
Архитектура распределённых приложений

Расчетно-графическая работа

Выполнил: студент группы ИП-312
Дорогин Н. С.

Проверил: преподаватель Кафедры вычислительных систем
Челканова Т.В.

Общая часть:

1. Реализовать CI и CD на базе репозитория Gitlab.
2. Продемонстрировать работу CI/CD, показать обновление ресурсов и приложения/приложений в кластере.
3. Файлы конфигурации/ресурсов k8s разместить в отдельной директории репозитория.

Задание на оценку 5:

1. Доработать в CI этапы `diff` и `deploy`. Для развертывания использовать Helmfile.
2. Создать чарт для развертывания приложения. Привести значения по умолчанию в соответствии с именем проекта.
3. Тег собранного ранее в CI образа передавать через аргументы командной строки.
4. Используя механизм окружений (`environment`) Helmfile настроить CI таким образом, чтобы разворачивалось два варианта окружения приложений. Все окружения должны работать одновременно. Должна быть возможность запустить деплой каждого из окружений независимо друг от друга. Приложения должны иметь уникальный url и, например, различные значения переменных окружения.

5. Этап deploy должен запускаться по нажатию кнопки.

6. Переменные определенные в environment не должны повторять структуру values. Файлы содержащие environment-переменные не должны использоваться в секциях values.

Ход работы:

Общая часть:

1) Для реализации CI/CD на базе нашего Gitlab-репозитория нам необходимо, чтобы из него мы имели доступ к кластеру Hawk.

С этой целью мы создадим в нашем репозитории специальную переменную типа File, которая будет хранить в себе, конфигурацию, предоставляющую ему доступ.

The screenshot shows the 'Variables' section of the GitLab CI/CD settings. On the left, there's a sidebar with navigation links like General, Integrations, Webhooks, etc., and a 'CI/CD' section which is currently selected. The main area displays four sections: Artifacts, Variables, Pipeline trigger tokens, and Deploy freezes. The 'Variables' section is expanded, showing a table for CI/CD Variables. One entry, 'KUBECONFIG', is listed with a value of '.....' and an environment of 'All (default)'. There are buttons for 'Reveal values' and 'Add variable'.

Key	Value	Environments	Actions
KUBECONFIG	All (default)	

This screenshot shows the same 'Variables' section, but with a different configuration. The 'KUBECONFIG' variable now has its value masked as '.....'. Below the table, there's a note about masking variables for security. The page also includes sections for Pipeline trigger tokens and Deploy freezes.

Variables can be accidentally exposed in a job log, or maliciously sent to a third party server. The masked variable feature can help reduce the risk of accidentally exposing variable values, but is not a guaranteed method to prevent malicious users from accessing variables. [How can I make my variables more secure?](#)

Key	Value	Environments	Actions
KUBECONFIG	All (default)	

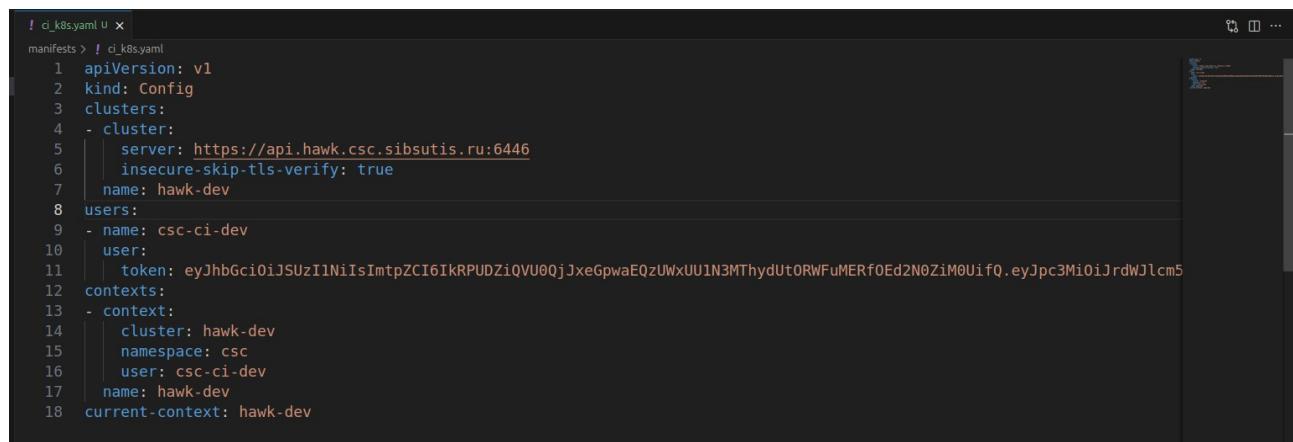
Содержание файла конфигурации берём в wiki кластера, которую мы использовали в лабораторной работе №3.

(<https://wiki.csc.sibsutis.ru/en/Hawk>)

Файл конфигурации имеет следующий вид.

(upd: не надо создавать этот файл в проекте и коммитить его, текст конфига надо вставить в value переменной в самом гитлабе, похоже ещё раз об этом упомянуто будет)

ci_k8s.yaml



```
! ci_k8s.yaml u | manifests > ! ci_k8s.yaml
1  apiVersion: v1
2  kind: Config
3  clusters:
4    - cluster:
5      |       server: https://api.hawk.csc.sibsutis.ru:6446
6      |       insecure-skip-tls-verify: true
7      |       name: hawk-dev
8  users:
9    - name: csc-ci-dev
10   user:
11     |       token: eyJhbGciOiJSUzIlNiIsImtpZCI6IkRPUDZiQVU0QjJxeGpwaEQzUWxUU1N3MThydUt0RWFuMERf0Ed2N0ZiM0UiFQ.eyJpc3MiOiJrdWJlc5
12 contexts:
13   - context:
14     |       cluster: hawk-dev
15     |       namespace: csc
16     |       user: csc-ci-dev
17     |       name: hawk-dev
18   current-context: hawk-dev
```

Для заполнения поля token нам нужно было получить специальный токен для CI.

Делается это с помощью следующей команды внутри кластера (можно на той же wiki скопировать):

```
ip312s07@hawk-access:~$ kubectl --kubeconfig hawk_k8s.yaml -n ip312 get secret ci-token --template={{.data.token}} | base64 -d
```

Теперь у нашего Gitlab репозитория будет доступ к кластеру Hawk.

Более того использование утилиты kubectl по умолчанию будет относиться именно к кластеру Hawk.

Выполнение тех команд, которые мы запускали вручную внутри кластера в третьей лабораторной при каждом изменении образа, можно будет автоматизировать прописав их один раз CI.

Так же нам теперь не придётся переписывать манифесты в самом кластере — kubectl будет использовать манифесты, которые прописаны у нас в репозитории.

Теперь нужно обеспечить обновление образа, который используется в нашем основном манифесте my_Deployment.yaml. Этот образ нужно сделать переменной.

