Launch

Руководство системного администратора

Единая платформа для высокопроизводительной потоковой аналитики, интегрирующее ML-модели для обработки разнородных данных: текста, статических изображений, видеопотока и аудиосигналов

Содержание

Введение

Настоящее руководство предназначено для системных администраторов и содержит полную информацию, необходимую для установки, настройки, эксплуатации и обслуживания платформы Launch.

Термины и сокращения

Argo CD – Инструмент непрерывной доставки для Kubernetes, реализующий GitOps

GPU – Graphics Processing Unit (графический процессор)

Kubernetes – платформа оркестрации контейнеров с открытым исходным кодом

HA (High Availability) – Высокая доступность – архитектура, обеспечивающая непрерывную работу системы

GitOps – Методология управления инфраструктурой через Gitрепозиторий

MinIO – S3-совместимое объектное хранилище для файлов и данных

TLS – Transport Layer Security (протокол защиты транспортного уровня)

Vault – Система управления секретами и конфиденциальными данными

ML – Machine Learning (машинное обучение)

ML-модель – Модель машинного обучения для анализа данных

ВМ – Виртуальная машина

ЦПУ – Центральный процессор

Общие сведения о системе

Платформа Launch предназначена для автоматизации задач, связанных с анализом больших объёмов мультимедийных данных с использованием машинного обучения.

Система создаёт унифицированную среду, в которой любые источники данных могут быть подключены к любым ML-моделям через стандартизированные интерфейсы. Результаты анализа интерпретируются согласно заданной бизнес-логике и передаются потребителям через различные каналы интеграции.

Требования к установке

Данный раздел описывает минимально необходимые вычислительные ресурсы для Launch. Значения представлены для работы с 50 источниками.

Промышленный кластер должен включать следующие группы узлов:

- Control-plane узлы Kubernetes: 3 шт.
- Worker-узлы общего назначения: 3 шт.
- GPU-узел для ML-инференса: 1 шт.
- Узлы хранилища MinIO: 4 шт., каждый с выделенными минимум 4 HDD-дисками под DirectPV.

Суммарная спецификация узлов, на которых размещается кластер:

- **CPU**: 128 ядер (256 потоков)
- **RAM**: 256 ГБ
- **NVMe**: 2 ТБ
- **HDD**: 8 × 4 ТБ
- **GPU**: 3 × 5060Ti

Узлы управления (Control-plane)

- **CPU**: 2 ядра
- **RAM**: 5 ГБ
- **NVMe**: 50 ГБ
- **HDD**: отсутствует

Worker-узлы общего назначения

- **CPU**: 21 ядро
- **RAM**: 43 ГБ
- **NVMe**: 80 ГБ
- **HDD**: отсутствует

GPU-узел для ML-нагрузки

- **CPU**: 6 ядер
- **RAM**: 18 ГБ
- **NVMe**: 140 ГБ
- **GPU**: 1 × NVIDIA (MIG/Full – зависит от конфигурации)

Для примера и ориентира по производительности можно использовать модель GPU NVIDIA GeForce RTX 5060 Ti.

Узлы MinIO

- **CPU**: 3 ядра
- **RAM**: 6 ГБ
- **NVMe**: 50 ГБ
- **HDD под DirectPV**: 4 × 2 ТБ на каждый узел

Подготовка к установке

Проверка перед установкой

1. Если установка производится в ВМ, то необходимо убедиться, что в конфигурации ВМ выбрана модель ЦПУ `native` или `host`, а не виртуализированная:

   ```
   cat /proc/cpuinfo
   ```

   Пример вывода при успешной проверке:

   ```
   processor : 0
   vendor_id : AuthenticAMD
   cpu family : 25
   model  : 24
   model name : AMD Ryzen Threadripper PRO 7975WX 32-Cores
   ...
   ```

2. Проверить поддержку инструкций AVX:

   ```
   cat /proc/cpuinfo |grep avx
   ```

3. На узлах кластера с GPU убедиться, что установлен драйвер Nvidia с поддержкой CUDA версии 12.x, и Nvidia GPU доступен. При этом библиотеки CUDA и Nvidia container toolkit устанавливать не следует.

