Sincroniza las apps de k

Te aseguras de que el estado de tus apps es igual que el de tu repo

Declarativo and versionado

git config --global --edit

git config --global core.editor “vim” ( o code o nano...)

Ssh-keygen –t ed25519 –C <nameInventado> para generar ssh  
eval “$(ssh-agent –s)” para saber el pid y activar el agente que almacena el ssh

Ssh-add virbuliña/.ssh/id\_ed25519 para añadir identidad

Añadimos ssh a gitlab

Instalamos wsl, y después docker y kubectl y por ultimo argo

[Course: Mastering GitOps with Argo CD | Udemy Business](https://indra.udemy.com/course/mastering-gitops-with-argo-cd/learn/lecture/37160390#overview)

[Course: Mastering GitOps with Argo CD | Udemy Business](https://indra.udemy.com/course/mastering-gitops-with-argo-cd/learn/lecture/37160394#overview)

TRUNK-BASE: cada app en directorios separados. Feature a main

GITFLOW: develop cada app and feature cada app. Los cambios se mergean a develop

Ej en develop mergeamos y nos traemos el cambio que hicimos en yml en la feature (git merge —noff <nameoffeaturebranch>. Cuando esta listo para release se crea una rama release con el nombre release/featureversion

Y mergeamos release/feature into develop y la release en main

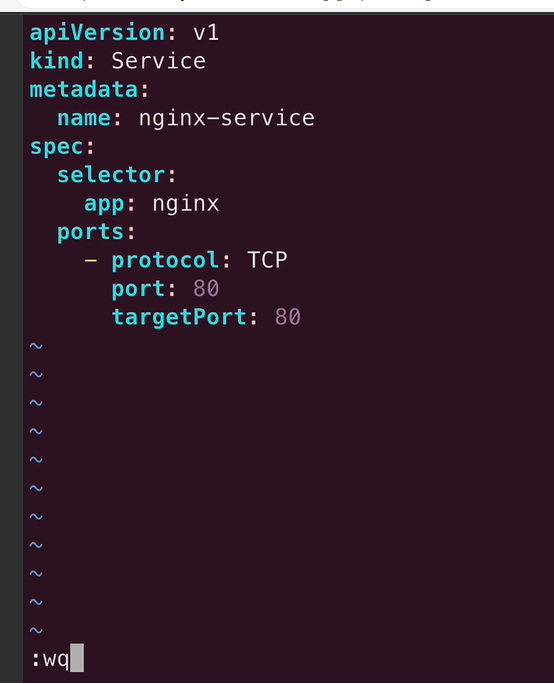
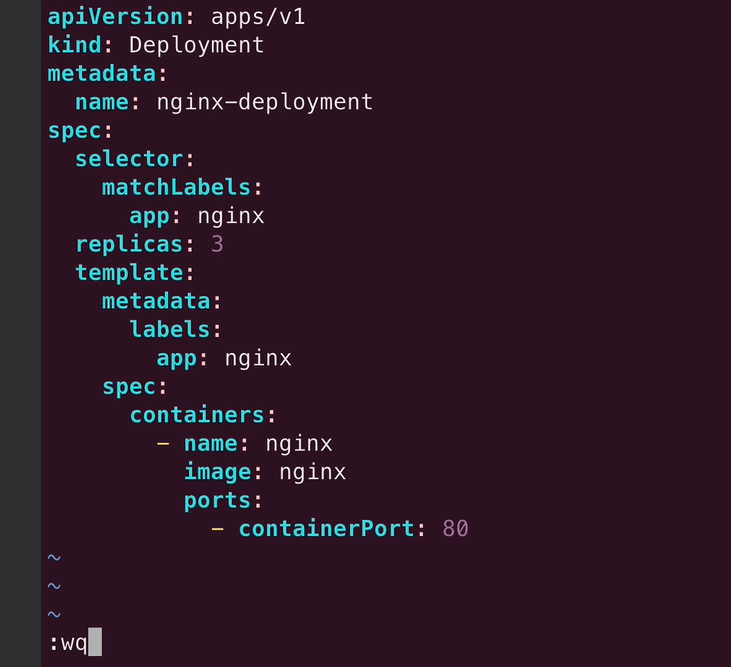
GITHUB FLOW: main para desarrollo y feature para pull req

