Workshop ci/cd pipeline in kubernetes

Today we are going to set up a CI/CD pipeline in kubernetes based on ArgoCD en Tekton. We will build a Spring Boot demo application in this pipeline and deploy it to the staging and production environment in the kubernetes cluster!

Prerequisites

- Github account
- Docker client
    - will enable us to run docker container
    - **On Windows / Mac make sure you assign at least 4GB of Ram to the daemon!!!!**
- Git client
- k3d client https://k3d.io/#installation
    - will enable us to run a kubernetes cluster based on Rancher k3s as docker containers
- kubectl client https://kubernetes.io/docs/tasks/tools/
    - will enable us to communicate with the kubernetes cluster
    - installing bash autocomplete will make things a lot easier!
    - also, many people configure 'k' as alias for 'kubectl'
- tekton client https://github.com/tektoncd/cli#installing-tkn
    - will enable us to manage tekton pipelines in the kubernetes cluster
- Windows users(!!): the command line instructions assume a Linux shell. On Windows please make use of a Linux based shell like Git Bash or the Windows Subsystem for Linux. If you want to use Powershell you will have to look up the appropriate commands for that.

Walkthrough

1. Install the docker daemon for your OS. When finished, check that it is installed correctly. You should be able to show the version and download a docker image:

```
ricohome@Enricos-MacBook-Work ~ % docker version
Client:
 Cloud integration: 1.0.17
 Version:           20.10.7
 API version:       1.41
 Go version:        go1.16.4
 Git commit:        f0df350
 Built:             Wed Jun  2 11:56:22 2021
 OS/Arch:           darwin/amd64
 Context:           default
 Experimental:      true

Server: Docker Engine - Community
```

```
 Engine:
  Version:          20.10.7
  API version:      1.41 (minimum version 1.12)
  Go version:       go1.13.15
  Git commit:       b0f5bc3
  Built:            Wed Jun  2 11:54:58 2021
  OS/Arch:          linux/amd64
  Experimental:     false
 containerd:
  Version:          1.4.6
  GitCommit:        d71fcd7d8303cbf684402823e425e9dd2e99285d
 runc:
  Version:          1.0.0-rc95
  GitCommit:        b9ee9c6314599f1b4a7f497e1f1f856fe433d3b7
 docker-init:
  Version:          0.19.0
  GitCommit:        de40ad0
```

```
ricohome@Enricos-MacBook-Work ~ % docker pull nginx
Using default tag: latest
latest: Pulling from library/nginx
33847f680f63: Pull complete
dbb907d5159d: Pull complete
8a268f30c42a: Pull complete
b10cf527a02d: Pull complete
c90b090c213b: Pull complete
1f41b2f2bf94: Pull complete
Digest: sha256:8f335768880da6baf72b70c701002b45f4932acae8d574dedfddaf967fc3ac90
Status: Downloaded newer image for nginx:latest
docker.io/library/nginx:latest
```

2. Install k3d client. Check it is installed correctly

```
ricohome@Enricos-MacBook-Work ~ % k3d --version
k3d version v4.4.4
k3s version latest (default)
```

3. Install kubectl client. Check it is installed correctly. It is ok if it is showing a connection error, this is because we haven't configured it yet for a kubernetes cluster.

```
ricohome@Enricos-MacBook-Work ~ % kubectl version
Client Version: version.Info{Major:"1", Minor:"19", GitVersion:"v1.19.4",
GitCommit:"d360454c9bcd1634cf4cc52d1867af5491dc9c5f", GitTreeState:"clean", BuildDate:"2020-11-12T01:09:16Z",
GoVersion:"go1.15.4", Compiler:"gc", Platform:"darwin/amd64"}
The connection to the server localhost:8080 was refused - did you specify the right host or port?
```