   ```
   nvidia-smi
   ```

4. Если физический GPU разделяется между несколькими ВМ с использованием технологии "Multi-Instance GPU", следует проверить на ВМ с GPU наличие лицензии Nvidia:

   ```
   nvidia-smi -q |grep -i license
   ```

   Пример вывода команды в случае успешной проверки:

   ```
       vGPU Software Licensed Product
           License Status        : Licensed (Expiry: 2025-11-2 13:15:9 GMT)
   ```

5. Убедиться, что для узлов кластера Kubernetes, на которые устанавливается Launch, работает подтверждение владения доменом DNS Let's Encrypt HTTP-01 challenge, необходимое для выпуска действительных сертификатов TLS. В случае, если выпуск сертификатов TLS Let's Encrypt невозможен, следует озаботиться выпуском и установкой в кластер Kubernetes сертификатов TLS другими способами:

- подтверждение владения доменом методами, отличными от HTTP-01 challenge, например, DNS-01 challenge;
- выпуск сертификатов на узле сети, где работает проверка Let's Encrypt HTTP-01, с последующим созданием секретов Kubernetes, содержащих выпущенные сертификатами в кластере Kubernetes Launch;
- настройка выпуска самоподписанных сертификатов посредством cert-manager в кластере Kubernetes Launch.

6. Убедиться, что от клиентов Launch к узлам кластера Kubernetes Launch разрешён сетевой доступ по нижеперечисленным протоколам и портам:
   - **TCP**: 80, 443;
   - **TCP (для отладки)**: 5432, 5001-5008;
   - **UDP**: 30819.

Подготовка к установке Kubernetes

1. Для всех узлов создаваемого кластера Kubernetes необходимо сделать:

   1.1. Убедиться, что на всех узлах основной сетевой интерфейс имеет одинаковое имя, например, `ens18`:

   ```
   ip addr show
   ```

   1.2. Создать пользователя, для примера здесь и далее в инструкции – `myuser`, с правами выполнения sudo без необходимости ввода пароля; пример файла `/etc/sudoers.d/myuser`:

   ```
   myuser ALL=NOPASSWD: ALL
   ```

   1.3. Подключение SSH по имени узла без необходимости ввода пароля; пример записи в файле `~/.ssh/config`:

   ```
   Host launch-dev-control-1 192.168.75.137
       Hostname 192.168.75.137
       Port 8022
       User myuser
   ```

2. На локальный компьютер установить утилиту `kubectl`:

   ```
   curl -LO "https://dl.k8s.io/release/$(curl -L -s
   https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl" && \
   sudo mv kubectl /usr/bin/kubectl
   ```

3. На локальный компьютер установить утилиту `helm`:

   ```
   curl https://baltocdn.com/helm/signing.asc | gpg --dearmor | sudo tee
   /usr/share/keyrings/helm.gpg > /dev/null && \
   echo "deb [arch=$(dpkg --print-architecture) signed-
   by=/usr/share/keyrings/helm.gpg]
   https://baltocdn.com/helm/stable/debian/ all main" | sudo tee
   /etc/apt/sources.list.d/helm-stable-debian.list && \
   sudo apt-get update && \
   sudo apt-get install helm
   ```

4. На локальном компьютере настроить Bash-completion для утилит `kubectl` и `helm`:

   ```
   sudo sh -c 'kubectl completion bash > /etc/bash_completion.d/kubectl' &&
   \
   sudo sh -c 'helm completion bash > /etc/bash_completion.d/helm' && \
   source /etc/bash_completion.d/kubectl && \
   source /etc/bash_completion.d/helm
   ```

5. На локальный компьютер установить Ansible, Git и другие программы и модули, необходимые для установки Launch:

```
sudo apt install -y ansible git python3-kubernetes python3-hvac
ansible-galaxy collection install community.hashi_vault
```

6. Клонировать на локальный компьютер репозиторий GitOps с конфигурацией установки компонентов Launch:

```
git clone git@gitlab.usetech.ru:UseTech/launch/gitops.git
```

