petclinic-main') {
37                     sh 'sh mvnw test'
38                     sh 'sh mvnw surefire-report:report'
39                     junit 'target/surefire-reports/TEST-*.xml'
40                 }
41             }
42         }
43
44         stage("Build artifact") {
45             steps {
46                 dir ('app_source_code/spring-petclinic-main') {
47                     sh 'sh mvnw package -DskipTests'
48                 }
49                 dir ('.') {
50                     echo "====Copying artifact to docker folder===="
51                     sh 'cp app_source_code/spring-petclinic-main/target/*.jar docker/output/app.jar'
52                     echo "====Archiving artifact for Jenkins===="
53                     archiveArtifacts(artifacts: 'app_source_code/spring-petclinic-main/target/*.jar')
54                     archiveArtifacts(artifacts: 'app_source_code/spring-petclinic-main/target/site/surefire-report.html')
55                     echo "====Artifact has copied to Docker folder and Archived for Jenkins===="
56                 }
57             }
58         }
59     }
60 }
```

main 0 0 0 You, 17 часов назад Строка 125, столбец 17 Пробелов: 2 UTF-8 CRLF JenkinsFile Spell

# Creating of the Docker Image and pushing to DockerHub:

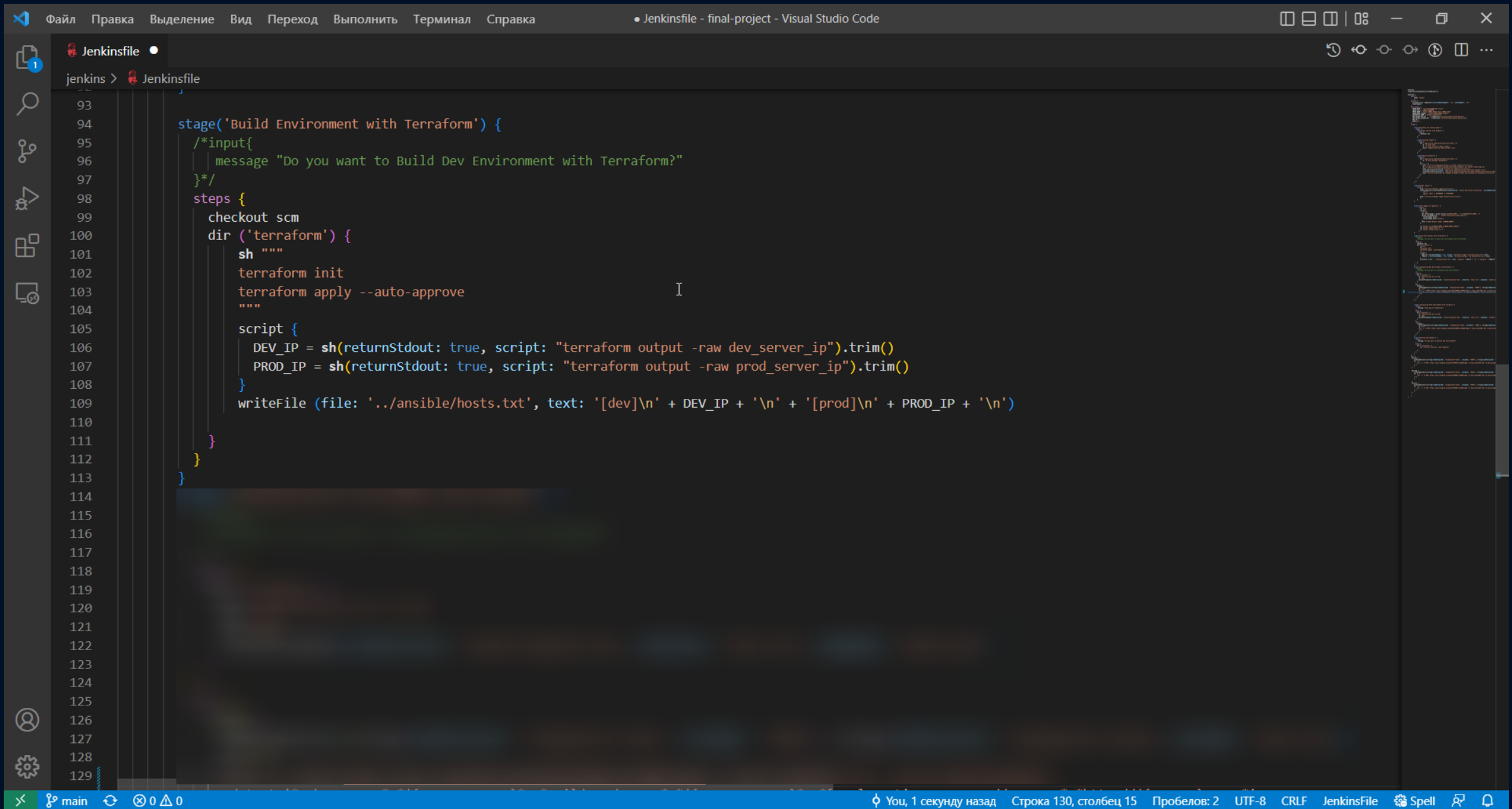


The screenshot shows the Visual Studio Code editor with a file named 'Jenkinsfile'. The editor is displaying a Jenkinsfile script with two main stages: 'Docker login' and 'Push images to registry'. The 'Docker login' stage uses 'withCredentials' to log into Docker Hub. The 'Push images to registry' stage builds a Docker image and pushes it to the registry, followed by cleanup commands for Docker images and system.