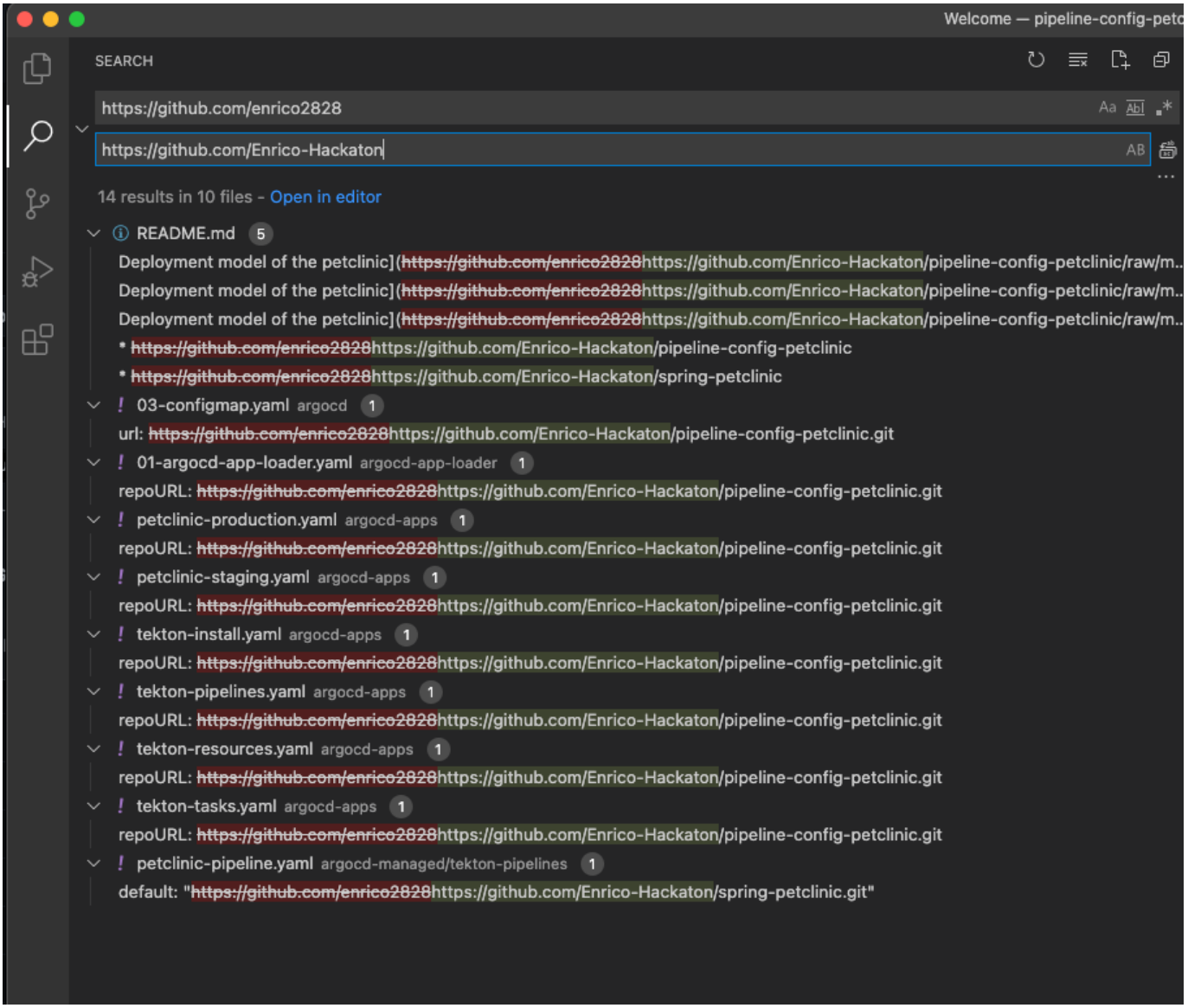
4. Install the tekton client. Check it is installed correctly.

```
ricohome@Enricos-MacBook-Work ~ % tkn
CLI for tekton pipelines

Usage:
tkn [flags]
tkn [command]

Available Commands:
  bundle              Manage Tekton Bundles
```

```
  clustertask          Manage ClusterTasks
  clustertriggerbinding Manage ClusterTriggerBindings
  condition            Manage Conditions
  eventlistener        Manage EventListeners
  hub                  Interact with tekton hub
  pipeline             Manage pipelines
  pipelinerun          Manage PipelineRuns
  resource             Manage pipeline resources
  task                 Manage Tasks
  taskrun              Manage TaskRuns
  triggerbinding       Manage TriggerBindings
  triggertemplate      Manage TriggerTemplates

Other Commands:
  completion           Prints shell completion scripts
  version              Prints version information

Flags:
  -h, --help   help for tkn

Use "tkn [command] --help" for more information about a command.
```

5. Fork the github repos that you need for this exercise and clone them to your desktop
    - https://github.com/enrico2828/spring-petclinic
    - https://github.com/enrico2828/pipeline-config-petclinic