MANIFEST:

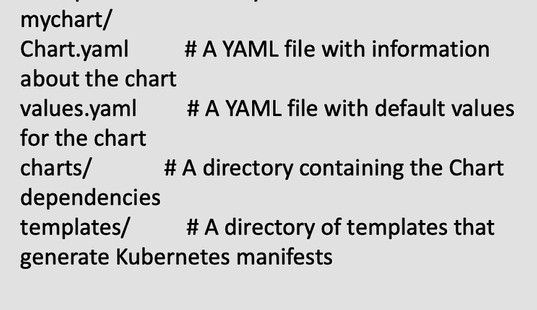
Vim deployment.yml

Creamos el servicio y el deployment y los aplicamos

Get pods and svc



HELM es un empaquetador for k. Empaquetar apps en una simple unidad. Puedes reusarlos. Riesgos a la pipeline es mejor testear antes de prd

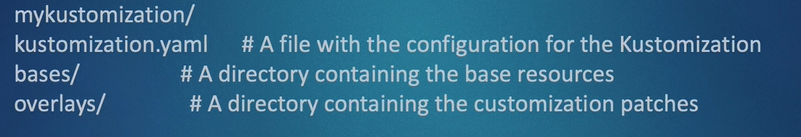


Helm create mychart

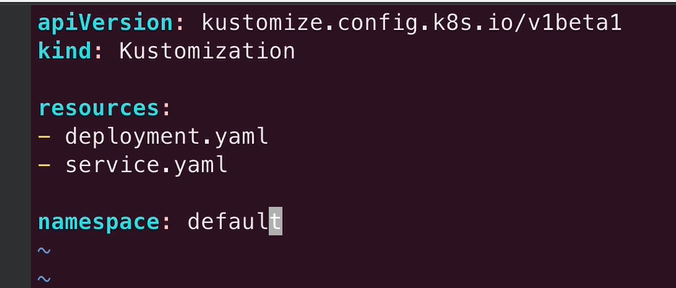
Vim values.yml para abrir los valores(img y tags)

Heml install <nombredelrelease> .

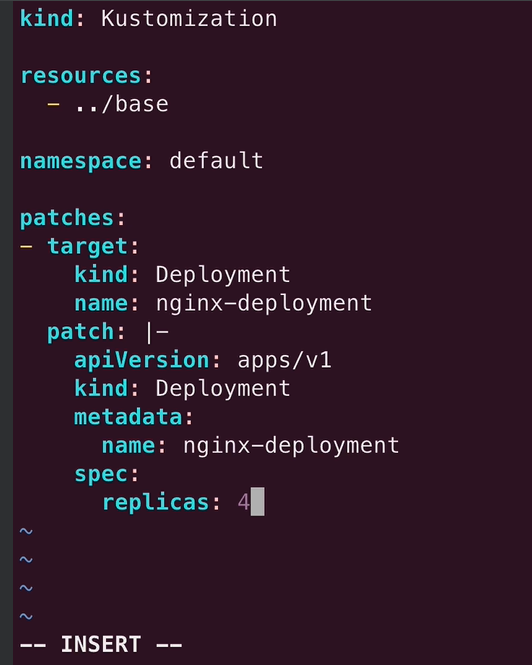
KUSTOMIZE: customiza objetos sin modificar orig yml files.



Una vez situados en la carpeta donde esta el deployment y el servicio (lo copiamos si hemos creado antes chart), creamos las carpetas base (con el kustomization.yml manifiestos disponibles es decir deployment y service)



y overlays(lo que necesitamos cambiar en los manifest parents que viven en base)