```
jenkins > Jenkinsfile
56
57
58
59
60
61
62 stage("Docker login") {
63     steps {
64         echo "=====docker login=====
65         withCredentials([usernamePassword(credentialsId: 'docker-hub-secret-access-key', usernameVariable: 'USERNAME', passwordVariable: 'PASSWORD')]){
66             sh """
67             docker login -u $USERNAME -p $PASSWORD
68             """
69         echo "=====docker login Success!=====
70     }
71 }
72
73
74
75 stage('Push images to registry') {
76     steps {
77         sh 'pwd'
78         script {
79             sh 'pwd'
80             def dockerImage = docker.build("${IMAGE_NAME}", "-f ${DOCKERFILE_NAME} .")
81             docker.withRegistry('', 'docker-hub-secret-access-key') {
82                 dockerImage.push()
83                 dockerImage.push("latest")
84             }
85             echo "Pushed Docker Image: ${IMAGE_NAME}"
86         }
87
88         sh "docker rmi ${IMAGE_NAME} ${IMAGE_NAME_LATEST}"
89         sh "docker image prune -a -f"
90         sh "docker system prune -a -f"
91     }
92 }
93
```

Visual Studio Code interface details: The top bar shows the file 'Jenkinsfile - final-project - Visual Studio Code'. The left sidebar contains icons for Explorer, Search, Source Control, Run and Debug, and Extensions. The bottom status bar shows 'main', '0 errors, 0 warnings', 'You, 17 часов назад', 'Строка 125, столбец 17', 'Пробелов: 2', 'UTF-8', 'CRLF', 'JenkinsFile', 'Spell', and a search icon.

# Building instances with Terraform:



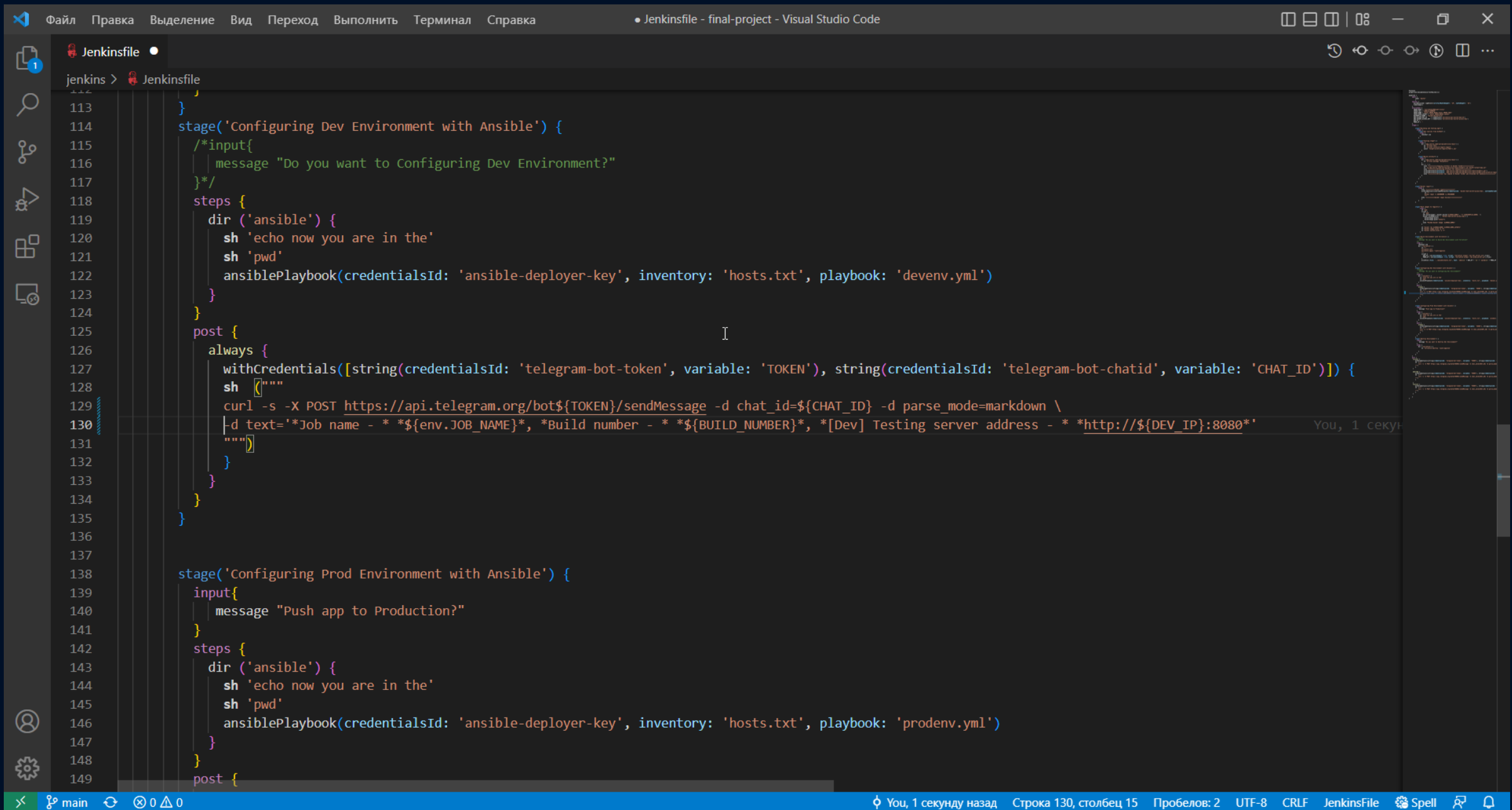
The screenshot shows the Visual Studio Code editor interface. The top menu bar includes 'Файл', 'Правка', 'Выделение', 'Вид', 'Переход', 'Выполнить', 'Терминал', and 'Справка'. The title bar indicates the current file is 'Jenkinsfile - final-project - Visual Studio Code'. The left sidebar shows the Explorer view with a file named 'Jenkinsfile'. The main editor area displays the following Jenkinsfile content:

```
jenkins > Jenkinsfile
93
94 stage('Build Environment with Terraform') {
95     /*input{
96         message "Do you want to Build Dev Environment with Terraform?"
97     }*/
98     steps {
99         checkout scm
100         dir ('terraform') {
101             sh """
102                 terraform init
103                 terraform apply --auto-approve
104             """
105             script {
106                 DEV_IP = sh(returnStdout: true, script: "terraform output -raw dev_server_ip").trim()
107                 PROD_IP = sh(returnStdout: true, script: "terraform output -raw prod_server_ip").trim()
108             }
109             writeFile (file: '../ansible/hosts.txt', text: '[dev]\n' + DEV_IP + '\n' + '[prod]\n' + PROD_IP + '\n')
110         }
111     }
112 }
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
```

The bottom status bar shows the current file is 'main', there are 0 errors and 0 warnings, and the cursor is at line 130, column 15. The status bar also displays 'UTF-8', 'CRLF', 'JenkinsFile', and 'Spell'.



# Configuring instances with Ansible:



The image shows a Visual Studio Code editor window with a file named 'Jenkinsfile'. The editor is displaying a Jenkinsfile script with two main stages: 'Configuring Dev Environment with Ansible' and 'Configuring Prod Environment with Ansible'. The first stage includes an input prompt, a directory change to 'ansible', running shell commands to echo the current directory and run 'pwd', and an Ansible playbook execution. The second stage includes an input prompt, a directory change to 'ansible', running shell commands to echo the current directory and run 'pwd', and an Ansible playbook execution. The status bar at the bottom indicates the current position is line 130, column 15, and the file encoding is UTF-8.

