1. Оптимизировать виртуалки – добавить диска куда нужно, мб найти как запустить упрощенную винду
2. Установить локальный репозиторий (Вроде есть вариант попроще и посложнее, лучше пока попроще) – Harbor и еще что-то
3. Создать namespace/ ServiceAccount/ Rolebinding
4. «Экспортируй kubeconfig для этого SA → положи на runner-docker в /etc/gitlab-runner/kubeconfig.» ? что это и зачем, хотя бы погуглить. Можно мб пока без этого
5. Погуглить про установку Gitlab через Omnibus – пока можно без этого. Но погуглить, как разворачивается gitlab в кластере на проде
6. Настроить окружение в локальном репозитории
7. Создать приложение на Fastapi в локальном репозитории
8. Создать докерфайл для приложения
9. Разобраться с синтаксисом деплоймента. Прочитать еще раз как правильно деплоить контейнеры
10. Создать ямлы в папке репы : namespace.yaml, deployment.yaml, service.yaml, ingress.yaml, configmap.yaml, secret.yaml . Запускаем
11. Посмотреть kubeconfig и почитать про него
12. Как реализуется ролбэк в кластере и вообще
13. Реализовать мониторинг. Почитать, как это делают на проде

Далее:

1. Починить кластер – его api
2. Раскатать первое приложение с NodePort
3. Посмотреть что такое RBAC для CI
4. Сделать еще одну worker-ноду, подключить ее
5. Посмотреть про все эти балансировщики нагрузок и т.д, выбрать для себя
6. Настроить нужный балансировщик
7. Вернуться к пайплайнам и первому проекту, проанализировать, попробовать повторить самостоятельно пайплайн
8. Разобраться с синтаксисом деплоймента. Прочитать еще раз как правильно деплоить контейнеры
9. Посмотреть kubeconfig и почитать про него
10. Пробежаться по истории настройки кластера, неплохо было бы законспектировать
11. Сделать мониторинг – погуглить, как это делается, в отдельном поде или нет
12. Посмотреть, что такое SonarQube, сделать с ним пайплайн и прокатить
13. Погуглить, что такое артефакты сборки, в яндекс курсе предлагается Nexus, MinIO. Попробовать установить и запустить пайплайн с этим
14. Если все получится с двумя воркерами, почитать про стратегии деплоя. Реализовать:

Blue/Green через два Deployment + переключение Service

Canary через Ingress (если подключишь NGINX Ingress)

1. Сделать приложение с бд, если останется время

* Порядок деплоя:  
  Ты делаешь git push

**GitLab CI:**

Собирает образ через Kaniko

Сканирует через Trivy

Деплоит через kubectl apply -f k8s/

В кластере:

Поды запускаются с limits, probes, PDB

Доступ — только через Ingress + TLS

Сеть — ограничена NetworkPolicy

Состояние — бэкапится ежедневно

* Есть некое состояние кластера, нужно делать его бэкап, вроде связано с etcd
* Гпт советует заменить Flannel на Calico
* Порядок: (Что такое controller-manager, scheduler)

Твой терминал

↓ (HTTPS + аутентификация)

kube-apiserver (на k8s-master)

↓ (запись в etcd)

controller-manager → создаёт ReplicaSet

↓

ReplicaSet → создаёт Pods

↓

scheduler → назначает Pods на worker-ноды

↓

kubelet на worker → запускает контейнеры

* Поменять сборку с Docker на Kaniko (вроде на этапе деплоя, но мб не только)
* Советы:

Поставь HAProxy на отдельную ВМ (или на jumpbox) → балансирует трафик к 6443

Установи Ingress Controller (Nginx или Traefik) в кластере

Настрой MetalLB (для L2-режима) или HAProxy как внешний LB → выдаёт внешний IP для Ingress

Замени NodePort на Ingress с TLS (self-signed или Let's Encrypt в локалке)

Деплой останется тем же (kubectl apply), но трафик будет идти через прод-стек:

User → HAProxy (LB) → Ingress → Service → Pod

* Возможно увеличить количество мастеров. На проде 3 – минимум
* Доработать деплой - Через Ingress + внешний LB (прод-стандарт)
* Записать бы потом хотя бы примерно порядок настройки кластера с нуля – основные моменты
* Добавить балансировщик перед api - Нет —kubectlи CI обращаются напрямую кinfra:6443. Подумать-погуглить, как это сделать
* Добавить балансировщик перед приложением - Нет — используется**NodePort**илиkubectl port-forward
* Развернуть kube-спреем
* Советую 3 ноды на мастер
* Добавить разделение на среды? и что вообще понимается под средой
* Ресурс Service. Тип NodePort
* Service типа ClusterIP и ExternalName
* Запуск pod с инструментарием(ping, dig, telnet …)
* Port forwarding
* Service типа Load balancer. Ресурс Ingress. Пускаем трафик из интернета
* Cert-manager. Автоматическое получение SSL сертификата
* Хранение данных: ConfigMap, Secrets
* Volumes: персистентность данных(PV, PVC)
* Config Volumes: подключаем файлы ConfigMap
* Настройка приложения через ENV и Secrets
* StatefulSets
* DaemonSet
* Jobs и запуск по расписанию Cronjobs
* NodeAffinity, NodeSelector
* PodDisruptionBudget, PriorityClass
* Масштабирование pods: HorizontalPodAutoscaler
* Масштабирование nodes: Autoscaler
* LimitRange/ResourceQuota
* Service Accounts & RBAC