Сам по себе манифест не поддерживает использование переменных, зато их поддерживают шаблоны.

my_Deployment.template.yaml

Также нужно создать просто my_Deployment.yaml, чтобы манифест, получившийся в результате подстановки образа в шаблон было чему присвоить.

Изменение образа в главном манифесте готово. Любые другие изменения, вносимые нами в манифестах в репозитории также будут сразу же отражаться на развертывании приложения, потому что, как уже было сказано, используются именно манифесты из репозитория.

Теперь для обновления приложения нам нужно сделать так, чтобы изменённые манифесты были сразу же применены.

Для этого в CI добавляем этап deploy:

```
  .gitlab-ci.yml
    stages:
      - build
      - deploy

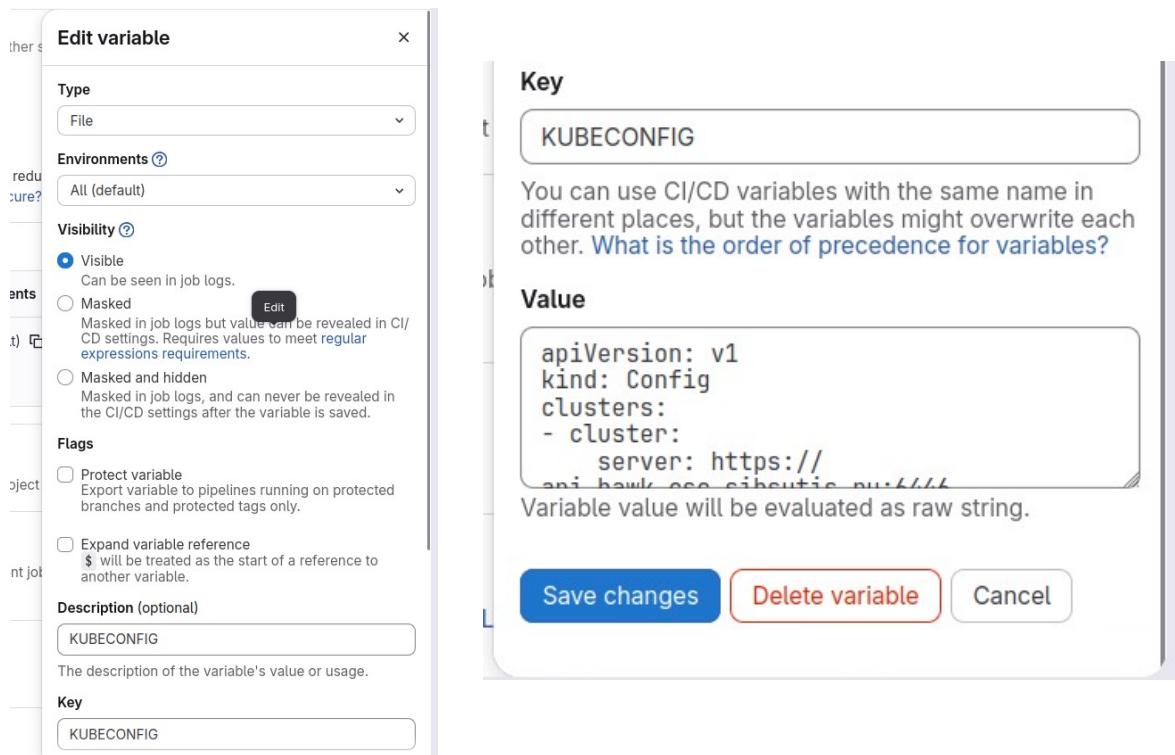
70  deploy-job:
71    stage: deploy
72    image: registry.csc.sibsutis.ru/csc-public/internal-container-images/k8s-tools:latest
73    before_script:
74      - export KUBECONFIG=manifests/ci_k8s.yaml
75      - apt-get update && apt-get install -y gettext-base
76    script:
77      - echo "START DEPLOYING..."
78      - |
79        if [ -n "$CI_COMMIT_TAG" ]; then
80          IMAGE_TAG=$CI_COMMIT_TAG
81        else
82          IMAGE_TAG=${CI_COMMIT_SHORT_SHA:0:8}
83        fi
84        echo "READING THE IMAGE TAG"
85      - export DEPLOYMENT_IMAGE=${HAWK_REGISTRY_IMAGE}:${IMAGE_TAG}
86      - envsubst < manifests/my_Deployment.template.yaml > manifests/my_Deployment.yaml
87
88      - echo "IMAGE WAS SUCCESFULLY REPLACED TO ${HAWK_REGISTRY_IMAGE}:${IMAGE_TAG}"
89
90      - kubectl -n ip312 apply -f manifests/my_Ingress.yaml
91      - kubectl -n ip312 apply -f manifests/my_Service.yaml
92      - kubectl -n ip312 apply -f manifests/my_Deployment.yaml
93
94      - echo "CHANGES WERE SUCCESFULLY APPLIED"
95    rules:
96      - when: manual
```

Присваиваем *KUBECONFIG* наш файл с конфигурацией и теперь команды kubectl по умолчанию действуют на кластер Hawk.

(upd:

строчка — - export KUBECONFIG=manifests/ci_k8s.yaml лишняя.

Вместо неё надо зайди на гитлабе в нашу переменную KUBECONFIG и сделать следующие настройки:



(текст конфига надо вставить в значение переменной, а файл конфига в проекте нам не нужен, его надо удалить в целях безопасности)

С помощью *envsubst* из *gettext-base* и присвоенного переменной *DEPLOYMENT_IMAGE* имени нашего нового образа мы из шаблона *my_Deployment.template.yaml* заполняем манифест *my_Deployment.yaml*.

Применяем манифести.

2) Продемонстрируем работу CI/CD.

Для этого попробуем внести в наше приложение некоторые изменения.

Вот как оно выглядело раньше:

	id	postId	name	email	body
	1	1	fsngrhd	sgdh@fgh	sdfgh

Поменяем цвета для заголовков таблицы:

```

    <pre>
        <div> src </div>
        <div> # AddCommentForm.css </div>
        <div> AddCommentForm.jsx </div>
        <div> Application.js </div>
        <div> # DataSet.css <span> M </span> </div>
        <div> DataSet.jsx </div>
        <div> DeleteButton.jsx </div>
        <div> index.js </div>
        <div> MyName.jsx </div>
        <div> router.js </div>
        <div> babel.config.json </div>
        <div> jest.setup.js </div>
        <div> package-lock.json </div>
        <div> package.json </div>
    </pre>

```

```

        7   font-size: 20px;
        8   table-layout: fixed;
        9 }
      10
      11 .Table td:nth-child(1) {
      12   width: 50px;
      13   padding: 0 10px;
      14 }
      15
      16 .headers {
      17   background-color: blue;
      18   color: white
      19 }
      20
    <pre>
  
```

Search or go to... []