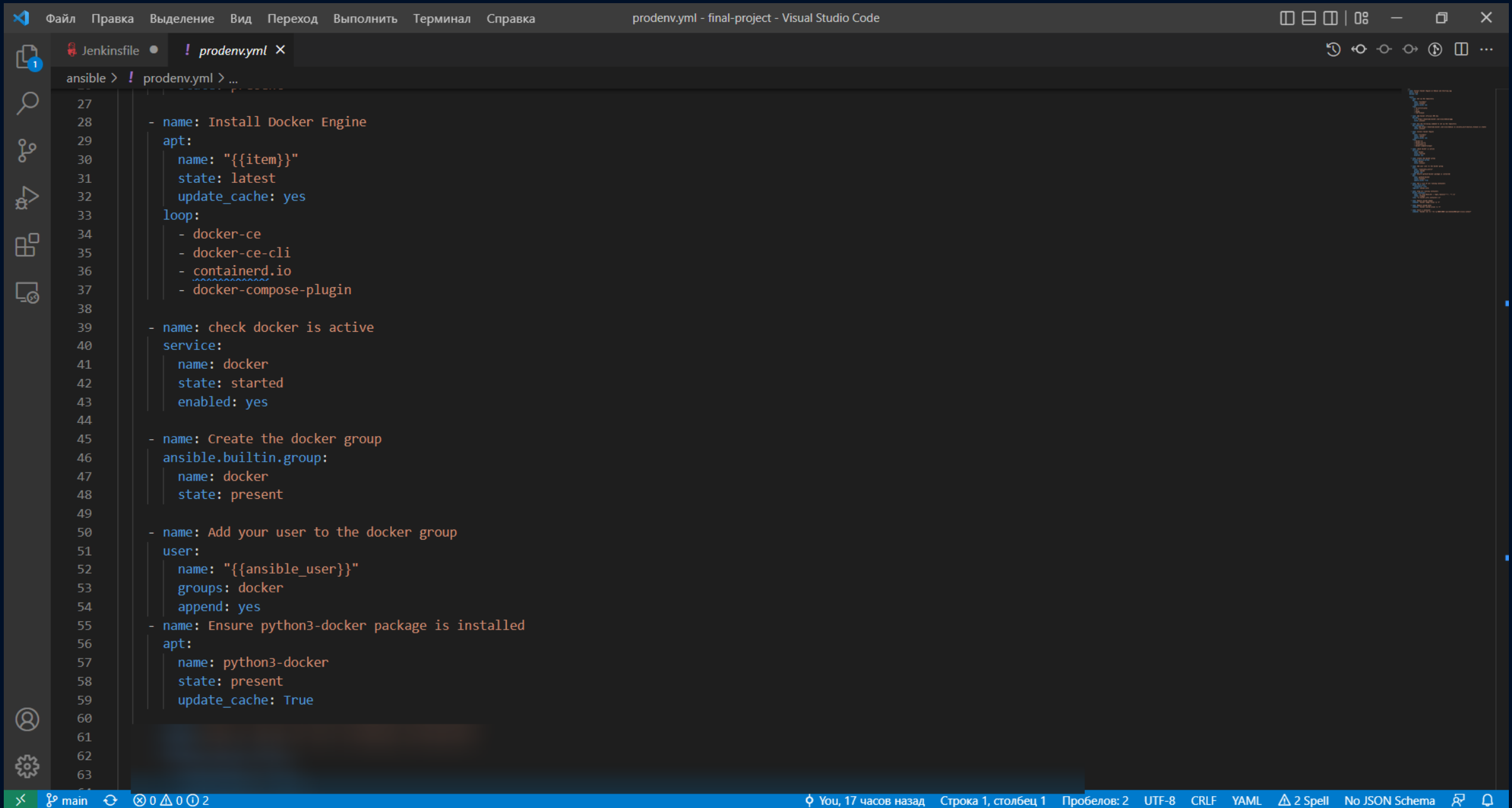
```
jenkins > Jenkinsfile
113 }
114 stage('Configuring Dev Environment with Ansible') {
115     /*input{
116         message "Do you want to Configuring Dev Environment?"