Ahora para aplicarlos en vez de –f usamos –k

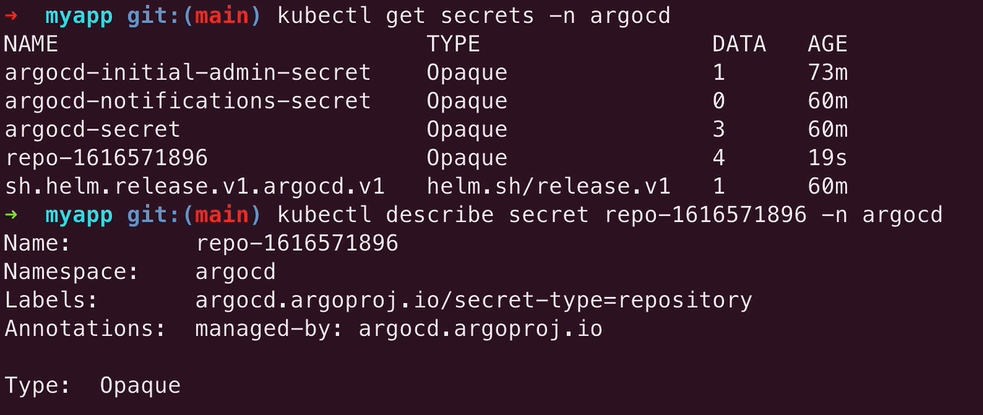
Kubectl apply –k .

Radys: state metadata. History of changes,Status, state...

Prometeus: metrics k cluster and

Git add –A (all)

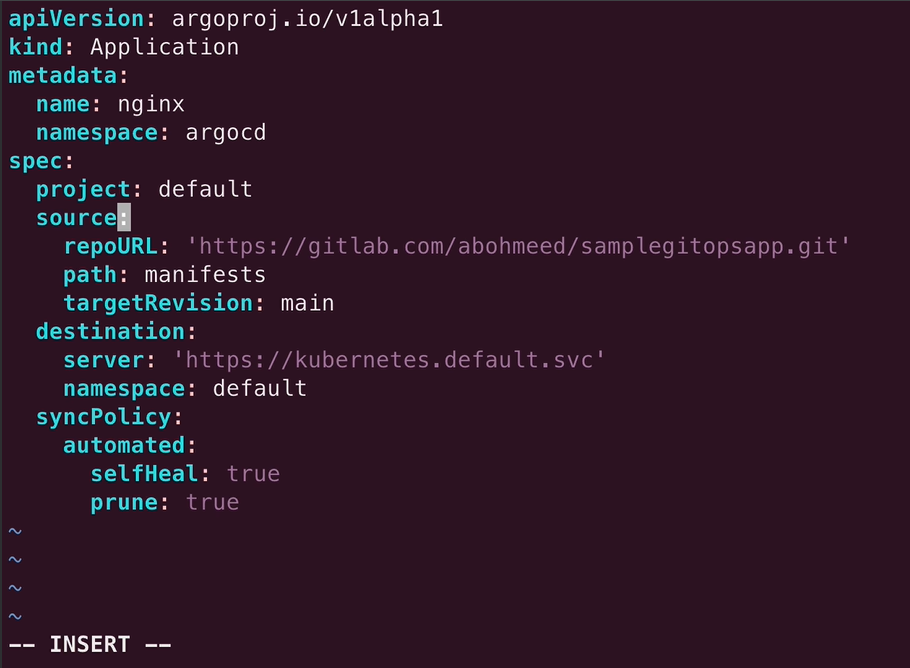
Argocd repo add <urlgit> —username <..> —password <gitlab token>  
kubectl get secrets –n argocd



CREAR APP:

Creamos un subdirectorio dentro del repo (namespace) argocd llamado argo-cd

Y en el fichero nginx.yml:

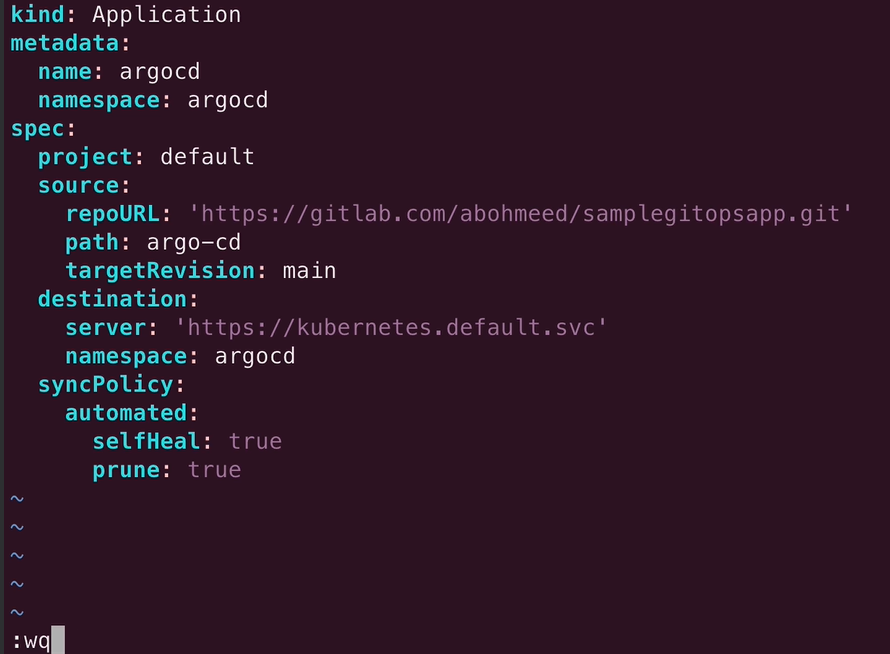
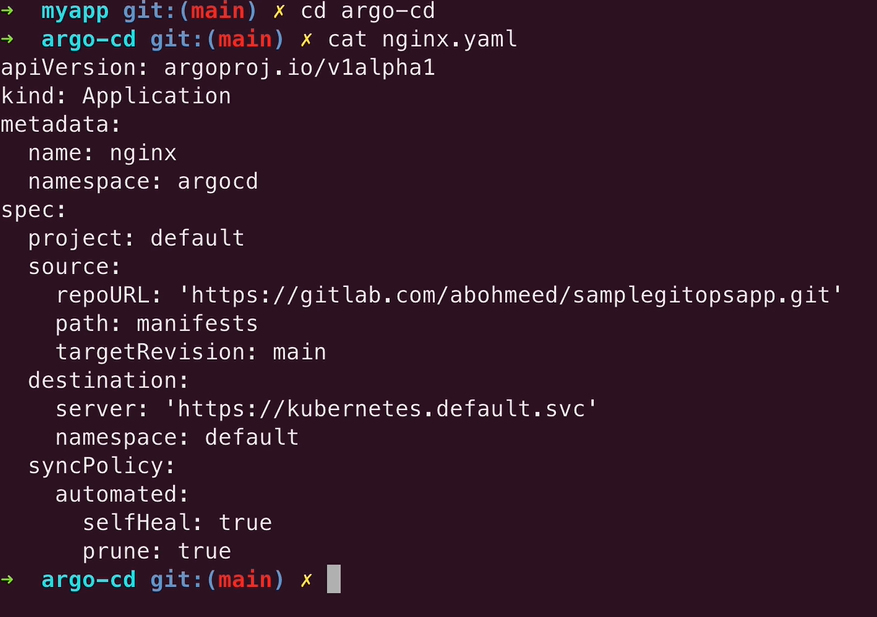


Cuando se hace un cambio y se sube a git, se crea un pull request, se aprueba y se mergea, argo lo detecta.

sync a mano: argocd app sync <nameofapp>

Argocd app sync <app>

Normalmente los manifests para cambiar alguna configuracion de los yml originales no se suben al repo