6. In the repo "pipeline-config-petclinic" replace the following urls with the urls to your repo forks, commit and push back to github:
    - https://github.com/enrico2828/spring-petclinic
    - https://github.com/enrico2828/pipeline-config-petclinic
    - git@github.com:enrico2828/pipeline-config-petclinic.git



7. Follow the instructions for "K3D initialization" in the Readme of the pipeline-config-petclinic repo. Then return to this document. Check that the cluster is ready:

```
ricohome@Enricos-MacBook-Work pipeline-config-petclinic % kubectl cluster-info
Kubernetes master is running at https://0.0.0.0:55232
CoreDNS is running at https://0.0.0.0:55232/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy
Metrics-server is running at https://0.0.0.0:55232/api/v1/namespaces/kube-system/services/https:metrics-server:/proxy
```

8. Some kubernetes basics:
    - How many nodes are there in our cluster?
        - kubectl get nodes
    - What namespaces are there?
        - kubect get ns
    - Switch namespace
        - kubectl config set-context --current --namespace kube-system

- Show all pods in namespace
  - kubectl get po
- Switch back to default namespace
  - kubectl config set-context --current --namespace default
- Start a pod with nginx
  - kubectl run mytestpod --image=nginx
- Show logs of a pod
  - kubectl logs mytestpod
- Show most important stats of pod
  - kubectl describe po mytestpod
- Kill pod
  - kubectl delete po mytestpod

9. Let's deploy ArgoCD now.

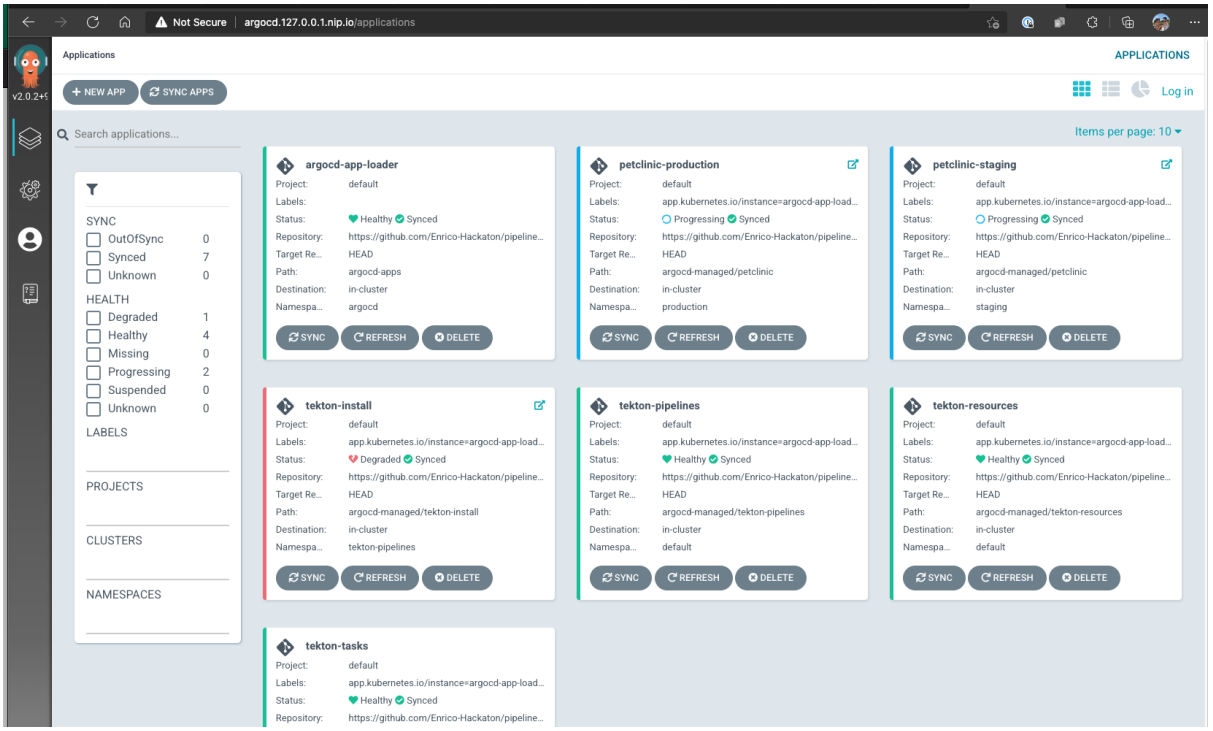
```
ricohome@Enricos-MacBook-Work pipeline-config-petclinic % kubectl create namespace argocd && kubectl apply -n argocd -f argocd/
```

