proficom
we make IT work

Deployment?
Where we code, we don‘t care about Deployments

Introduction to GitOps

DevOps Saxony
Meet beginners, enthusiasts and experts!

15.05.2023

CLOUD NATIVE
COMPUTING FOUNDATION

Who am I?

Yet another **Daniel**

Continuous Testing and Delivery

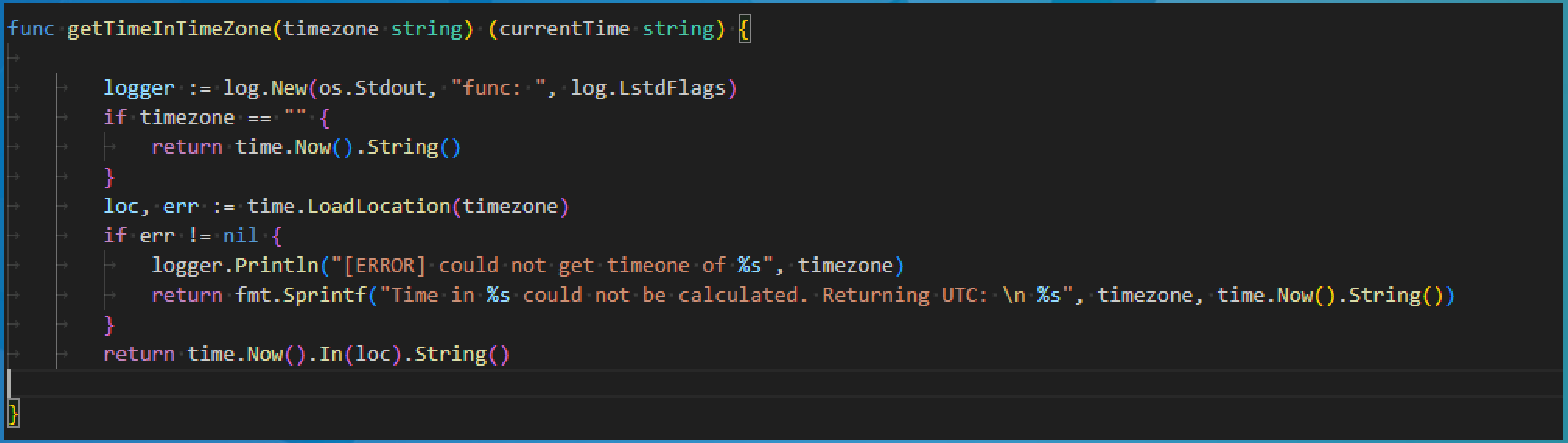
Big fan of DevOps

Always learning new stuff

Let‘s pretent we have an awsome website

https://awsome-site-staging.my-devbox.de/

And we want to deliver a great new feature…



```
func getTimeInTimeZone(timezone string) (currentTime string) {

    logger := log.New(os.Stdout, "func: ", log.LstdFlags)
    if timezone == "" {
        return time.Now().String()
    }
    loc, err := time.LoadLocation(timezone)
    if err != nil {
        logger.Println("[ERROR] could not get timeone of %s", timezone)
        return fmt.Sprintf("Time in %s could not be calculated. Returning UTC: \n %s", timezone, time.Now().String())
    }
    return time.Now().In(loc).String()

}
```

How do we do configuration management?

GitOps…

Whats the fuzz about?

- Golden Source of Truth in SCM
  - Not just the configurations itself
  - Versions have to be pinned and (automatically) updated on new release
- Everyone should be able to just clone the repository and get the same system as me
- Changes in the infrastructure or configuration are just a git-commit away

How to GitOps?

- Used (most of the time) with Kubernetes
  - But other infrastructures are possible