Дорогин Никита Сергеевич / my-simple-distributed-application / Jobs / #110670

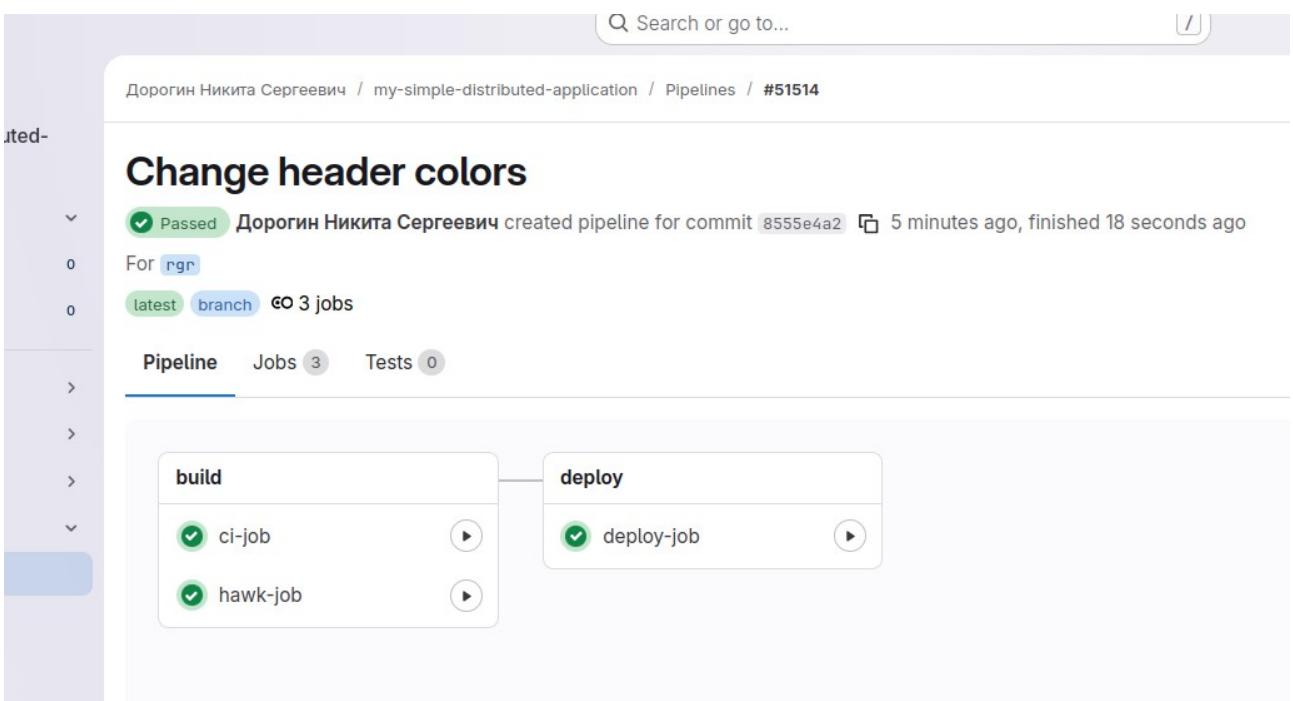
Search visible log output []

```

14 Gitaly correlation ID: 01KB7ZSR6BTA3V5HTTWE9ZG65M
15 Fetching changes with git depth set to 20...
16 Reinitialized existing Git repository in /builds/ip312s07/my-simple-distributed-application/.git/
17 Created fresh repository.
18 Checking out 8555e4a2 as detached HEAD (ref is rgr)...
19 Skipping Git submodules setup
20 Executing "step_script" stage of the job script 00:04
21 Using effective pull policy of [always] for container registry.csc.sibutis.ru/csc-public/internal-container-images/k8s-tools:latest
22 Using docker image sha256:a0d168939a7d15d8893a1123d107891954164c7fffb5e05e405a7468aa453ac7ad for registry.csc.sibutis.ru/csc-public/internal-container-images/k8s-tools:latest with digest registry.csc.sibutis.ru/csc-public/internal-container-images/k8s-tools@sha256:a0d168939a7d15d8893a1123d107891954164c7fffb5e05e405a7468aa453ac7ad
23 /bin/sh: eval: line 169: apt-get: not found
24 $ export KUBECONFIG=manifests/ci_k8s.yaml
25 $ apt-get update && apt-get install -y gettext-base
26 $ echo "START DEPLOYING..."
27 START DEPLOYING...
28 $ if [ -n "$CI_COMMIT_TAG" ]; then # collapsed multi-line command
29 READING THE IMAGE TAG
30 $ export DEPLOYMENT_IMAGE=${HAWK_REGISTRY_IMAGE}:$(IMAGE_TAG)
31 $ envsubst < manifests/my_Deployment.template.yaml > manifests/my_Deployment.yaml
32 $ echo "IMAGE WAS SUCCESSFULLY REPLACED TO ${HAWK_REGISTRY_IMAGE}:$(IMAGE_TAG)"
33 IMAGE WAS SUCCESSFULLY REPLACED TO cr.hawk.csc.sibutis.ru/hawk-dev/ip312s07-my-simple-distributed-application:8555e4a2
34 $ kubectl -n ip312 apply -f manifests/my_Ingress.yaml
35 ingress.networking.k8s.io/ip312s07-ingress unchanged
36 $ kubectl -n ip312 apply -f manifests/my_Service.yaml
37 service/ip312s07-service unchanged
38 $ kubectl -n ip312 apply -f manifests/my_Deployment.yaml
39 deployment.apps/ip312s07-app configured
40 $ echo "CHANGES WERE SUCCESFULLY APPLIED"
41 CHANGES WERE SUCCESFULLY APPLIED
42 Cleaning up project directory and file based variables 00:01
43 Job succeeded

```

Elapsed time: 10 seconds
 Queued: 0 seconds
 Timeout: 1h (from project) ?
 Runner: #11 (-raTMe4E) elrunner
 Source: Push
 Commit 8555e4a2 []
 Change header colors
 Pipeline #51514 [] Running for rgr
 deploy
 Related jobs
 → [] deploy-job



После завершения этапа deploy мой Deployment в соответствии с политикой RollingUpdate будет по одной заменять поды со старыми версиями — добавляет новый под, удаляет старый. При просмотре подов может показаться, что реплик на одну больше чем нужно, но так и должно быть.

```
vpstechka=7f94790080-7111C ip312s07@hawk-access:~$ kubectl --kubeconfig hawk_k8s.yaml -n ip312 get pods
NAME          READY   STATUS    RESTARTS   AGE
endviyou-db6b94dcd-ddc6w   1/1     Running   1 (7d11h ago)   10d
endviyou-db6b94dcd-k972r   1/1     Running   1 (7d11h ago)   10d
ip312s06-app-7dbc9c66f-hkkrx 1/1     Running   1 (7d11h ago)   9d
ip312s06-app-7dbc9c66f-jggrg 1/1     Running   1 (7d11h ago)   15d
ip312s07-app-55f868db78-stn9x 0/1     Running   0           36s
ip312s07-app-7bc44b6bcd-pgdg7 1/1     Running   1 (46m ago)    47m
ip312s07-app-7bc44b6bcd-sx6z7 1/1     Running   0           48m
ip312s07-app-7bc44b6bcd-z5n72 1/1     Running   0           47m
ip312s08-app-7cbf447584-kwxcm 1/1     Running   0           2d10h
```