Репозиторий конфигурации GitOps

```
|-- ansible               <-- Ansible-playbook-и
|   |-- initial-setup     <-- Ansible-playbook создания секретов в Vault
|   `-- k3s-ansible       <-- Ansible-playbook установки кластера
Kubernetes
|
`-- apps                  <-- Файлы конфигурации приложений Argo CD
|   `-- usetech           <-- Группа окружений
|       |-- launch-demo   <-- Окружение
|       |   `-- argocd
|       |   |   |-- app.yaml                    <-- Описание приложения
|       |   |   |-- appset.yaml                 <-- Описание набора
приложений
|       |   |   |-- argocd-values.yaml          <-- Параметры Helm-чарта
|       |   |   `-- argocd-virtualservice.yaml  <-- Istio VirtualService
|       |   |-- cert-manager
|       |   |-- ...
|       |   |-- launch-stage-1-app.yaml   <-- Корневое приложение стадии 1
|       |   `-- launch-stage-1-app.yaml   <-- Корневое приложение стадии 2
|       |-- launch-dev
|       |-- launch-qa
|       `-- ...
|
|-- charts                <-- Helm-чарты компонентов Launch
|   |-- argo-cd
|   |-- cert-manager
|   `-- ...
|
|-- clusters              <-- Файлы конфигурации кластеров Kubernetes
|   `-- usetech
|       |-- launch-demo
|       |   |-- group_vars
|       |   |   |-- all.yml      <-- Конфигурация кластера Kubernetes
|       |   |   `-- vault.yaml   <-- Конфигурация секретов Vault
|       |   `-- hosts.ini        <-- Адреса узлов кластера Kubernetes
|       |-- launch-dev
|       `-- ...
|
|-- manual                      <-- Файлы конфигурации для действий,
выполняемых вручную
|   `-- directpv                <-- Оператор DirectPV CSI
|
```

Далее в инструкции по установке Launch все относительные пути указываются от корня локальной копии репозитория GitOps, если явно не сказано иное.

Установка Kubernetes

Установка HA-кластера Kubernetes

1. В локальной копии репозитория GitOps перейти в каталог `clusters` и на основе каталога `usetech/launch-demo` создать новый каталог с Ansible inventory, для примера - `МОЙ/КЛАСТЕР`, описывающим создаваемый кластер Kubernetes:

```
cp -R usetech/launch-demo МОЙ/КЛАСТЕР
```

2. Отредактировать файл `МОЙ/КЛАСТЕР/hosts.ini` и задать параметры подключения к узлам создаваемого кластера Kubernetes, например:

```
[master]
launch-demo-control-1
launch-demo-control-2
launch-demo-control-3

[node]
launch-demo-worker-1
launch-demo-worker-2
launch-demo-worker-3
launch-demo-gpu-1
launch-minio-1
launch-minio-2
launch-minio-3
launch-minio-4

[k3s_cluster:children]
master
node
```

3. Задать параметры конфигурации создаваемого кластера Kubernetes, такие как имя пользователя, с которым осуществляется SSH-подключение к узлам кластера, IP-адрес балансировщика трафика и номера версий программного обеспечения, в файле `МОЙ/КЛАСТЕР/group_vars/all.yml`. Ниже приведён пример значений параметров:

```
k3s_version:  v1.32.9k3s1
ansible_user: myuser
system_timezone: Europe/Moscow
calico_iface: "ens18"
calico_ebpf: true
calico_tag: v3.29.6
apiserver_endpoint: 192.168.75.149
k3s_token: Another-SUPER-DUPER-secret-password
extra_agent_args: >-
  {{ extra_args }}
  --kubelet-arg max-pods=200
kube_vip_tag_version: v0.8.10
metal_lb_ip_range: 192.168.75.150-192.168.75.150
custom_registries: true
custom_registries_yaml: |
  mirrors:
    mirror.gcr.io:
      endpoint:
        - "https://mirror.gcr.io"
```

4. Перейти в каталог `clusters` и запустить установку кластера Kubernetes:

```
ansible-playbook ../ansible/k3s-ansible/site.yml -
i ./МОЙ/КЛАСТЕР/hosts.ini
```

5. Дождаться завершения установки, затем скопировать файл конфигурации подключения к созданному кластеру Kuberenetes на локальный компьютер и проверить подключение:

```
mkdir -p ~/.kube
scp launch-demo-control-1:/home/myuser/.kube/config ~/.kube/launch-demo
export KUBECONFIG=~/.kube/launch-demo
kubectl get nodes
```

Настройка Kubernetes после установки

Добавить для узлов кластера Kubernetes метки:

- `launch-role/infra=true`;
- `launch-role/launch=true`;
- `launch-role/ml=true`;
- `launch-role/minio=true`.

```
kubectl label node launch-demo-worker-1 launch-role/infra=true
kubectl label node launch-demo-worker-2 launch-role/infra=true
kubectl label node launch-demo-worker-3 launch-role/infra=true
kubectl label node launch-demo-worker-1 launch-role/launch=true
kubectl label node launch-demo-worker-2 launch-role/launch=true
kubectl label node launch-demo-worker-3 launch-role/launch=true
kubectl label node launch-demo-gpu-1 launch-role/ml=true
kubectl label node launch-minio-1 launch-role/minio=true
kubectl label node launch-minio-2 launch-role/minio=true
kubectl label node launch-minio-3 launch-role/minio=true
kubectl label node launch-minio-4 launch-role/minio=true
```

Если в дальнейшем в кластер Kubernetes будут добавляться новые узлы, следует назначить метки добавляемых узлов кластера согласно ролям означенных узлов.

Настройка дисков данных для кластера MinIO

Действия, приведённые в данном разделе, следует выполнять только в том случае, если требуется установка HA-кластера MinIO в составе кластера Kubernetes Launch.

1. В локальной копии репозитория GitOps перейти в каталог с файлами конфигурации установки CSI-драйвера DirectPV – `manual/directpv`
2. Установить набор манифестов CSI-драйвера `DirectPV`:

   ```
   kubectl apply -f directpv-install.yaml
   ```

3. Выполнить обнаружение дисков на узлах MinIO, при этом параметр `--nodes` задаёт ограничение области обнаружения по именам узлов кластера Kubernetes, а параметр `--drives` задаёт ограничение по именам дисковых устройств:

   ```
   kubectl directpv discover \
     --nodes launch-minio-{1...4} \
     --drives sd{b...e} \
     --output-file directpv-drives.yaml
   ```

   В результате выполнения команды в текущем каталоге будет создан файл `directpv-drives.yaml`, содержащий данные об обнаруженных дисках.

4. Необходимо убедиться, что были обнаружены именно те диски, которые должны использованы MinIO для хранения данных.
5. Выполнить инициализацию дисков `DirectPV`:

   ```
   kubectl directpv init --dangerous directpv-drives.yaml
   ```

6. Проверить состояние дисков `DirectPV`:

   ```
   kubectl directpv info
   ```

Установка инфраструктурных компонентов и приложения Launch

Актуализация параметров конфигурации

1. Установка GitOps-инструмента непрерывной доставки Argo CD выполняется в несколько этапов.

   1.1. На первом этапе настраивается доступ Argo CD к репозиторию GitOps с конфигурацией установки инфраструктурных и прикладных компонентов Launch, а также устанавливается Helm-чарт Argo CD с временным доступом к интерфейсу управления Argo CD через `kubectl port-forward`.

   1.2. Затем Argo CD автоматически устанавливает необходимые компоненты, такие как Istio и Cert-manager, а также выполняет финальную настройку Argo CD с сертификатом TLS и доступом к интерфейсу управления через Istio ingress gateway.

2. В локальной копии репозитория GitOps перейти в каталог `apps` и на основе каталога `usetech/launch-demo` создать новый каталог с конфигурацией установки компонентов Launch. Для примера далее в инструкции будет использоваться каталог `usetech/launch-demo`.

3. Актуализировать DNS-имя сайта, например, `demo.launch.usetech.ru`, во всех yaml-манифестах и прочих файлах конфигурации в каталоге `apps/usetech/launch-demo`. В частности, необходимо отредактировать файл `apps/usetech/launch-demo/stage-1/argo-cd/argocd-values.yaml` и задать DNS-имя сервера Argo CD в параметре `.global.domain`:

   ```
   global:
     domain: demo.launch.usetech.ru
   ...
   ```

Начальная установка Argo CD

1. В локальной копии репозитория GitOps перейти в каталог с конфигурацией Argo CD:

```
cd apps/usetech/launch-demo/stage-1/argo-cd
```

2. Установить Helm-чарт Argo CD:

```
helm repo add argo https://argoproj.github.io/argo-helm
helm repo update argo
helm upgrade --install \
  --create-namespace \
  --namespace argocd \
  --version 9.1.3 \
  --values argocd-values.yaml \
  --wait \
  argo-cd argo/argo-cd
```

3. Открыть новую консоль и включить port-forwarding для временного доступа к интерфейсу управления Argo CD:

```
kubectl -n argocd port-forward svc/argo-cd-argocd-server 8080:443
```

4. Вернуться к прежней консоли.