> Argo CD is now being installed to the cluster. It should be available under http://argocd.127.0.0.1.nip.io. You can check the deployment status. All pods must have status 1/1.

```
ricohome@Enricos-MacBook-Work pipeline-config-petclinic % kubectl  get po -n argocd
NAME                                  READY   STATUS    RESTARTS   AGE
argocd-redis-759b6bc7f4-9j7t7         1/1     Running   0          5m23s
argocd-dex-server-9dc558f5-58rdt      1/1     Running   0          5m23s
argocd-repo-server-5fbf484547-qz5cz   1/1     Running   0          5m23s
argocd-application-controller-0       1/1     Running   0          5m23s
argocd-server-588d8f485b-k8cpt        1/1     Running   0          5m23s
```

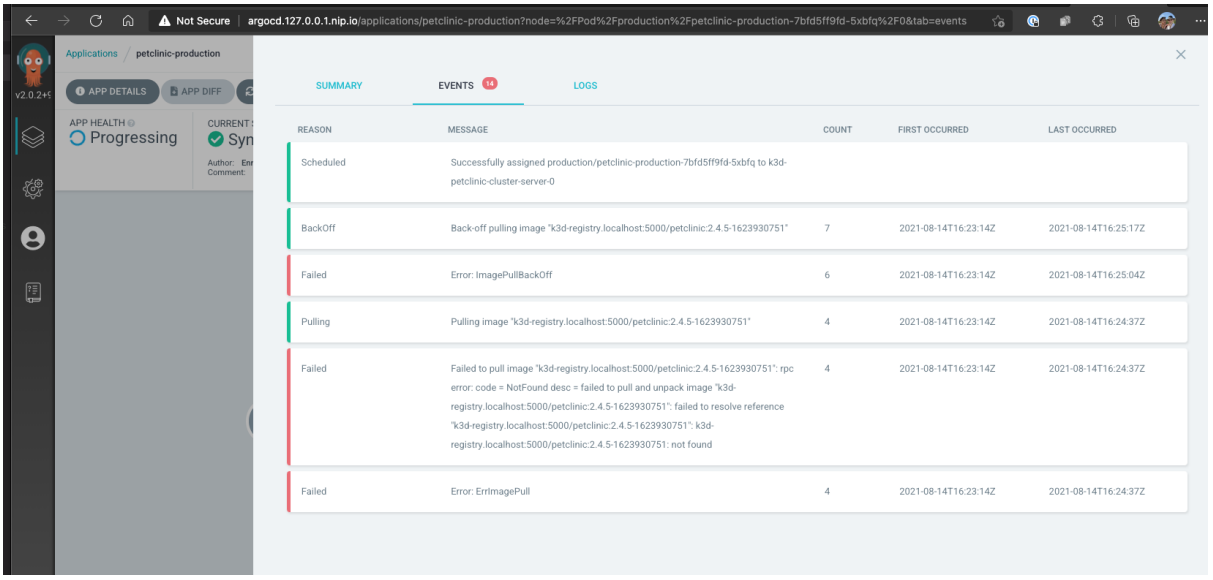
10. The remaining apps will be installed by argocd. All we need to do is add configuration that will point argocd to the repository location that contains the app definition for the remaining apps and argocd will deploy them. The app definitions are in the folder "argocd-apps" in the pipeline-config-petclinic repo and point to the filder "argocd-managed".

```
ricohome@Enricos-MacBook-Work pipeline-config-petclinic % kubectl apply -n argocd -f argocd-app-loader/01-argocd-app-loader.yaml
application.argoproj.io/argocd-app-loader created
```



You can see the kubernetes resources controlled by ArgoCD. ArgoCD will indicate with the sync and health status whether the app has been deployed successfully.

11. The petclinic apps will fail to deploy. Can you find the reason why? You can check with kubectl commands or through the ArgoCD UI.