117     }*/
118     steps {
119         dir ('ansible') {
120             sh 'echo now you are in the'
121             sh 'pwd'
122             ansiblePlaybook(credentialsId: 'ansible-deployer-key', inventory: 'hosts.txt', playbook: 'devenv.yml')
123         }
124     }
125     post {
126         always {
127             withCredentials([string(credentialsId: 'telegram-bot-token', variable: 'TOKEN'), string(credentialsId: 'telegram-bot-chatid', variable: 'CHAT_ID')]) {
128                 sh ["""
129                 curl -s -X POST https://api.telegram.org/bot${TOKEN}/sendMessage -d chat_id=${CHAT_ID} -d parse_mode=markdown \
130                 -d text='*Job name - * ${env.JOB_NAME}*, *Build number - * ${BUILD_NUMBER}*, *[Dev] Testing server address - * http://${DEV_IP}:8080*'
131                 """]
132             }
133         }
134     }
135 }
136
137
138 stage('Configuring Prod Environment with Ansible') {
139     input{
140         message "Push app to Production?"
141     }
142     steps {
143         dir ('ansible') {
144             sh 'echo now you are in the'
145             sh 'pwd'
146             ansiblePlaybook(credentialsId: 'ansible-deployer-key', inventory: 'hosts.txt', playbook: 'prodenv.yml')
147         }
148     }
149     post {
```

# Ansible Playbook.yml:

```
ansible > ! prodenv.yml > ...
You, 17 часов назад | 1 author (You)
1  |-- You, 17 часов назад • first commit ...
2  - name: Install Docker Engine on Debian and Starting app
3    hosts: prod
4    become: yes
5
6    tasks:
7      - name: Set up the repository