5. Вывести на консоль пароль администратора Argo CD:

```
kubectl -n argocd get secret/argocd-initial-admin-secret \
  -o jsonpath='{.data.password}' | base64 -d; echo
```

6. Открыть в веб-браузере временный адрес интерфейса управления Argo CD https://localhost:8080/argocd/ и проверить вход с именем пользователя `admin` и паролем, полученным на предыдущем шаге. На данном шаге следует согласиться на установление небезопасного соединения.

Настройка доступа Argo CD к репозиторию GitOps

1. Создать ключ SSH для доступа к репозиторию Git с конфигурацией установки компонентов Launch:

```
ssh-keygen -t ed25519 -f launch-demo-gitops -C launch-demo-gitops
```

2. В веб-интерфейсе GitLab в меню `Settings -> Repository -> Deploy keys` проекта UseTech/Launch/GitOps добавить публичный ключ SSH `launch-demo-gitops.pub`, созданный на предыдущем шаге.

3. Создать файл `launch-gitops-repo-secret.yaml` с описанием секрета Kubernetes для подключения к репозиторию Git:

```
apiVersion: v1
kind: Secret
metadata:
  name: launch-gitops-repo
  namespace: argocd
  labels:
    argocd.argoproj.io/secret-type: repository
stringData:
  name: launch-gitops-repo
  url: git@gitlab.usetech.ru:UseTech/launch/gitops.git
  sshPrivateKey: |
    -----BEGIN OPENSSH PRIVATE KEY-----
        ДАННЫЕ ПРИВАТНОГО КЛЮЧА
    -----END OPENSSH PRIVATE KEY-----
```

4. Создать секрет Kubernetes, описанный в файле `launch-gitops-repo-secret.yaml`:

```
kubectl -n argocd apply -f launch-gitops-repo-secret.yaml
```

5. В локальной копии репозитория GitOps перейти в каталог, содержащий конфигурацию установки компонентов Launch:

```
cd apps/usetech/launch-demo
```

6. Создать корневое приложение Argo CD, запускающее первую стадию установки компонентов Launch:

```
kubectl -n argocd apply -f launch-stage-1-app.yaml
```

7. Дождаться успешного завершения первой стадии установки компонентов Launch – все приложения должны иметь статус Healthy. Затем подключиться к интерфейсу управления Argo CD.

Подключение к интерфейсу управления Vault

1. Дождаться, когда Argo CD завершит установку хранилища секретов Vault.
2. Вывести на консоль root-токен Vault, хранящийся в секрете Kubernetes `vault-unseal-keys`:

```
kubectl -n vault get secrets vault-unseal-keys \
  -o jsonpath={.data.vault-root} |base64 --decode; echo
```

3. С использованием root-токена подключиться к интерфейсу управления Vault.
4. При необходимости управления Vault через CLI следует сначала установить клиент Vault, а затем выполнить команды для подключения к серверу Vault:

```
export VAULT_ADDR=https://vault.demo.launch.usetech.ru
export VAULT_TOKEN=$(kubectl -n vault get secrets vault-unseal-keys -o
jsonpath={.data.vault-root} |base64 --decode)
vault status
vault secrets list
```

Актуализация имени окружения Launch в настройках Vault и в секретах

Здесь и далее в инструкции, после внесения любых изменений в локальную копию репозитория GitOps необходимо отправить изменения на сервер Git, выполнив команды `git commit` и `git push`, чтобы Argo CD актуализировал свою конфигурацию GitOps.