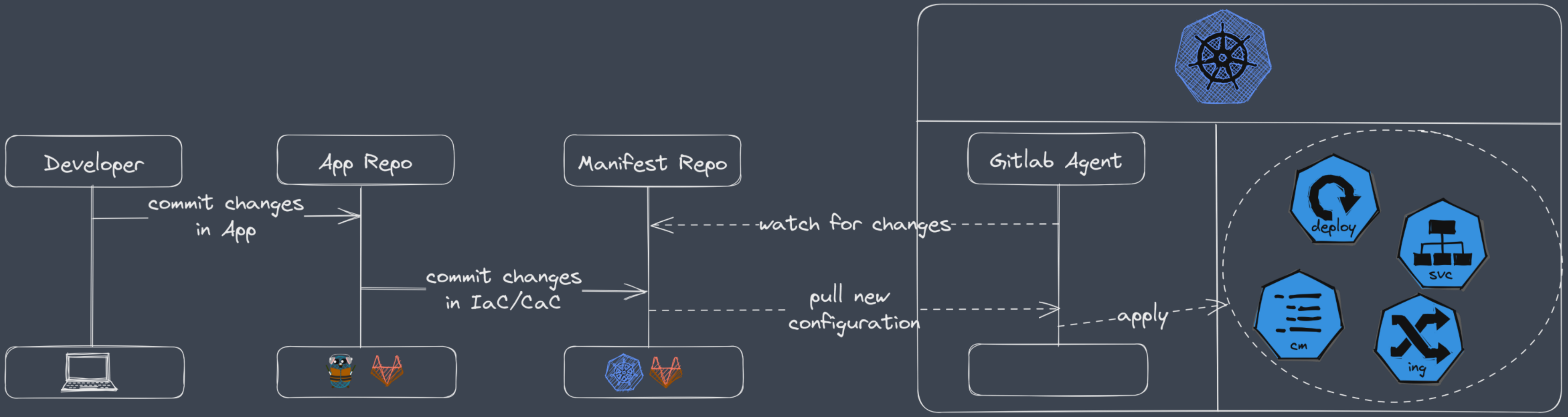
GitOps by Configuration Push

- CI-Server actively publishes the new configuration into the system
  - kubectl, Terraform, Ansible, Helm, …
- ❖ mirrors deployment-activities of developer (CLI-commands, etc.)
- ❖ Fire-and-Forget-rollout
- ❖ CI-Server needs to know system credentials

GitOps by Configuration-Pull

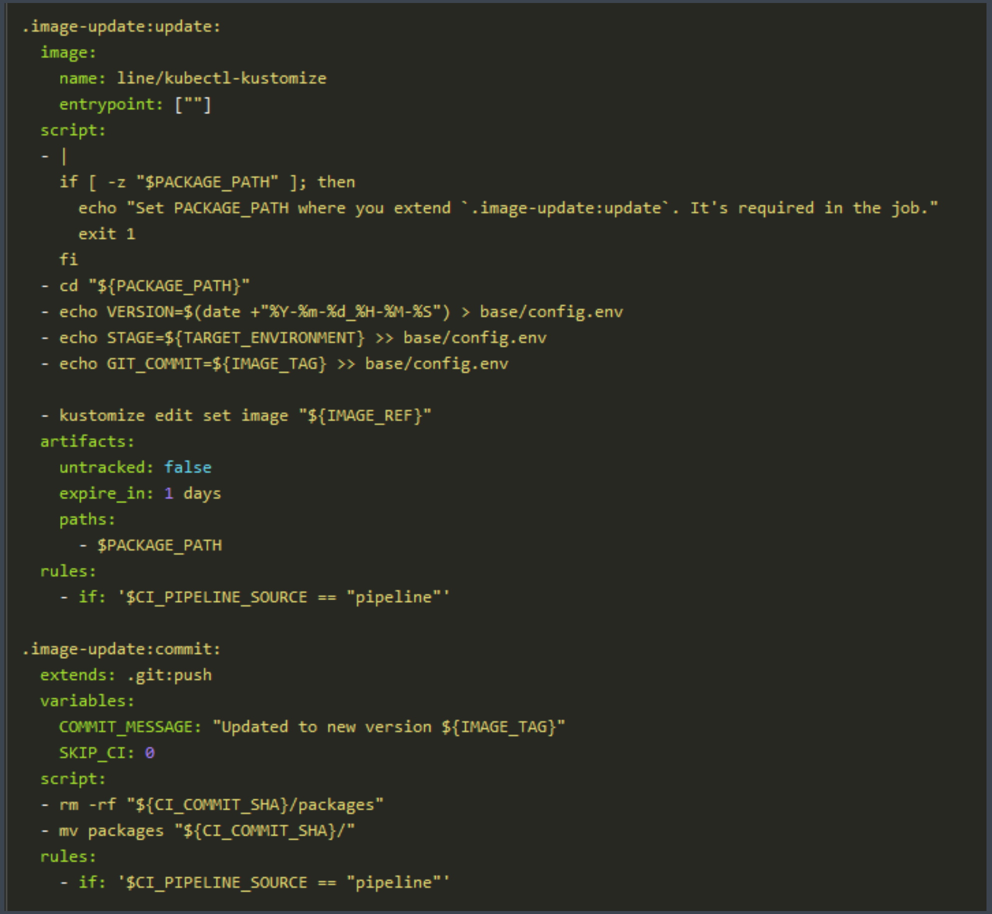
- A Controller inside the system gets new configurations and pulls it into existance
  - argoCD, Flux, GitLab GitOps, …
- ❖ Enforces a declarative, immutable description of the desired system
- ❖ Changes during runtime can be reconciled
- ❖ No potential exposure of system credentials

GitOps with GitLab



Apply changes to the manifests

- Changes are done by a special CI-Job
  - kustomize to apply new configurations to the new packages
    commit: „Updated to new version …“