По завершении процесса замены всех подов, их снова будет столько, сколько указано в replicas у манифеста Deployment, в моём случае — 3.

```
ip312s07@hawk-access:~$ kubectl --kubeconfig hawk_k8s.yaml -n ip312 get pods
NAME          READY   STATUS    RESTARTS   AGE
endviyou-db6b94dcd-ddc6w   1/1     Running   1 (7d11h ago)   10d
endviyou-db6b94dcd-k972r   1/1     Running   1 (7d11h ago)   10d
ip312s06-app-7dbc9c66f-hkkrx 1/1     Running   1 (7d11h ago)   9d
ip312s06-app-7dbc9c66f-jggrg 1/1     Running   1 (7d11h ago)   15d
ip312s07-app-55f868db78-qjstb 1/1     Running   0           20m
ip312s07-app-55f868db78-stn9x 1/1     Running   2 (21m ago)    22m
ip312s07-app-55f868db78-wzw87 1/1     Running   0           20m
ip312s08-app-7cbf447584-kwxcm 1/1     Running   0           2d8h
```

Процесс замены можно отслеживать:

```
ip312s07@hawk-access:~$ kubectl --kubeconfig hawk_k8s.yaml rollout status deployment/ip312s07-app -n ip312
Waiting for deployment "ip312s07-app" rollout to finish: 1 out of 3 new replicas have been updated...
```

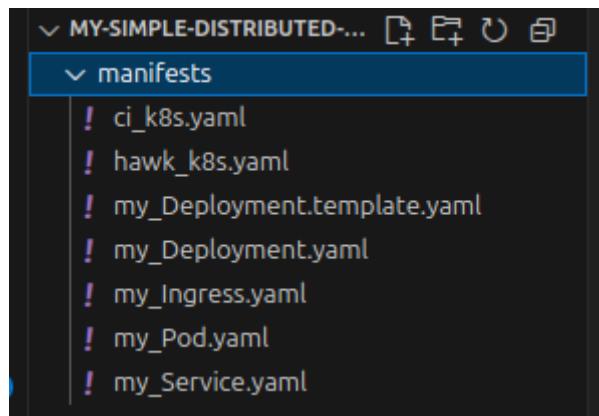
```
ip312s07@hawk-access:~$ kubectl --kubeconfig hawk_k8s.yaml rollout status deployment/ip312s07-app -n ip312
Waiting for deployment "ip312s07-app" rollout to finish: 1 out of 3 new replicas have been updated...
Waiting for deployment "ip312s07-app" rollout to finish: 1 out of 3 new replicas have been updated...
Waiting for deployment "ip312s07-app" rollout to finish: 1 out of 3 new replicas have been updated...
Waiting for deployment "ip312s07-app" rollout to finish: 2 out of 3 new replicas have been updated...
Waiting for deployment "ip312s07-app" rollout to finish: 2 out of 3 new replicas have been updated...
Waiting for deployment "ip312s07-app" rollout to finish: 2 out of 3 new replicas have been updated...
Waiting for deployment "ip312s07-app" rollout to finish: 1 old replicas are pending termination...
Waiting for deployment "ip312s07-app" rollout to finish: 1 old replicas are pending termination...
deployment "ip312s07-app" successfully rolled out
```

Новый вид таблицы (записи из прошлой таблицы не сохраняются):

	<input type="text"/> Name	<input type="text"/> Email	<input type="button" value="Add Comment"/>
	<input type="text"/> Comment body		<input type="button" value="Delete Selected (0)"/>
<input type="button" value="Comment"/>			
	id	postId	name
	1	1	dfxcg
			email
			sfzdgf@fgh
			body
			fghjk

(Интересный факт: если много раз нажимать на кнопку обновления страницы, вас перенесёт на страницу, где работает приложение с другим подом. Дело в том, что все поды обслуживаются одним и тем же Service, которым в свою очередь пользуется один и тот же Ingress и у них, получается, один и тот же домен)

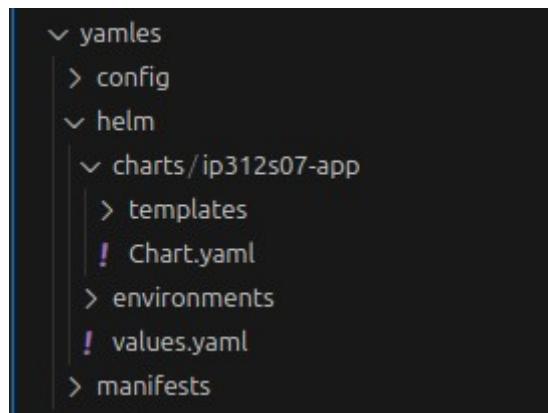
3) Как уже можно было заметить из CI, все файлы конфигурации/ресурсов лежат в отдельной директории репозитория:



Задание на оценку 5:

1, 2) Реализуем этапы diff и deploy с помощью Helmfile.

Для начала изменим структуру директории в которой лежат наши ресурсы и конфигурации



Содержанием папки manifests мы больше пользоваться не будем, поэтому на неё можно не обращать внимания.

Папка config хранит hawk_k8s.yaml и ci_k8s.yaml. // ЭТОГО БОЛЬШЕ НЕ НУЖНО

Папка helm содержит всё то, чем будет пользоваться наш Helmfile для развертывания приложения.

Сам helmfile.yaml лежит в корневой директории.

Теперь пройдёмся по содержанию необходимых файлов:

helmfile.yaml

```
releases:
  - name: ip312s07-app-{{ .Environment.Name }}
    namespace: ip312
    chart: ./yamles/helm/charts/ip312s07-app
    values:
      - ./yamles/helm/values.yaml.gotmpl
```

Helmfile представляет собой инструмент для управления Helm-релизами.

Helm — инструмент для создания манифестов из шаблонов и подаваемых им значений values.

Helm-чарт — набор тех самых шаблонов.