1. Отредактировать файл `apps/ИМЯ/ОКРУЖЕНИЯ/vault/vault-ha-cluster.yaml`:
   1.1. в разделе `.externalConfig.externalConfig.secrets` задать имя Secret Engine кластера Launch, например, "launch-demo";
   1.2. в разделе `.externalConfig.externalConfig.policies` задать имя политики, например, "allow_read_launch_demo".
2. Отправить на сервер Git изменения в репозитории GitOps.
3. В веб-интерфейсе Argo CD обновить (Refresh) конфигурацию приложения Vault из репозитория, затем запустить синхронизацию (Sync) приложения Vault. Проверить в интерфейсе управления Vault, что настройки применились. Применение настроек может занять несколько минут.
4. Актуализировать пути секретов `vault:ИМЯ_ОКРУЖЕНИЯ/data/...` во всех манифестах секретов, например:

```
apiVersion: v1
kind: Secret
metadata:
  name: minio-root-secret
  namespace: minio
  annotations:
    vault.security.banzaicloud.io/vault-addr: "https://vault.vault:8200"
    vault.security.banzaicloud.io/vault-tls-secret: "vault-tls"
    vault.security.banzaicloud.io/vault-role: "secrets_webhook"
type: Opaque
data:
  # Изменить "launch-demo" на актуальное ИМЯ_ОКРУЖЕНИЯ:
  # config.env: vault:launch-demo/data/minio/minio-root-
secret#config.env
  config.env:
dmF1bHQ6bGF1bmNoLWRlbW8vZGF0YS9taW5pby9taW5pby1yb290LXNlY3JldCNjb25maWcu
ZW52
```

5. Отправить на сервер Git изменения в репозитории GitOps.

Создание секретов в Vault

1. В локальной копии репозитория GitOps перейти в каталог с плейбуками Ansible для начальной настройки Launch – `ansible/initial-setup`:

   ```
   cd ansible/initial-setup
   ```

2. Отредактировать файл дампа реалма Keycloak `realm.json`, и актуализировать в нём DNS-имена серверов Launch. Это также может быть сделано после установки приложений Launch через интерфейс управления Keycloak.

3. В файле `clusters/usetech/launch-demo/group_vars/vault.yaml` задать секреты, такие как имя пользователя и пароль Docker Registry, а также данные для подключения к прочим ресурсам.

4. Запустить плейбук Ansible для создания секретов в Vault. После создания секреты можно посмотреть и изменить в интерфейсе управления Vault.

   ```
   ansible-playbook --extra-vars "@../../clusters/usetech/launch-
   demo/group_vars/vault.yaml" create-vault-secrets.yaml
   ```

5. Проверить, что секреты действительно созданы в Vault и, при необходимости, скорректировать содержимое секретов.

Настройка совместного использования GPU

Для того, чтобы несколько pod-ов могли совместно использовать один GPU, следует настроить программное разделение GPU – "time-slicing". Для этого необходимо:

1. Отредактировать файл `apps/usetech/launch-demo/stage-2/gpu-operator/gpu-time-slicing-config.yaml` и задать ограничение на число pod-ов, одновременно имеющих доступ к GPU, в значении параметра `data.sharing.timeSlicing.resources.replicas`.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: gpu-time-slicing-config
  namespace: gpu-operator
data:
  any: |-
    version: v1
    flags:
      migStrategy: none
    sharing:
      timeSlicing:
        # Disallow pods to request more than 1 GPU
        failRequestsGreaterThanOne: true
        resources:
        - name: nvidia.com/gpu
          # Number of pods concurrently using the GPU
          replicas: 4
```

2. Отправить на сервер Git изменения в репозитории GitOps.

Установка приложений Launch

1. В локальной копии репозитория GitOps перейти в каталог, содержащий конфигурацию установки компонентов Launch:

```
cd apps/usetech/launch-demo
```

2. Создать корневое приложение Argo CD, запускающее вторую стадию установки компонентов Launch:

```
kubectl -n argocd apply -f launch-stage-2-app.yaml
```

3. Дождаться успешного завершения второй стадии установки компонентов Launch.
4. Вывести на консоль root-токен Vault, хранящийся в секрете Kubernetes vault-unseal-keys:

```
kubectl -n vault get secrets vault-unseal-keys \
  -o jsonpath={.data.vault-root} |base64 --decode; echo
```

5. Подключиться к интерфейсу управления Vault.
6. Просмотреть имя пользователя и пароль администратора MinIO в секрете Vault `launch-demo/minio/minio-root-secret`.
7. Подключиться к интерфейсу управления MinIO.
8. Загрузить файлы ML-моделей в S3-бакет launch-demo-models:

```
|-- launch-demo-models
|   |-- audio
|   |   `-- ...
|   |-- text
|   |   `-- ...
|   |-- video
|   |    |-- ppe
|   |    |   `-- ...
|   |    `-- yolo
|   |        `-- ...
|   ...
```