8        apt:
9          name: "{{item}}"
10         state: present
11         update_cache: yes
12       loop:
13         - ca-certificates
14         - curl
15         - gnupg
16         - lsb-release
17
18      - name: Add Docker official GPG key
19        apt_key:
20          url: https://download.docker.com/linux/debian/gpg
21          state: present
22
23      - name: Use the following command to set up the repository
24        apt_repository:
25          repo: deb https://download.docker.com/linux/debian {{ ansible_distribution_release }} stable
26          state: present
27
28
29
30
31
32
33
34
35
36
37
```

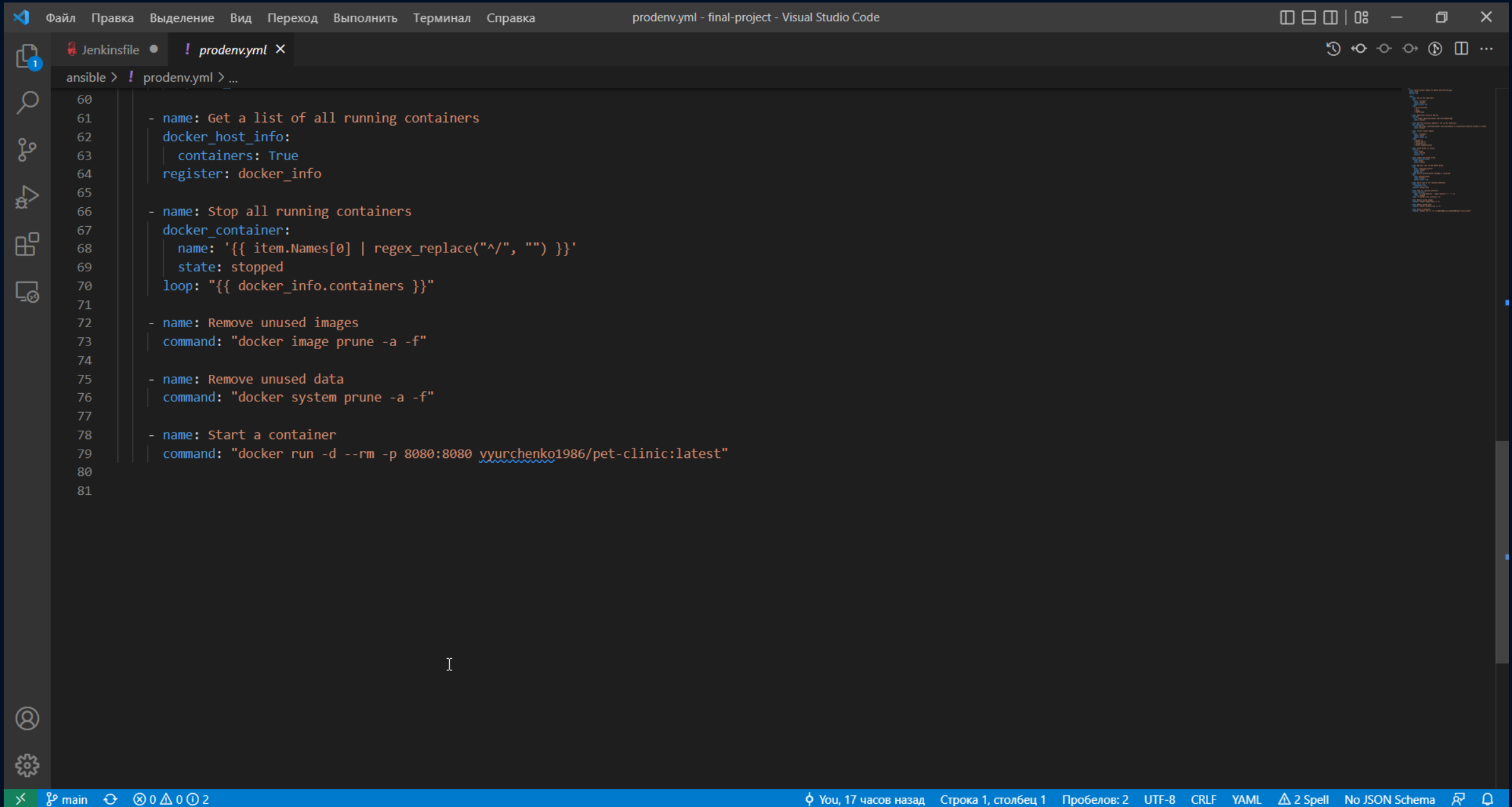
main 0 0 0 2 You, 17 часов назад Строка 1, столбец 1 Пробелов: 2 UTF-8 CRLF YAML 2 Spell No JSON Schema

# Ansible Playbook.yml:



```
ansible > ! prodenv.yml > ...
27
28 - name: Install Docker Engine
29   apt:
30     name: "{{item}}"
31     state: latest
32     update_cache: yes
33   loop:
34     - docker-ce
35     - docker-ce-cli
36     - containerd.io
37     - docker-compose-plugin
38
39 - name: check docker is active
40   service:
41     name: docker
42     state: started
43     enabled: yes
44
45 - name: Create the docker group
46   ansible.builtin.group:
47     name: docker
48     state: present
49
50 - name: Add your user to the docker group
51   user:
52     name: "{{ansible_user}}"
53     groups: docker
54     append: yes
55
56 - name: Ensure python3-docker package is installed
57   apt:
58     name: python3-docker
59     state: present
60     update_cache: True