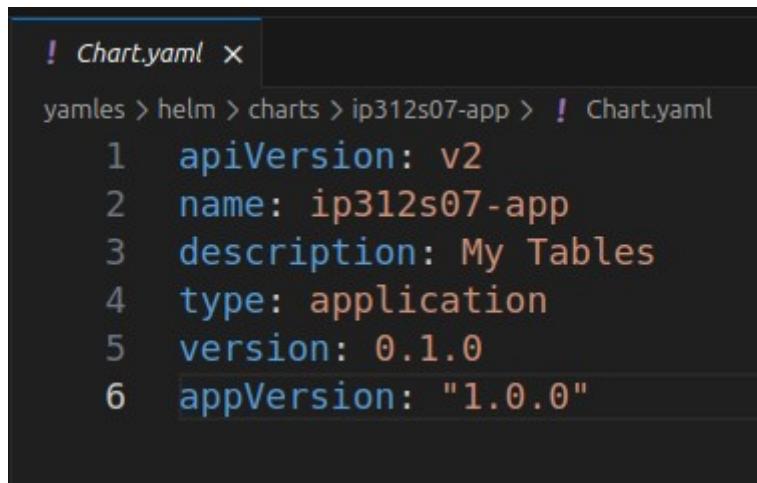
Helm-релиз это «новое состояние» приложения при применении тех самых манифестов, которые сделал Helm.

Поэтому самое важное, что нужно указать для helmfile — это релизы.

У релиза есть имя (name), пространство имен (namespace) в котором он происходит, набор шаблонов, из которых создаются манифести, применение которых приведёт приложение в «новое состояние» (chart), и значения, подстановка которых в шаблоны сделает нам манифести.

Важно! В поле chart надо указывать директорию, в которой лежит чарт, само название файла вписывать не надо — Helmfile подцепит чарт автоматически.

Chart.yaml



```
! Chart.yaml x
yamles > helm > charts > ip312s07-app > ! Chart.yaml
1   apiVersion: v2
2   name: ip312s07-app
3   description: My Tables
4   type: application
5   version: 0.1.0
6   appVersion: "1.0.0"
```

Сам по себе он не содержит для нас особо полезной информации, только метаданные, однако его наличие необходимо для корректной работы.

По соседству с ним в той же директории *charts/ip312s07* располагается папка с шаблонами, которые Helmfile сам находит, зная только местонахождение Chart.yaml.

Эти шаблоны представляют собой те же самые манифесты, которые мы писали ранее, только в некоторых местах с переменными вместо конкретных значений:

templates/t_my_deployment.yaml

```
! t_my_deployment.yaml ×
yamles > helm > charts > ip312s07-app > templates > ! t_my_deployment.yaml
 1  apiVersion: apps/v1
 2  kind: Deployment
 3  metadata:
 4    labels:
 5      owner: ip312s07
 6      environment: {{ .Release.Name }}
 7      name: {{.Release.Name}}
 8      namespace: {{.Release.Namespace}}
 9  spec:
10    replicas: {{.Values.replicaCount}}
11    revisionHistoryLimit: {{.Values.revisionHistoryLimit}}
12    selector:
13      matchLabels:
14        app: ip312s07-tables
15        owner: ip312s07
16        environment: {{ .Release.Name }}
17    strategy:
18      rollingUpdate:
19        maxSurge: 25%
20        maxUnavailable: 25%
21      type: RollingUpdate
22
23    template:
24      metadata:
25        labels:
26          app: ip312s07-tables
27          owner: ip312s07
28          environment: {{ .Release.Name }}
29    spec:
30      containers:
31        - name: ip312s07-table
32          image: "{{.Values.image.repository}}:{{.Values.image.tag}}"
33          imagePullPolicy: IfNotPresent
34          command: {{.Values.command | toJson}}
35          securityContext:
36            allowPrivilegeEscalation: false
37            capabilities:
38              drop: ["ALL"]
39          resources:
```

```
40     requests:
41       cpu: {{.Values.resources.requests.cpu}}
42       memory: {{.Values.resources.requests.memory}}
43     limits:
44       cpu: {{.Values.resources.limits.cpu}}
45       memory: {{.Values.resources.limits.memory}}
46   readinessProbe:
47     httpGet:
48       path: /
49       port: {{.Values.app.port}}
50       failureThreshold: 3
51       initialDelaySeconds: 15
52       periodSeconds: 10
53   livenessProbe:
54     httpGet:
55       path: /
56       port: {{.Values.app.port}}
57       failureThreshold: 3
58       initialDelaySeconds: 15
59       periodSeconds: 10
60   startupProbe:
61     httpGet:
62       path: /
63       port: {{.Values.app.port}}
64       failureThreshold: 3
65       initialDelaySeconds: 15
66       periodSeconds: 10
67
```

templates/t_my_service.yaml

```
! t_my_service.yaml x
yamles > helm > charts > ip312s07-app > templates > ! t_my_service.yaml
1 apiVersion: v1
2 kind: Service
3 metadata:
4   labels:
5     app: ip312s07-tables
6     owner: ip312s07
7     name: "{{.Release.Name}}-service"
8     namespace: {{.Release.Namespace}}
9 spec:
10   ports:
11     - name: http
12       port: {{.Values.service.port}}
13       protocol: TCP
14       targetPort: {{.Values.service.targetPort}}
15   selector:
16     app: ip312s07-tables
17     owner: ip312s07
18     environment: {{ .Release.Name }}
19   type: {{.Values.service.type}}
```

