

Федеральное государственное бюджетное образовательное
учреждение высшего образования «Сибирский государственный
университет телекоммуникаций и информатики» (СибГУТИ)

Кафедра Вычислительных систем

РАСЧЕТНО-ГРАФИЧЕСКОЕ ЗАДАНИЕ
по дисциплине «Архитектуры вычислительных систем»

Выполнил: студент группы ИС-242
Журбенко Владислав Евгеньевич
Проверил: доцент кафедры ВС
Романюта Алексей Андреевич

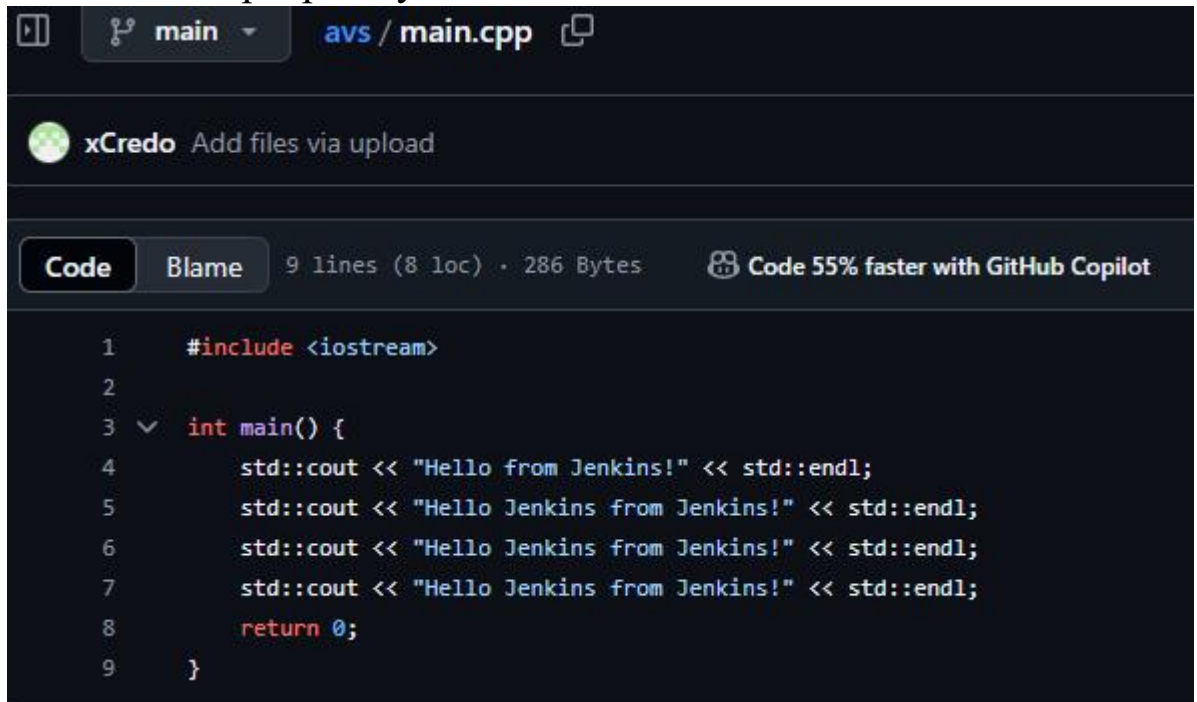
Новосибирск 2024 г.

Задание РГР:

- Запуск jenkins и разработка сценариев CD
- Интерфейс доступен по https (jenkins)
- В jenkins создан пайплайн/задача/таск, работающий с репозиторием gitlab/github
- При изменении в репозитории автоматически запускается задача обновления docker-контейнера с содержимым репозитория (Либо сборка образа с новым приложением и обновление контейнера)
- Оформить как docker-compose.yml
- При пересоздании проекта все настройки jenkins должны сохраниться
- Реализовать без использования ngrok и webhook

Ход работы:

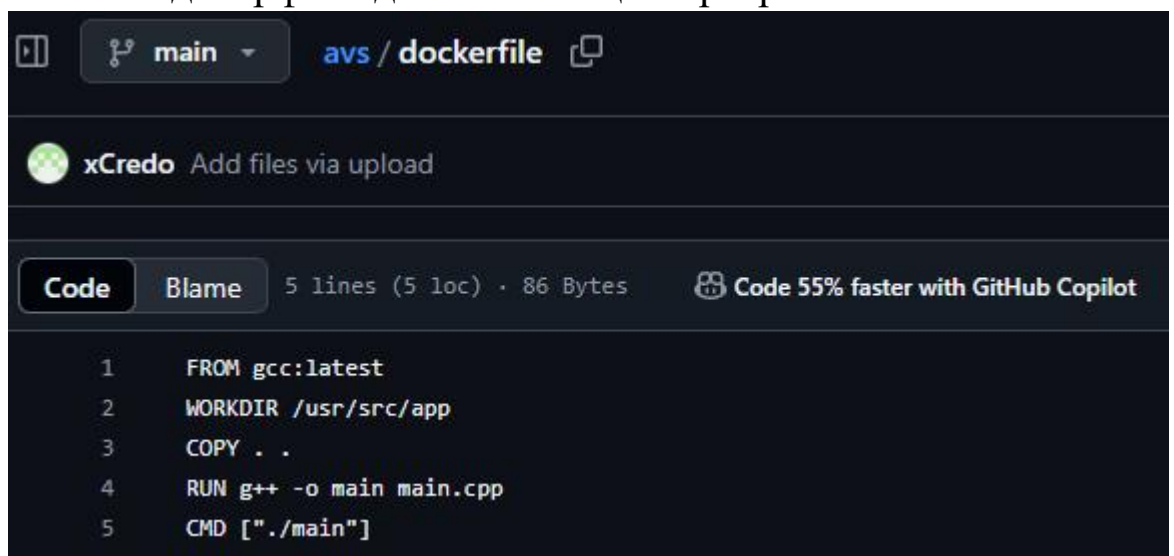
1. Создал репозиторий.
2. Написал программу для теста



The screenshot shows a GitHub repository interface. At the top, there's a navigation bar with a file icon, a dropdown menu showing 'main', and the file path 'avs / main.cpp'. Below this is a section for 'xCredo' with the text 'Add files via upload'. The main content area has tabs for 'Code' and 'Blame'. The 'Code' tab is selected, showing the content of 'main.cpp'. The code is a C++ program that includes <iostream> and has a main function that prints four lines of text: 'Hello from Jenkins!', 'Hello Jenkins from Jenkins!', 'Hello Jenkins from Jenkins!', and 'Hello Jenkins from Jenkins!'. The code is 9 lines long, 8 loc, and 286 Bytes. A badge indicates 'Code 55% faster with GitHub Copilot'.

```
1  #include <iostream>
2
3  int main() {
4      std::cout << "Hello from Jenkins!" << std::endl;
5      std::cout << "Hello Jenkins from Jenkins!" << std::endl;
6      std::cout << "Hello Jenkins from Jenkins!" << std::endl;
7      std::cout << "Hello Jenkins from Jenkins!" << std::endl;
8      return 0;
9  }
```

3. Написал докерфайл для компиляции программы



The screenshot shows a GitHub repository interface. At the top, there's a navigation bar with a file icon, a dropdown menu showing 'main', and the file path 'avs / dockerfile'. Below this is a section for 'xCredo' with the text 'Add files via upload'. The main content area has tabs for 'Code' and 'Blame'. The 'Code' tab is selected, showing the content of 'dockerfile'. The Dockerfile contains five lines of instructions: 'FROM gcc:latest', 'WORKDIR /usr/src/app', 'COPY . .', 'RUN g++ -o main main.cpp', and 'CMD ["/main"]'. The Dockerfile is 5 lines long, 5 loc, and 86 Bytes. A badge indicates 'Code 55% faster with GitHub Copilot'.

```
1  FROM gcc:latest
2  WORKDIR /usr/src/app
3  COPY . .
4  RUN g++ -o main main.cpp
5  CMD ["/main"]
```

4. Создал Jenkinsfile для запуска приложения в контейнере.

```

1 pipeline {
2     agent any //Пайплайн выполняется на локальных и удалённых машинах
3
4     environment {
5         IMAGE_NAME = 'avs' //Имя докер-контейнера, -образа.
6         DOCKER_IMAGE = 'avs:latest'
7         PORT = '8085' // Порт для использования контейнером
8     }
9
10    stages {
11        stage('Checkout') {
12            steps {
13                git branch: 'master', url: 'https://github.com/xCredo/avs.git' // Клонирование ветки с репозитория
14            }
15        }
16
17        stage('Build Docker Image') {
18            steps {
19                script {
20                    sh "docker build -t ${DOCKER_IMAGE} ." //Собираем докер-образ
21                }
22            }
23        }
24
25        stage('Deploy Docker Container') {
26            steps {
27                script {
28                    // Проверяем запущен ли контейнер, описанный в переименных окружения
29                    // и если запущен,тогда останавливаем и удаляем
30                    sh """
31                        if [ \$(docker ps -aq -f name=${IMAGE_NAME}) ]; then
32                            docker stop ${IMAGE_NAME}
33                            docker rm ${IMAGE_NAME}
34                        fi
35                    """
36                    //// Проверяем занят ли порт и еслион занят,
37                    //// тогда заершаем процессф на порту
38                    sh """
39                        if ss -tuln | grep :${PORT}; then
40                            echo "Port ${PORT} is in use. Freeing it..."
41                            fuser -k ${PORT}/tcp || true
42                        fi
43                    """
44                    // Запускаем новый контейнер, пробрасывает порт 8181 на порт 80 внутри контейнера
45                    // и указываем имя образа
46                    sh "docker run -d --name ${IMAGE_NAME} -p 8181:80 ${DOCKER_IMAGE}"
47                }
48            }
49        }
50    }
51
52    post { // Описание выводимых сообщений в случае успешного/неуспешного выполнения пайплайна.
53        success {
54            echo 'Deployment completed successfully!'
55        }
56        failure {
57            echo 'Deployment failed. Check logs for details.'
58        }
59    }
60 }

```

5. Для работы по https сгенерировал сертификат для домена example.localhost через утилиту mkcert:

<https://github.com/FiloSottile/mkcert>

6. Для работы через https необходимо прописать dns запись в hosts файле.

```
# This file was automatically generated by WSL. To stop automatic generation of
this file, add the following entry to /etc/wsl.conf:
# [network]
# generateHosts = false
127.0.0.1    localhost
127.0.1.1    Gamarjoba-pc.  Gamarjoba-pc

# The following lines are desirable for IPv6 capable hosts
::1        ip6-localhost ip6-loopback
fe00::0    ip6-localnet
ff00::0    ip6-mcastprefix
ff02::1    ip6-allnodes
ff02::2    ip6-allrouters
```

7. Docker-compose для сборки проекта:

```
services:
  traefik:
    image: traefik
    command:
      - "--api.insecure=true"
      - "--providers.docker=true"
      - "--providers.file.filename=/traefik_dynamic_conf.yml"
      - "--entrypoints.web.address=:80"
      - "--entrypoints.websecure.address=:443"
    ports:
      - "80:80"      # HTTP трафик
      - "443:443"    # HTTPS трафик
    volumes:
      - "/var/run/docker.sock:/var/run/docker.sock:ro"
      - "./traefik_dynamic_conf.yml:/traefik_dynamic_conf.yml"
      - "../certs/example.localhost.pem:/certs/example.localhost.pem"
      - "../certs/example.localhost-key.pem:/certs/example.localhost-key.pem"
    networks:
      - web

  jenkins:
    build: ./jenkins
    user: root
    environment:
      JAVA_OPTS: -Djenkins.install.runSetupWizard=false
      JENKINS_OPTS: --prefix=/jenkins --argumentsRealm.roles.user=admin --argumentsRealm.passwd.admin=admin --argumentsRealm.roles.admin=admin
    volumes:
      - jenkins_home:/var/jenkins_home
      - /var/run/docker.sock:/var/run/docker.sock
    labels:
      - "traefik.enable=true"
      - "traefik.http.routers.jenkins.tls=true"
      - "traefik.http.routers.jenkins.entrypoints=websecure"
      - "traefik.http.routers.jenkins.rule=Host(`example.localhost`) && PathPrefix(`/jenkins`)"
      - "traefik.http.services.jenkins.loadbalancer.server.port=8080"
    networks:
      - web

volumes:
  jenkins_home:

networks:
  web:
    driver: bridge
```

8. Traefic_dynamic_conf.yml

```
3  tls:
2    certificates:
1      - certFile: ../certs/example.localhost.pem
4        keyFile: ../certs/example.localhost-key.pem
```

9. Структура проекта:

```
avs [?] master][?]
> tree
.
├── certs
│   ├── example.localhost-key.pem
│   └── example.localhost.pem
├── dockerfile
├── infra
│   ├── docker-compose.yml
│   ├── jenkins
│   │   └── dockerfile
│   └── ! traefik_dynamic_conf.yml
├── Jenkinsfile
└── main.cpp
```


10. Запуск сборки

```
avs/infra [⌚ master][?]  
> docker compose up -d --build  
[+] Building 0.0s (0/1)                                docker:default  
[+] Running 0/1  
[+] Building 0.8s (10/10) FINISHED                    docker:default  
⇒ [jenkins internal] load build definition from dockerfile 0.0s  
⇒ ⇒ transferring dockerfile: 544B 0.0s  
⇒ [jenkins internal] load metadata for docker.io/jenkins/jenkins:lates 0.6s  
⇒ [jenkins internal] load .dockerignore 0.0s  
⇒ ⇒ transferring context: 2B 0.0s  
⇒ [jenkins 1/5] FROM docker.io/jenkins/jenkins:latest@sha256:db5f83e66 0.0s  
⇒ CACHED [jenkins 2/5] RUN apt-get update && apt-get install -y lsb-re 0.0s  
⇒ CACHED [jenkins 3/5] RUN curl -fsSLo /usr/share/keyrings/docker-arch 0.0s  
⇒ CACHED [jenkins 4/5] RUN echo "deb [arch=$(dpkg --print-architecture 0.0s  
⇒ CACHED [jenkins 5/5] RUN apt-get update && apt-get install -y docker 0.0s  
⇒ [jenkins] exporting to image 0.0s  
⇒ ⇒ exporting layers 0.0s  
⇒ ⇒ writing image sha256:e76623a226423174739a14f401e5166490eae778a992 0.0s  
[+] Running 4/40 docker.io/library/infra-jenkins 0.0s  
✓ Service jenkins Built 0.8s  
✓ Network infra_web Created 0.2s  
✓ Container infra-jenkins-1 Started 0.4s  
✓ Container infra-traefik-1 Started 0.5s
```