61
62
63
```

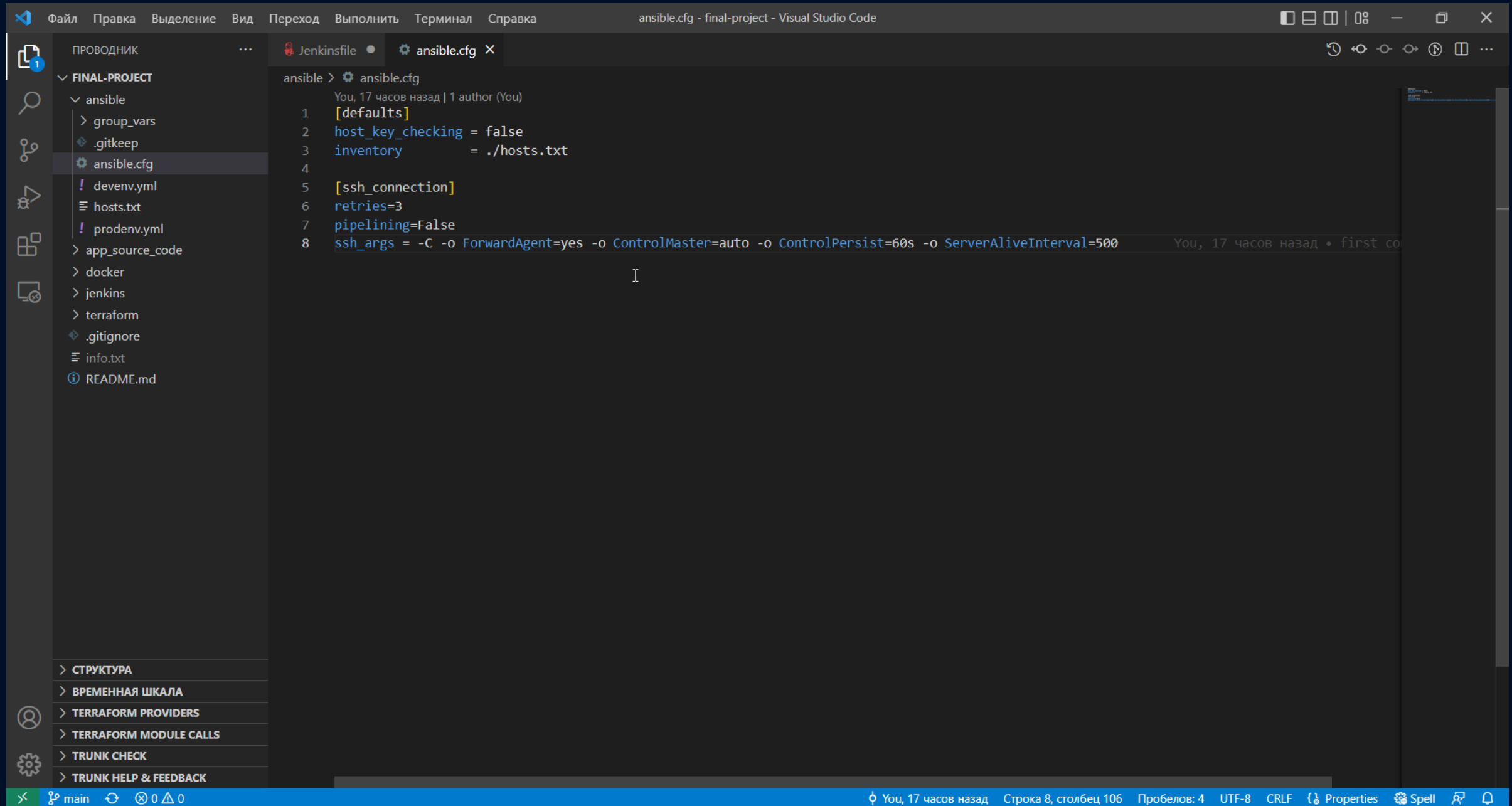
# Ansible Playbook.yml:



```
ansible > ! prodenv.yml > ...
60
61 - name: Get a list of all running containers
62   docker_host_info:
63     containers: True
64   register: docker_info
65
66 - name: Stop all running containers
67   docker_container:
68     name: '{{ item.Names[0] | regex_replace("^/", "") }}'
69     state: stopped
70   loop: "{{ docker_info.containers }}"
71
72 - name: Remove unused images
73   command: "docker image prune -a -f"
74
75 - name: Remove unused data
76   command: "docker system prune -a -f"
77
78 - name: Start a container
79   command: "docker run -d --rm -p 8080:8080 vyurchenko1986/pet-clinic:latest"
80
81
```

main 0 0 0 2 You, 17 часов назад Строка 1, столбец 1 Пробелов: 2 UTF-8 CRLF YAML 2 Spell No JSON Schema

# Ansible Ansible.cfg:



The screenshot shows the Visual Studio Code interface with the 'ansible.cfg' file open. The file is located in the 'final-project' directory, under the 'ansible' subdirectory. The file content is as follows:

```
ansible > ansible.cfg
You, 17 часов назад | 1 author (You)
1 [defaults]
2 host_key_checking = false
3 inventory = ./hosts.txt
4
5 [ssh_connection]
6 retries=3
7 pipelining=False
8 ssh_args = -C -o ForwardAgent=yes -o ControlMaster=auto -o ControlPersist=60s -o ServerAliveInterval=500
```

The status bar at the bottom indicates the current line and column: 'Строка 8, столбец 106'. The file is encoded in UTF-8 and uses CRLF line endings.



# Live demonstration

**Thank you for your  
attention!**