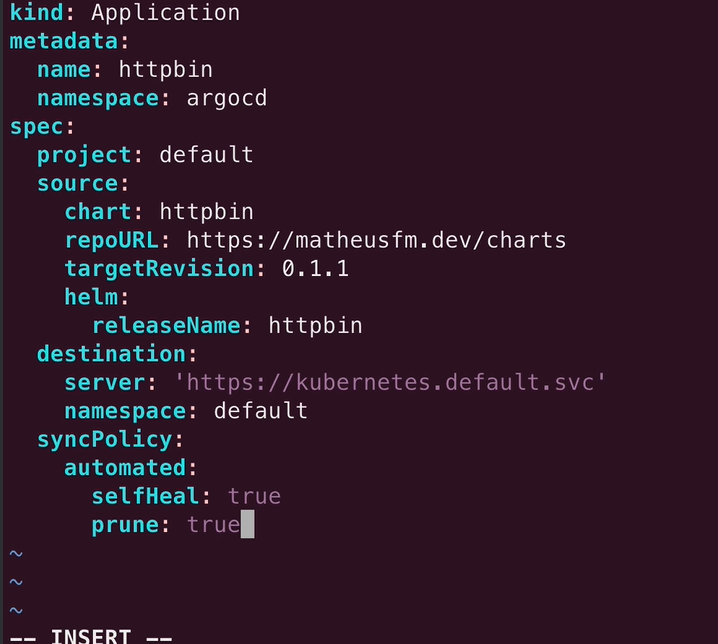


Diferente path y namespace

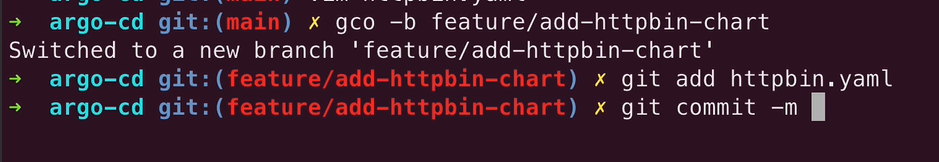
Aplicamos

Y aparece en argo otra app

CREAR UN CHART:



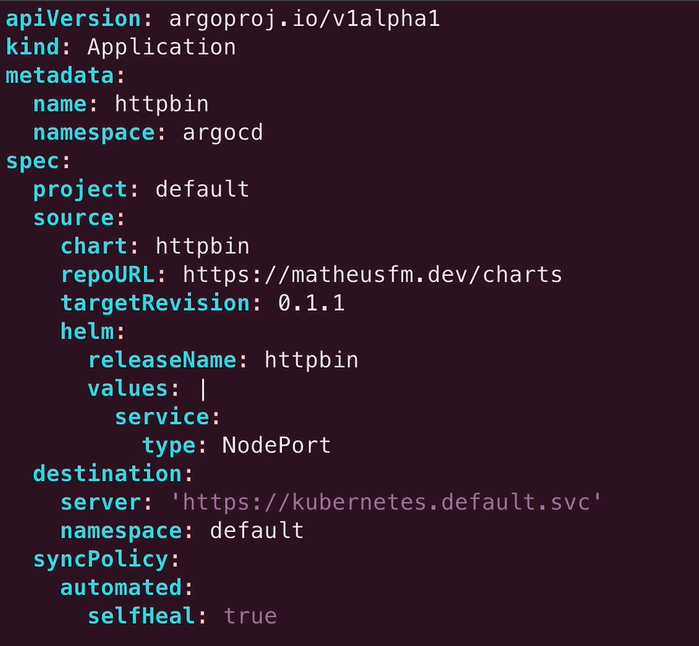
En vez de aplicar los cambios lo subimos al repo para que argo lo sincronice en k



No hay un helm release

Si queremos cambiar las replicas:

En el main abrimos el httbin.yml



Añadimos el type service

Creamos una rama feature con el nombre del cambio que hemos hecho y añadimos este yml

Vemos que el tipo de servicio de httpbin la app ha cambiado de default a nodeport

Se puede hacer de otra forma:



Kubectl get nodes –o wide: para ver el IP del node

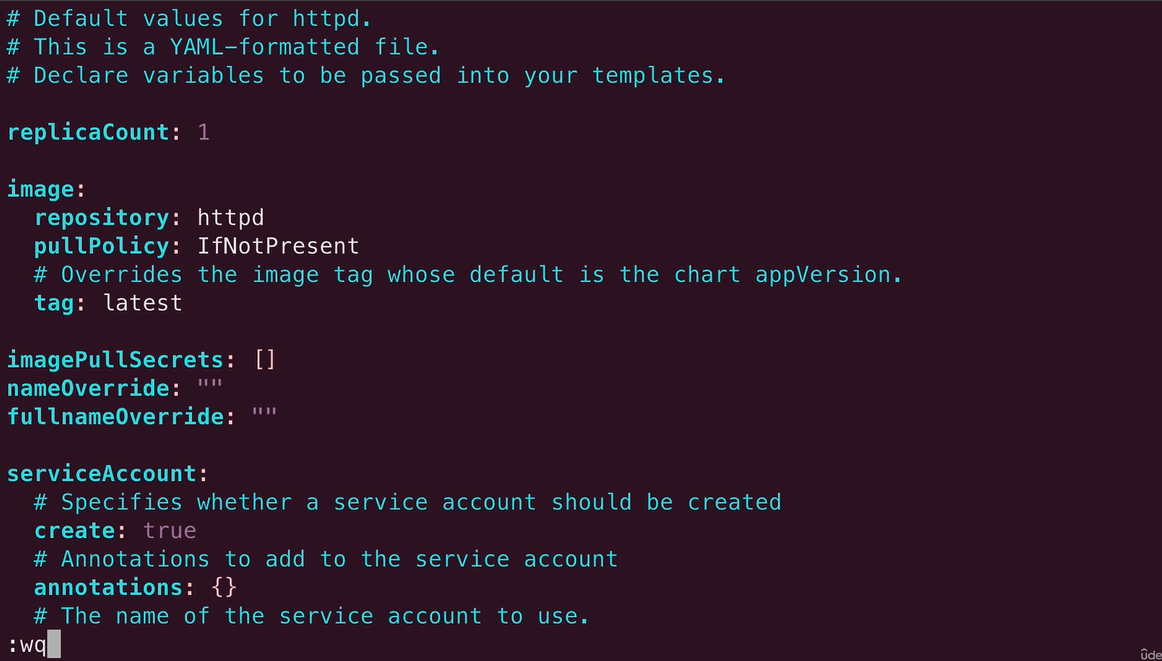


CREAR UN CHART LOCAL:

Helm create httpd



Ponemos image name httpd y tag latest



Y abrimos un Chart.yml

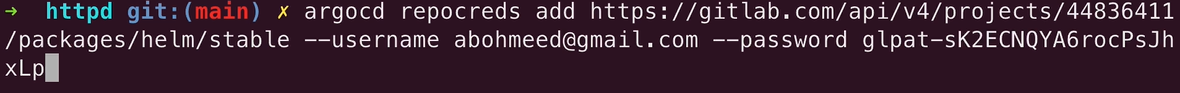
Heml package .

Deploy an app chart de un repo privado:

Empaqueta todos los ficheros del chart

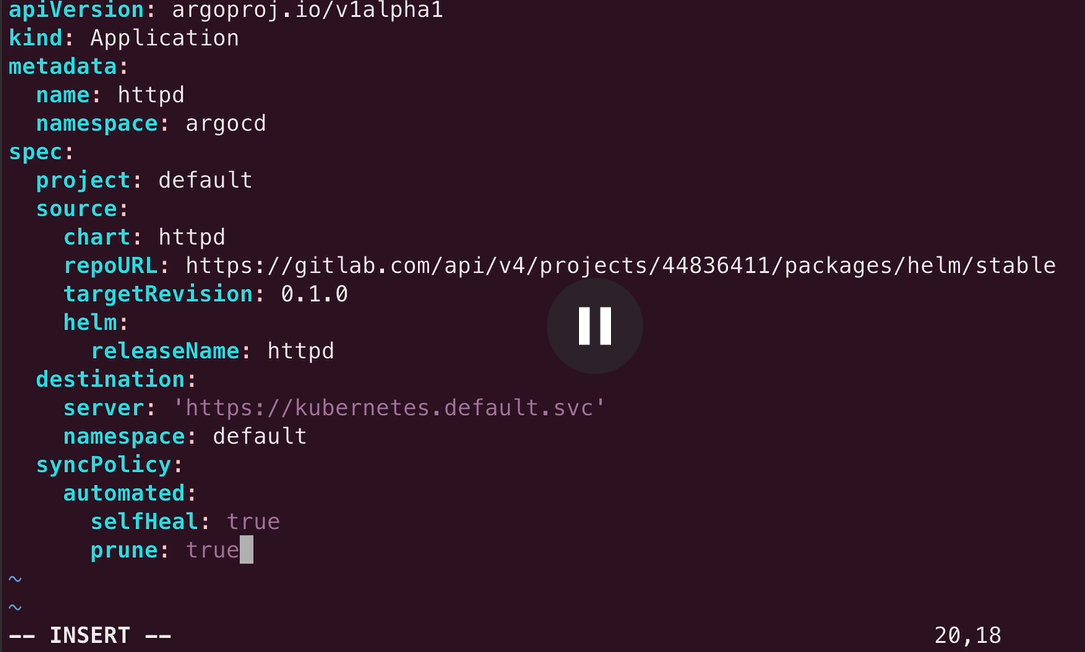
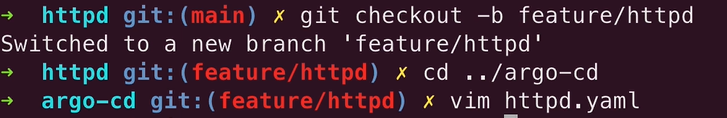