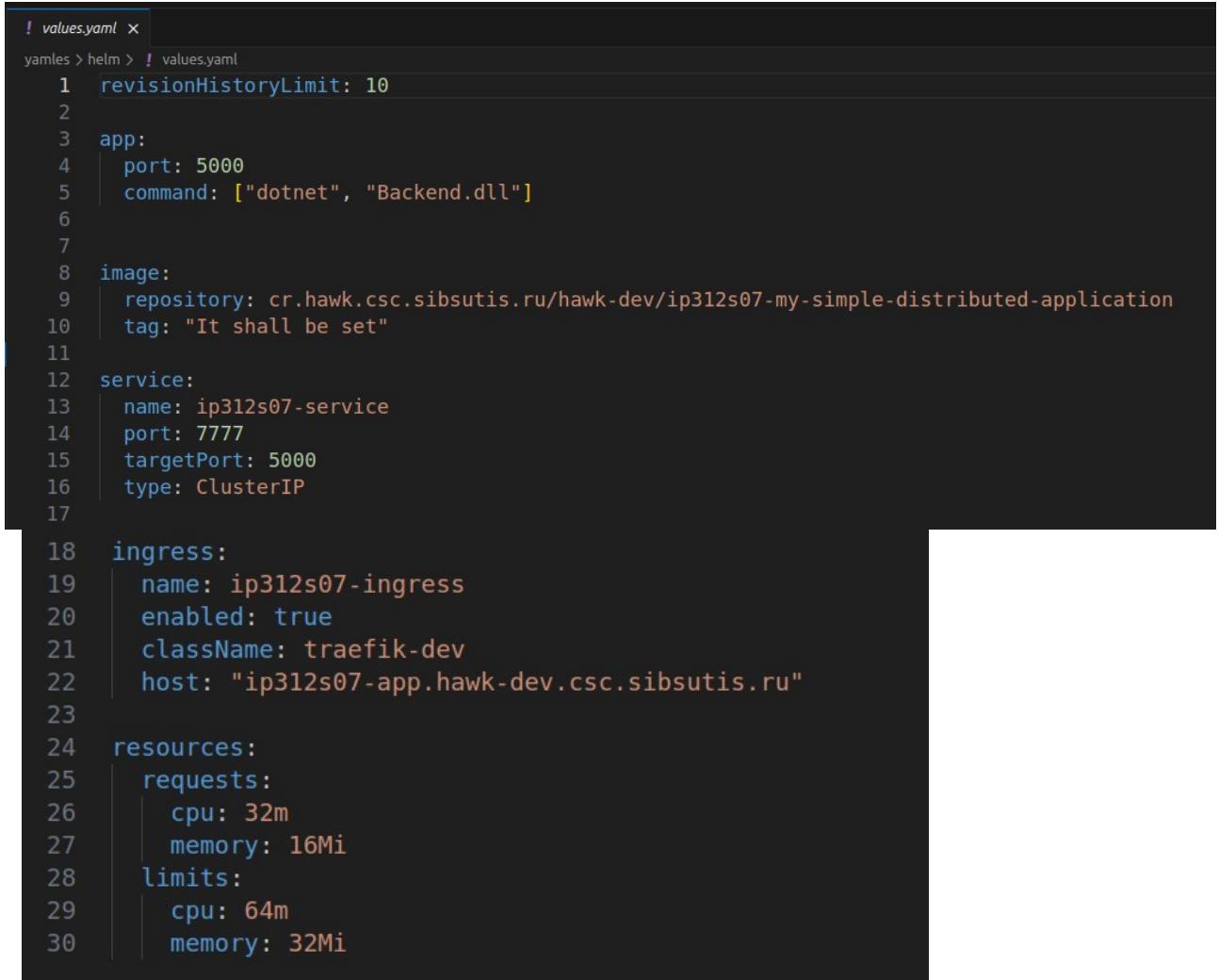
templates/t_my_ingress.yaml

```
! t_my_ingress.yaml x
yamles > helm > charts > ip312s07-app > templates > ! t_my_ingress.yaml
1  apiVersion: networking.k8s.io/v1
2  kind: Ingress
3  metadata:
4    labels:
5      owner: ip312s07
6      name: "{{ .Release.Name }}-ingress"
7      namespace: {{ .Release.Namespace }}
8  spec:
9    ingressClassName: traefik-dev
10   rules:
11     - host: {{ .Values.ingressHost | quote }}
12       http:
13         paths:
14           - path: /
15             pathType: Prefix
16             backend:
17               service:
18                 name: "{{ .Release.Name }}-service"
19                 port:
20                   number: {{ .Values.service.port }}
```

На 2 уровня выше, вне папки charts располагается файл со значениями values.yaml.gotmpl:

values.yaml (в конец названия необходимо добавить .gotmpl)

(мой VS code не знает формат .yaml.gotmpl, поэтому текст у меня в итоге подсвечивается одним цветом, как в обычном текстовом файле, а не как на скринах ниже)



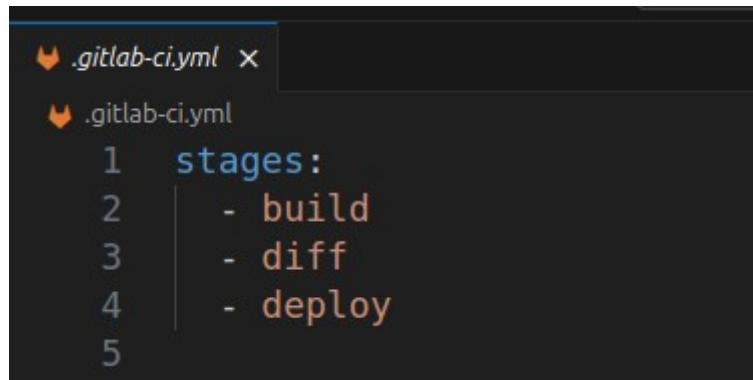
```
! values.yaml x
yamles > helm > ! values.yaml
1 revisionHistoryLimit: 10
2
3 app:
4   port: 5000
5   command: ["dotnet", "Backend.dll"]
6
7
8 image:
9   repository: cr.hawk.csc.sibsutis.ru/hawk-dev/ip312s07-my-simple-distributed-application
10  tag: "It shall be set"
11
12 service:
13   name: ip312s07-service
14   port: 7777
15   targetPort: 5000
16   type: ClusterIP
17
18 ingress:
19   name: ip312s07-ingress
20   enabled: true
21   className: traefik-dev
22   host: "ip312s07-app.hawk-dev.csc.sibsutis.ru"
23
24 resources:
25   requests:
26     cpu: 32m
27     memory: 16Mi
28   limits:
29     cpu: 64m
30     memory: 32Mi
```

Здесь мы просто записываем значения для полей, существующих в шаблоне (не обязательно всех, какие-то значения можно указывать по умолчанию прямо в шаблоне, если мы не хотим менять их в зависимости от values).

В image.tag пишем несуразную строку, поскольку тэг мы будем заменять, подавая его вход команде helmfile через аргументы.

Теперь необходимо прописать в CI выполнение всего вышеописанного.

Для начала добавим новые стадии.



```
1 stages:
2   - build
3   - diff
4   - deploy
```

Весь этап diff (помимо считывания тэга, которое у нас везде есть) представляет собой одну строчку:

helmfile diff

```
diff-job:
  stage: diff
  image: registry.csc.sibsutis.ru/csc-public/internal-container-images/k8s-tools:latest
  before_script:
    - export KUBECONFIG=yamles/config/ci_k8s.yaml
  script:
    - echo "DIFF:"
    - |
      if [ -n "$CI_COMMIT_TAG" ]; then
        IMAGE_TAG=$CI_COMMIT_TAG
      else
        IMAGE_TAG=${CI_COMMIT_SHORT_SHA:0:8}
      fi
      echo "READING THE IMAGE TAG"
    - helmfile diff --set image.tag=${IMAGE_TAG}
  rules:
    - when: always
```

Вывод этой команды в пайплайне выглядит примерно вот так:

```

28 $ helmfile -e env-one diff --set image.tag=${IMAGE_TAG}
29 Building dependency release=ip312s07-app, chart=yamles/helm/charts/ip312s07-app
30 Comparing release=ip312s07-app, chart=yamles/helm/charts/ip312s07-app, namespace=ip312
31 ip312, ip312s07-app, Deployment (apps) has changed:
32 # Source: ip312s07-app/templates/t_my_deployment.yaml
33 apiVersion: apps/v1
34 kind: Deployment
35 metadata:
36   labels:
37     owner: ip312s07
38     name: ip312s07-app
39     namespace: ip312
40 spec:
41   replicas:
42   revisionHistoryLimit: 10
43   selector:
44     matchLabels:
45       app: ip312s07-tables
46       owner: ip312s07
47   strategy:
48     rollingUpdate:
49       maxSurge: 25%
50       maxUnavailable: 25%
51     type: RollingUpdate
52
53 template:
54   metadata:
55     labels:
56       app: ip312s07-tables
57       owner: ip312s07
58   spec:
59     containers:
60     - name: ip312s07-table
61     - image: "cr.hawk.csc.sibsutis.ru/hawk-dev/ip312s07-my-simple-distributed-application:7255be4a"
62     + image: "cr.hawk.csc.sibsutis.ru/hawk-dev/ip312s07-my-simple-distributed-application:e9ee5c86"

```

Самое главное здесь в строчках 61-62. (на флаг -е пока не обращаем внимания, про него чуть дальше)

Изменение образа (а это собственно единственное изменение, которое мы вносим в манифесты из одной версии в другую) продемонстрировано.

Теперь переделаем этап деплоя:

```

deploy-job:
  stage: deploy
  image: registry.csc.sibsutis.ru/csc-public/internal-container-images/k8s-tools:latest
  before_script:
    - export KUBECONFIG=yamles/config/ci_k8s.yaml
  script:
    - echo "START DEPLOYING..."
    - |
      if [ -n "$CI_COMMIT_TAG" ]; then
        IMAGE_TAG=$CI_COMMIT_TAG
      else
        IMAGE_TAG=${CI_COMMIT_SHORT_SHA:0:8}
      fi
      echo "READING THE IMAGE TAG"

    - echo "USING TAG ${IMAGE_TAG}"i
    - helmfile apply --set image.tag=${IMAGE_TAG}

    - echo "CHANGES WERE SUCCESFULLY APPLIED"
  rules:
    - when: manual

```

Три применения kubectl заменились одним применением helmfile.
CI готов.

3) Как уже можно было заметить, в командах helmfile мы подаём на вход тэг образа в качестве аргумента командной строки:

```
else
  IMAGE_TAG=${CI_COMMIT_SHORT_SHA:0:8}
fi
echo "READING THE IMAGE TAG"
- helmfile diff --set image.tag=${IMAGE_TAG}

rules:
  - echo "READING THE IMAGE TAG"

  - echo "USING TAG ${IMAGE_TAG}" i

  - helmfile apply --set image.tag=${IMAGE_TAG}
```

Поданный нами тэг присваивается полю image.tag в values.

4) Теперь нам нужно реализовать для нашего приложения 2 независимых друг от друга окружения.

Добавляем environments в helmfile.yaml.

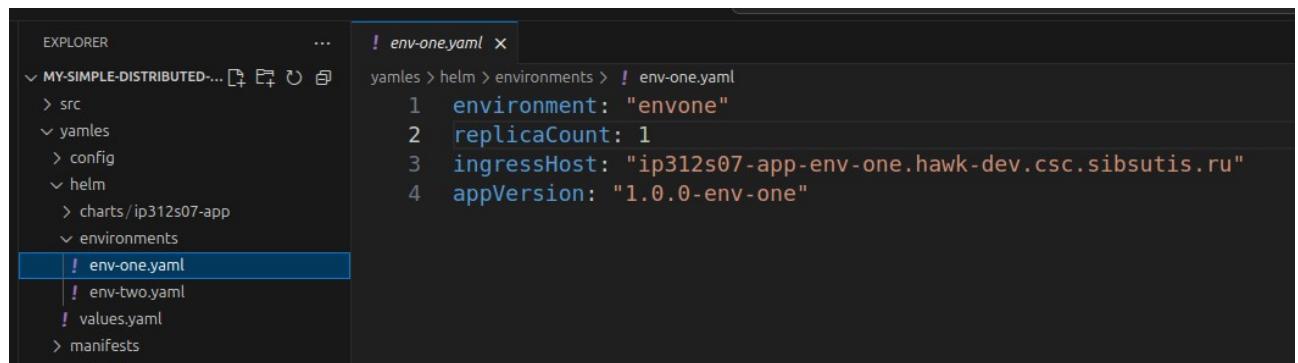
Важно! Окружения должны идти перед релизами и находится от них в отдельной секции.

Секции делятся с помощью ---

```
! helmfile.yaml x
! helmfile.yaml

1
2 environments:
3   env-one:
4     values:
5       - yamles/helm/environments/env-one.yaml
6
7   env-two:
8     values:
9       - yamles/helm/environments/env-two.yaml
10
11 ---
12
13 releases:
14   - name: ip312s07-app-{{ .Environment.Name }}
15     namespace: ip312
16     chart: ./yamles/helm/charts/ip312s07-app
17     values:
18       - ./yamles/helm/values.yaml.gotmpl
```

Создаём файлы окружений:



The screenshot shows a code editor with two panes. The left pane is the Explorer view, showing a project structure with folders like 'src', 'yamles', 'helm', and 'environments'. Inside 'environments', there are files: 'env-one.yaml' (selected), 'env-two.yaml', and 'values.yaml'. The right pane shows the content of 'env-one.yaml':

```
environment: "envone"
replicaCount: 1
ingressHost: "ip312s07-app-env-one.hawk-dev.csc.sibsutis.ru"
appVersion: "1.0.0-env-one"
```

В нашем случае окружения будут отличаться друг от друга url приложений и некоторыми строчными полями.

Важно! Поля values и environment не должны иметь схожей структуры! releases.values не должен содержать ссылок на файлы окружения!

В values ingress в поле host теперь будет записываться адрес из Environment.

```
18   ingress:
19     name: ip312s07-ingress
20     enabled: true
21     className: traefik-dev
22     host: {{.Environment.Values.ingressHost | quote}}
```

Также имеет место необходимость добавить для деплоимента и его контейнеров специальный label,

```
metadata:
  labels:
    owner: ip312s07
    environment: {{ .Release.Name }}
```

чтобы selector у Service не выбирал для обслуживания поды из чужого окружения и данные приложений не смешивались.

```
selector:
  app: ip312s07-tables
  owner: ip312s07
  environment: {{ .Release.Name }}
```