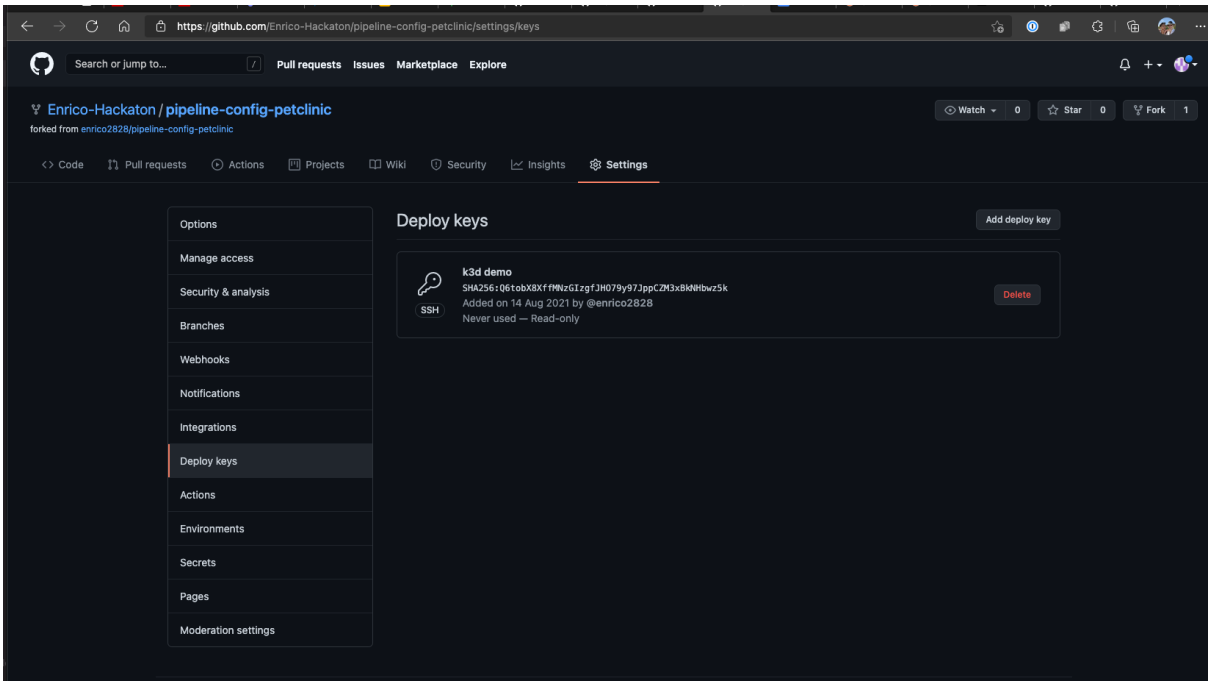


12. Now we need to make sure our pipeline scripts can commit to the github repos. We need to create a ssh key and then add the public key to our github repos, and the private key as a kubernetes secret to the cluster so that the pipeline scripts can use it.

ricohome@Enricos-MacBook-Work pipeline-config-petclinic % ssh-keygen -t rsa -b 4096 -C "test@here.com"

```
Generating public/private rsa key pair.
Enter file in which to save the key (/Users/ricohome/.ssh/id_rsa): ./id_rsa
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in ./id_rsa.
Your public key has been saved in ./id_rsa.pub.
The key fingerprint is:
SHA256:EsgKQeVNB5jOdCm1Orjxj7QFfaOBldCncaDBMy3F6rg test@here.com
The key's randomart image is:
+---[RSA 4096]----+
|o.o+O*+.         |
| ..@*O+o         |
|. +oX=*          |
| ..==. .         |
| o+= o.oS        |
| .+.o +..        |
| ..o o           |
| E. =            |
|   o .           |
+----[SHA256]-----+
```

Add the .pub key to the **pipeline-config-petclinic** github repo. Repository -> Settings -> Deploy Keys -> Add. Make sure you enable write access!



Create a secret file 'github-pipeline-config-petclinic-ssh-key-secret.yaml'. Add the private key base64 encoded to the template (use 'cat <private_key> | base64)

```
apiVersion: v1
kind: Secret
metadata:
 name: github-pipeline-config-petclinic-ssh-key
 namespace: default
data:
 id_rsa:  <insert base64 encoded private key here>
```

Apply to the cluster:

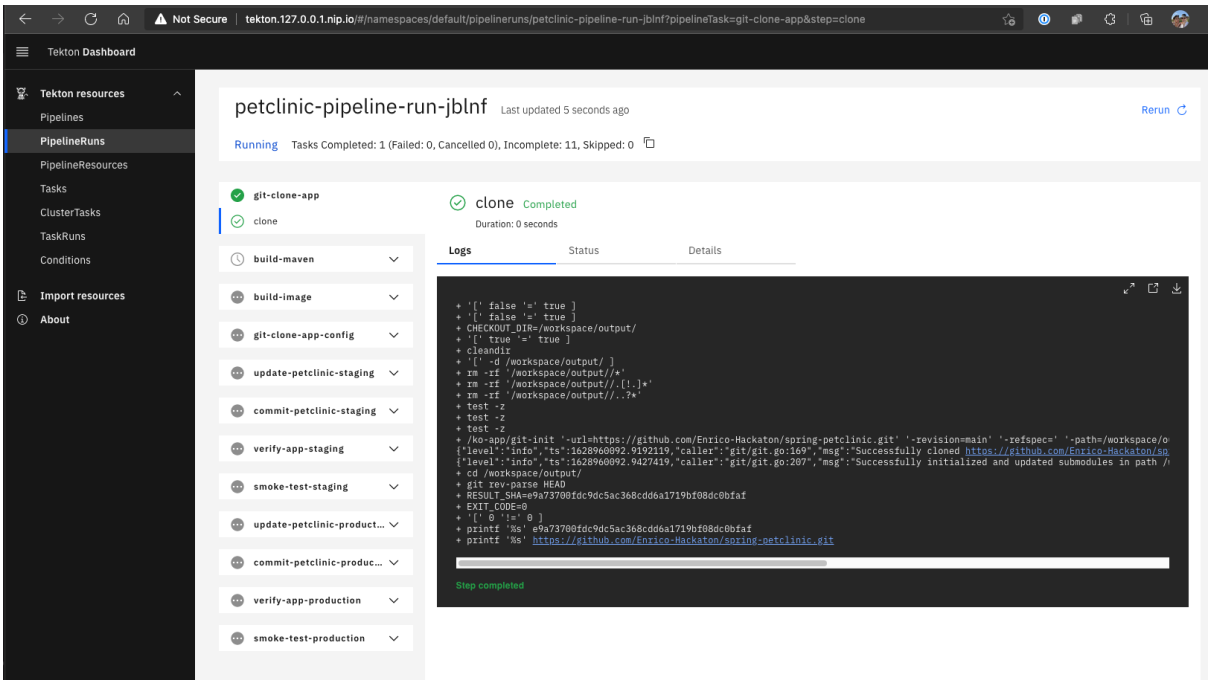
kubectl apply -f github-pipeline-config-petclinic-ssh-key-secret.yaml

13. Now, its time to start the pipeline! The command has been prepared in a bash script

ricohome@Enricos-MacBook-Work pipeline-config-petclinic % ./startbuild-petclinic.sh
PipelineRun started: petclinic-pipeline-run-jblnf

In order to track the PipelineRun progress run:
tkn pipelinerun logs petclinic-pipeline-run-jblnf -f -n default

14. You can also follow the progress in the tekton ui. http://tekton.127.0.0.1.nip.io



When the pipeline is run for the first time, a lot of stuff needs to be downloaded and it might take some time. This is a good opportunity to inspect the pipeline steps a bit closer. You can find the pipeline script in repo 'pipeline-config-petclinic' in folder "tekton-pipelines". The pipeline calls generic pipeline task in folder "tekton-tasks". Everything can also be inspected from within the tekton ui, just have a look around!

Eventually, all pipeline steps should be executed successfully, and if they fail or get stuck, then the fun part begins and we can investigate using the kubectl commands that we introduced earlier!

The app will be accessible on

- http://petclinic-staging.127.0.0.1.nip.io
- http://petclinic-production.127.0.0.1.nip.io

Advanced Exercises!

15. Introduce a bug in the petclinic app that will make it fail, start the pipeline and observe. Is the error detected where you expected it to be detected?

16. Introduce a new feature to the petclinic app that is visible in the ui and can be configured by an environment variable. Set a default value.
    - Start the pipeline and see that the feature is visible in the UI.
    - Now configure the app deployment with the feature toggle to have the non-default function. (add an environment variable to the helm chart of the app and set its value).
        - the change needs to be committed and pushed to the git repository
        - observe argocd and the app in production
17. Think of an additional pipeline step that you want to execute, either with the existing tekton tasks or by adding a new tekton task. Add the step to the pipeline.
    - e.g. add an extra step with the maven dependency check
18. After the ArgoCD App has been updated, it takes some time before ArgoCD detects the change applies it. Can you decrease the time? (increase the frequency that ArgoCD checks for Git changes or trigger a manual sync after the app has been updated)

CLEAN UP

When all done, don't forget to clean up!

Follow the "Erase your traces" of the Readme.md in repo pipeline-config-petclinic.