Ahora al remoto:



Personal access token

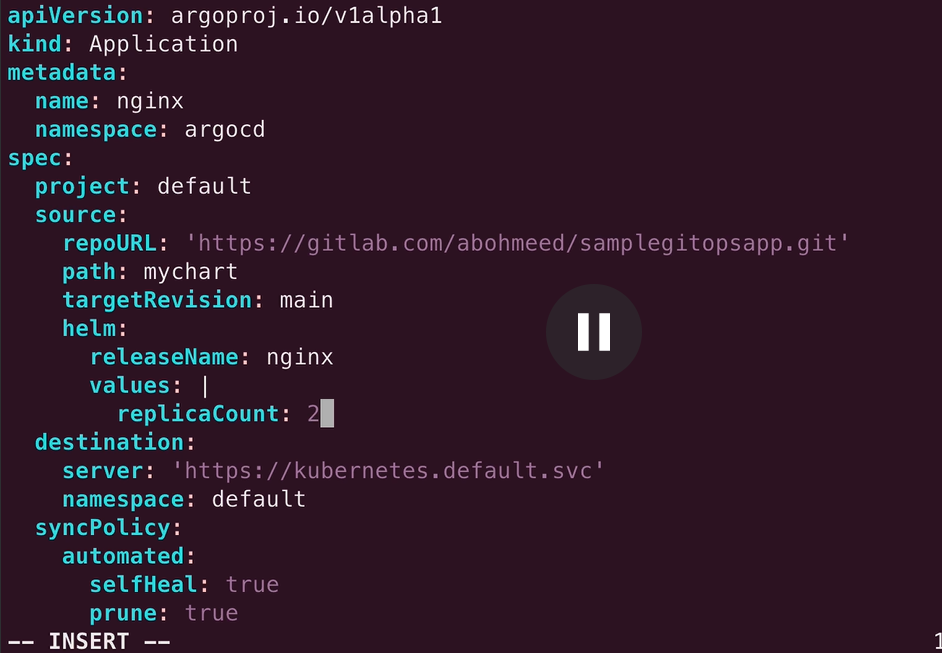
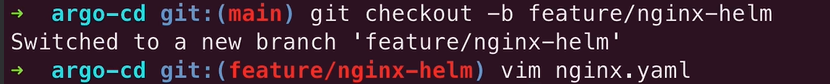
Creamos el httpd.yml



Cambia la url del repo

Push a gitlab

Otro ejemplo creamos una rama feature/



He cambiado el path que en vez de manifests es mychart, añadido el heml que es nginx y el numero de replicas

Push

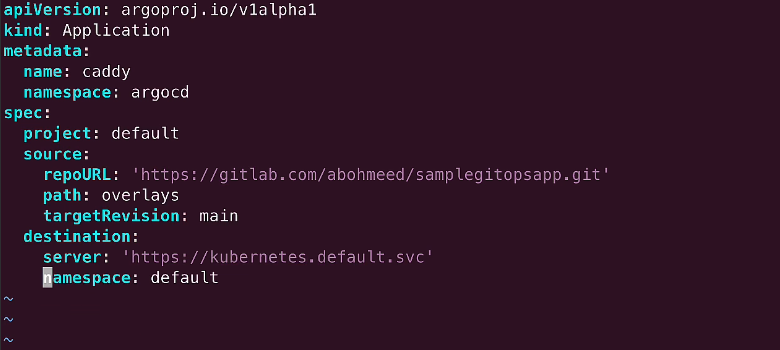
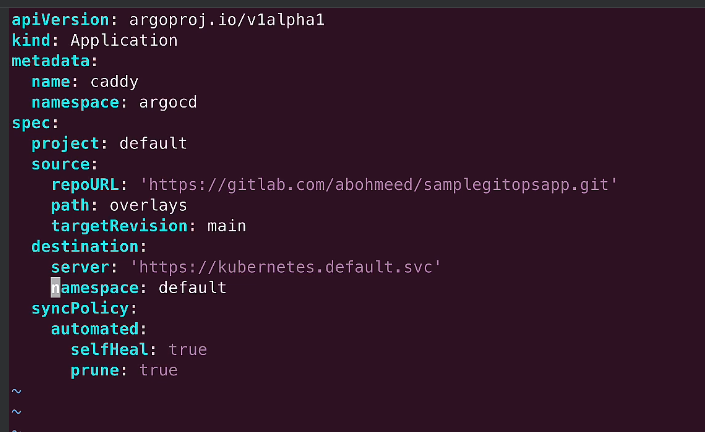
Vemos que nginx app ha cambiado el path de manifests to mychart