Чтобы окружения можно было запускать независимо друг от друга и деплоить так же независимо сделаем для каждого окружения свою job в CI.

```
.gitlab-ci.yml x
.gitlab-ci.yml
109 env1-deploy-job:
110   script:
111     - echo "START DEPLOYING..."
112     - |
113       if [ -n "$CI_COMMIT_TAG" ]; then
114         IMAGE_TAG=$CI_COMMIT_TAG
115       else
116         IMAGE_TAG=${CI_COMMIT_SHORT_SHA:0:8}
117       fi
118       echo "READING THE IMAGE TAG"
119
120     - echo "USING TAG ${IMAGE_TAG}"i
121
122     - helmfile -e env-one apply --set image.tag=${IMAGE_TAG}
123
124     - echo "CHANGES WERE SUCCESFULLY APPLIED"
125   rules:
126     - when: manual
127
128 env2-deploy-job:
129   stage: deploy
130   image: registry.csc.sibutis.ru/csc-public/internal-container-images/k8s-tools:latest
131   before_script:
132     - export KUBECONFIG=yamles/config/ci_k8s.yaml
133   script:
134     - echo "START DEPLOYING..."
135     - |
136       if [ -n "$CI_COMMIT_TAG" ]; then
137         IMAGE_TAG=$CI_COMMIT_TAG
138       else
139         IMAGE_TAG=${CI_COMMIT_SHORT_SHA:0:8}
140       fi
141       echo "READING THE IMAGE TAG"
142
143     - echo "USING TAG ${IMAGE_TAG}"
144     - helmfile -e env-two apply --set image.tag=${IMAGE_TAG}
145
146     - echo "CHANGES WERE SUCCESFULLY APPLIED"
147   rules:
```

Они абсолютно идентичны, но с флагом `-e` в команде `helmfile` мы можем указать конкретное окружение, к которому применяются изменения. (`diff` я тоже раздвоил)

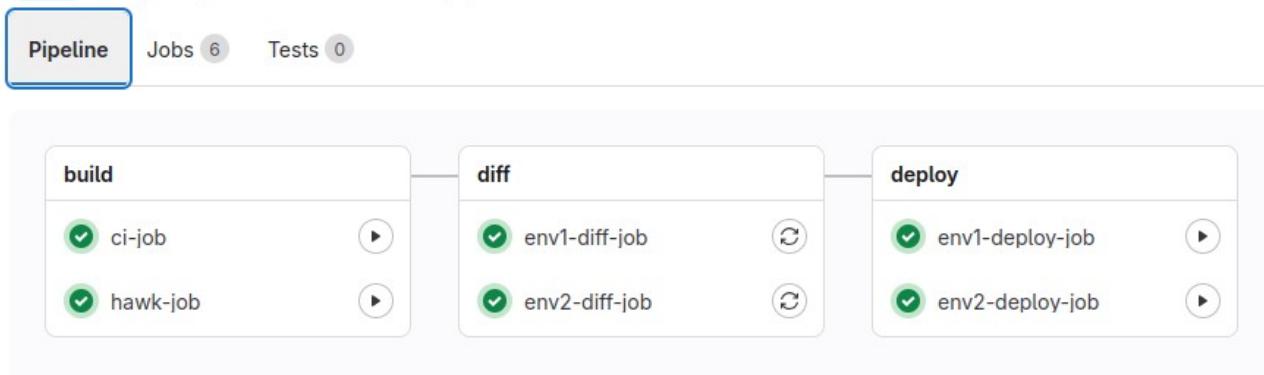
Теперь давайте попробуем изменить цвет фона заголовков таблицы, допустим, на оранжевый:

Make orange color

Passed Дорогин Никита Сергеевич created pipeline for commit 2820bc9f 9 hours ago, finished 9 hours ago

For rgr-5

branch eo 6 jobs 13 minutes 14 seconds, queued for 10 seconds



```
92     failureThreshold: 3
93     initialDelaySeconds: 15
94     periodSeconds: 10
95     startupProbe:
96       httpGet:
97         path: /
98         port: 5000
99         failureThreshold: 3
100        initialDelaySeconds: 15
101       periodSeconds: 10
102 Upgrading release=ip312s07-app-env-one, chart=yamles/helm/charts/ip312s07-app, namespace=ip312
103 Release "ip312s07-app-env-one" has been upgraded. Happy Helm-ing!
104 NAME: ip312s07-app-env-one
105 LAST DEPLOYED: Sun Nov 30 16:51:18 2025
106 NAMESPACE: ip312
107 STATUS: deployed
108 REVISION: 12
109 TEST SUITE: None
110 Listing releases matching ^ip312s07-app-env-one$  
111 ip312s07-app-env-one ip312 11 2025-11-30 16:23:06.146408195 +0000 UTC failed ip312s07-app-0.1.0 1.0.0
112 UPDATED RELEASES:  
113 NAME      NAMESPACE  CHART          VERSION DURATION  
114 ip312s07-app-env-one ip312 ./yamles/helm/charts/ip312s07-app 0.1.0 2s
115 $ echo "CHANGES WERE SUCCESFULLY APPLIED"
116 CHANGES WERE SUCCESFULLY APPLIED
117 Cleaning up project directory and file based variables
118 Job succeeded
```

Commit 2820bc9f Make orange color

Pipeline #51696 Passed for rgr-5

deploy

Related jobs

- env1-deploy-job
- env2-deploy-job

env1-deploy-job (#11126 Таблица + ip312s07-app-env-one.hawk-dev.csc.sibutis.ru

	<input type="text"/> Name	<input type="text"/> Email	<input type="button" value="Add Comment"/>		
	<input type="button" value="Delete Selected (0)"/>				
	id	postId	name	email	body
	1	1	gfg	dvfb@grdhtf	dgfghj

Поды в кластере:

ip312s07-app-env-one-7a01b6557-12-jga	1/1	Running	0	15m
ip312s07-app-env-one-9fd864595-kzrb8	1/1	Running	0	9h
ip312s07-app-env-two-6b849888f9-5ql8t	1/1	Running	0	9h

5) Теперь я хочу, чтобы оранжевой была таблица из первого окружения, а из второго какого-нибудь другого цвета. Заодно продемонстрируем, что deploy запускается только по нажатию кнопки.

(ошибка в описании: env2)

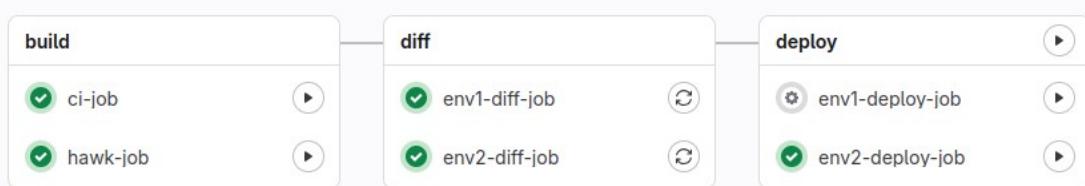
Make pink for env1

Blocked Дорогин Никита Сергеевич created pipeline for commit 29b4b56f ⚡

For rgr-5

latest branch ⚡ 6 jobs

Pipeline Jobs 6 Tests 0



Two browser tabs are shown side-by-side. Both tabs have a header with columns "name", "email", and "b". The left tab has a yellow background for its rows. The right tab has a pink background for its rows. The data in both tables is identical:

name	email	b
gfg	dvfb@grdhtf	dgt
zdxbf	szcvd@sedgrhj	

6) В пункте 2 я уже демонстрировал values.yaml.gotmpl, а в пункте 4 — файл окружения, сделав замечание, что поля у них не совпадают, и helmfile.yaml, где в values релиза только ссылка на файл values.yaml.gotmpl