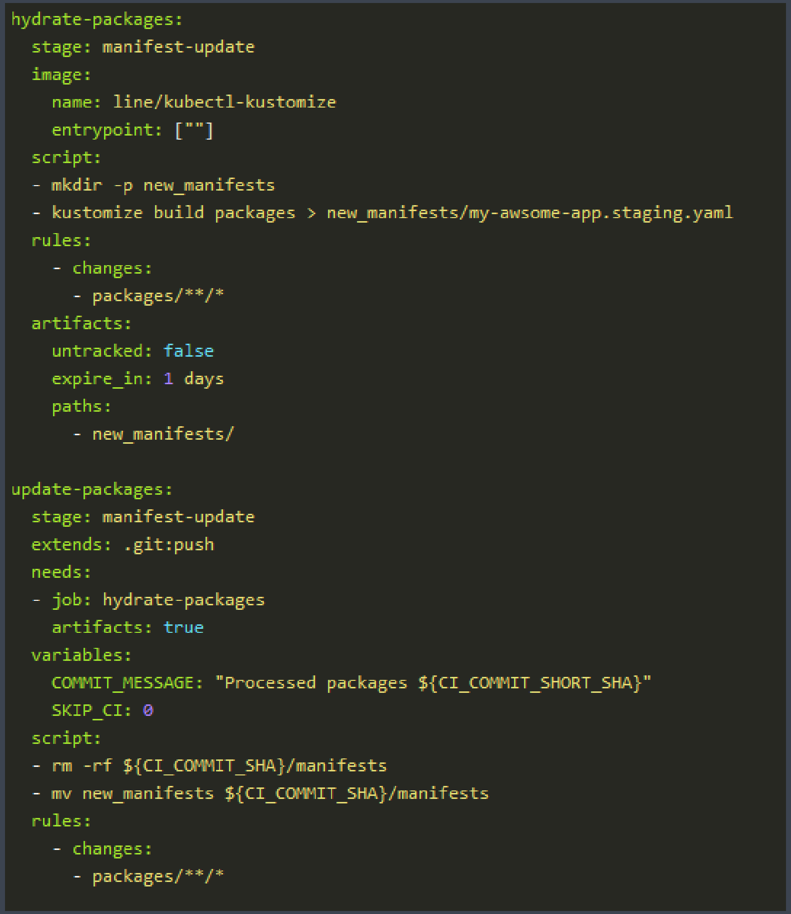
```
.image-update:update:
  image:
    name: line/kubectl-kustomize
    entrypoint: [""]
  script:
  - |
    if [ -z "$PACKAGE_PATH" ]; then
      echo "Set PACKAGE_PATH where you extend `.image-update:update`. It's required in the job."
      exit 1
    fi
  - cd "${PACKAGE_PATH}"
  - echo VERSION=$(date +"%Y-%m-%d_%H-%M-%S") > base/config.env
  - echo STAGE=${TARGET_ENVIRONMENT} >> base/config.env
  - echo GIT_COMMIT=${IMAGE_TAG} >> base/config.env

  - kustomize edit set image "${IMAGE_REF}"
  artifacts:
    untracked: false
    expire_in: 1 days
    paths:
      - $PACKAGE_PATH
  rules:
    - if: '$CI_PIPELINE_SOURCE == "pipeline"'

.image-update:commit:
  extends: .git:push
  variables:
    COMMIT_MESSAGE: "Updated to new version ${IMAGE_TAG}"
    SKIP_CI: 0
  script:
  - rm -rf "${CI_COMMIT_SHA}/packages"
  - mv packages "${CI_COMMIT_SHA}/"
  rules:
    - if: '$CI_PIPELINE_SOURCE == "pipeline"'
```

Apply changes to the manifests

- Changes are done by a special CI-Job
  - kustomize to apply new configurations to the new packages commit: *„Updated to new version …“*
- Changes are picked up by a second CI-Job
  - Running kustomize from the package data
  - Creates the „truth“-manifest with commit *„Processed packages ….“*



```
hydrate-packages:
  stage: manifest-update
  image:
    name: line/kubectl-kustomize
    entrypoint: [""]
  script:
  - mkdir -p new_manifests
  - kustomize build packages > new_manifests/my-awsome-app.staging.yaml
  rules:
    - changes:
      - packages/**/*
  artifacts:
    untracked: false
    expire_in: 1 days
    paths:
      - new_manifests/

update-packages:
  stage: manifest-update
  extends: .git:push
  needs:
  - job: hydrate-packages
    artifacts: true
  variables:
    COMMIT_MESSAGE: "Processed packages ${CI_COMMIT_SHORT_SHA}"
    SKIP_CI: 0
  script:
  - rm -rf ${CI_COMMIT_SHA}/manifests
  - mv new_manifests ${CI_COMMIT_SHA}/manifests
  rules:
    - changes:
      - packages/**/*
```

Let‘s see if our website changed…

https://awsome-site-staging.my-devbox.de/

Conclusions

- GitOps defines the **process** in which the **desired state** of an application is **described** and managed as **IaC**/CaC in a **source code management system**
- Two major ways and ideas to implement GitOps:
  - Push-Pipelines
  - Pull-Pipelines

  with advantages and disadvantages on both sides
- All tools to deploy and manage applications from CLI can be unsed and instrumantalized for both implementations

proficom®
we make IT work

Thank you for your attention

in X ◎ YouTube

Feel free to connect

Daniel Horn

E-Mail: dhorn@proficom.de

Web: https://my-devbox.de/
(you can find my SNS accounts there)

GitHub: https://github.com/addihorn

Other Works: https://gitlab.tools.my-devbox.de/explore