Get pods dice los pods(apps) corriendo

ME SALTO LO DE DEPLOY APP TO ARGO USING KUSTOMIZE, MANAGING SECRETS

Disable the sync of an app:

Pasamos de esto



Push

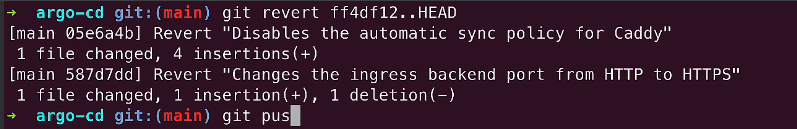
Y ahora ya hacemos rollback

Para loguearse en argo:



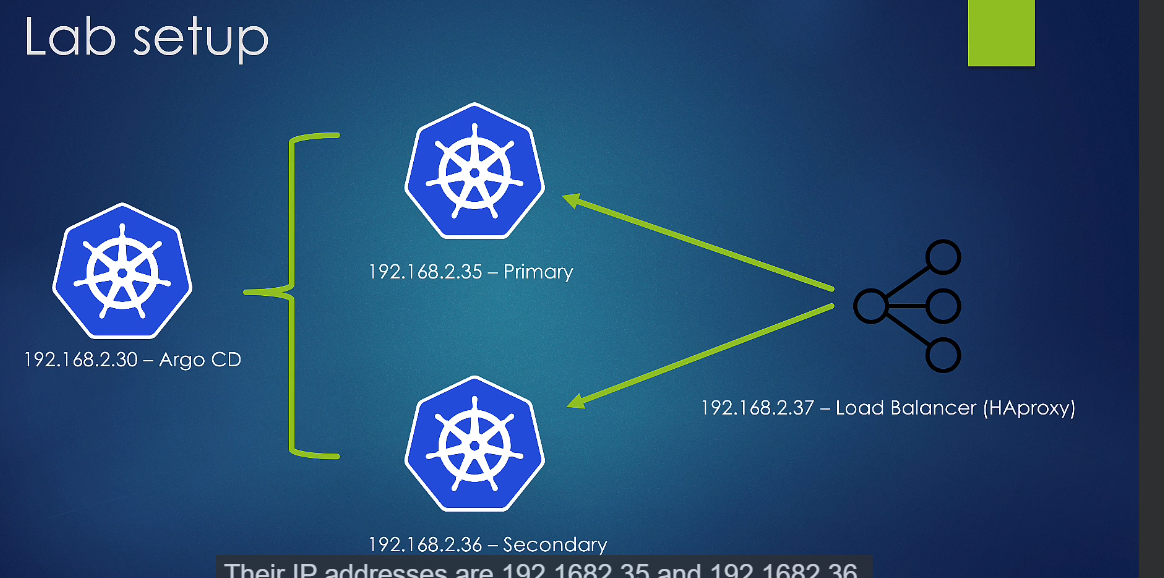
Argocd app rollback <idcommit>

Ponemos la config a sincronizacion automatica con el repo de nuevo para que no se aplique el https: Git revert

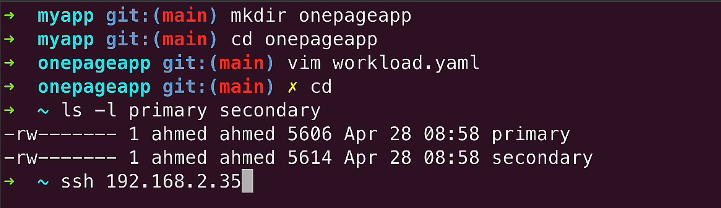


Argocd app history <nameofapp>

Git log —oneline



Configmap, deployment, service and ingress



Application set:

