CULTURE CHANGE
USE ZABBIX IN KUBERNETES

HELLO!

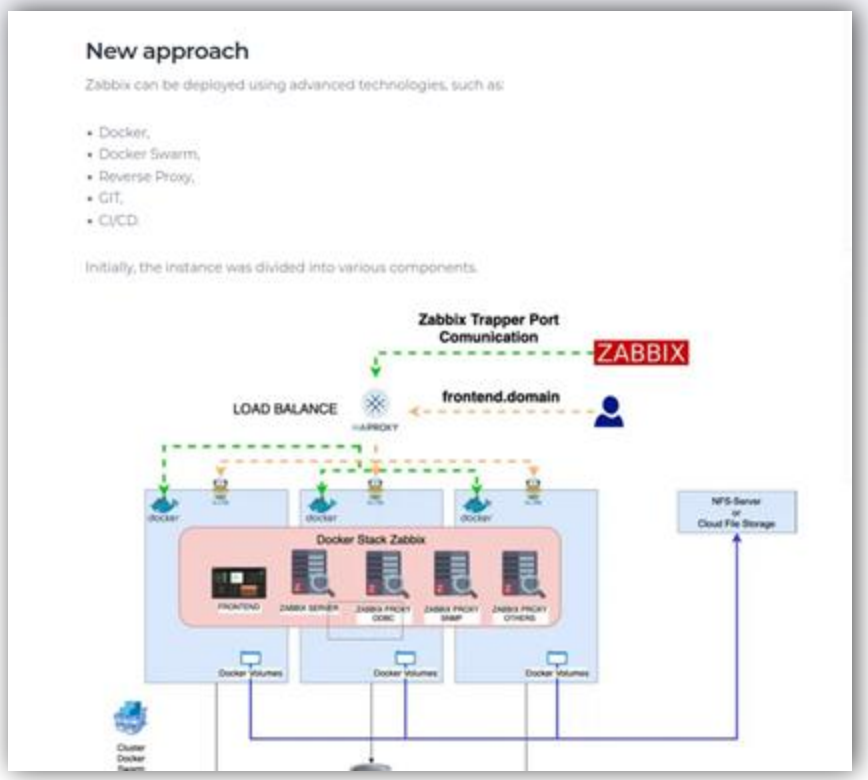
I am Robert Silva

CKA | Zabbix Trainer | Zabbix Expert

DevOps Engineer at JLCP

Does Zabbix work in containers?

I started using Zabbix in containers 2 years ago



https://blog.zabbix.com/scaling-zabbix-with-containers/13155/

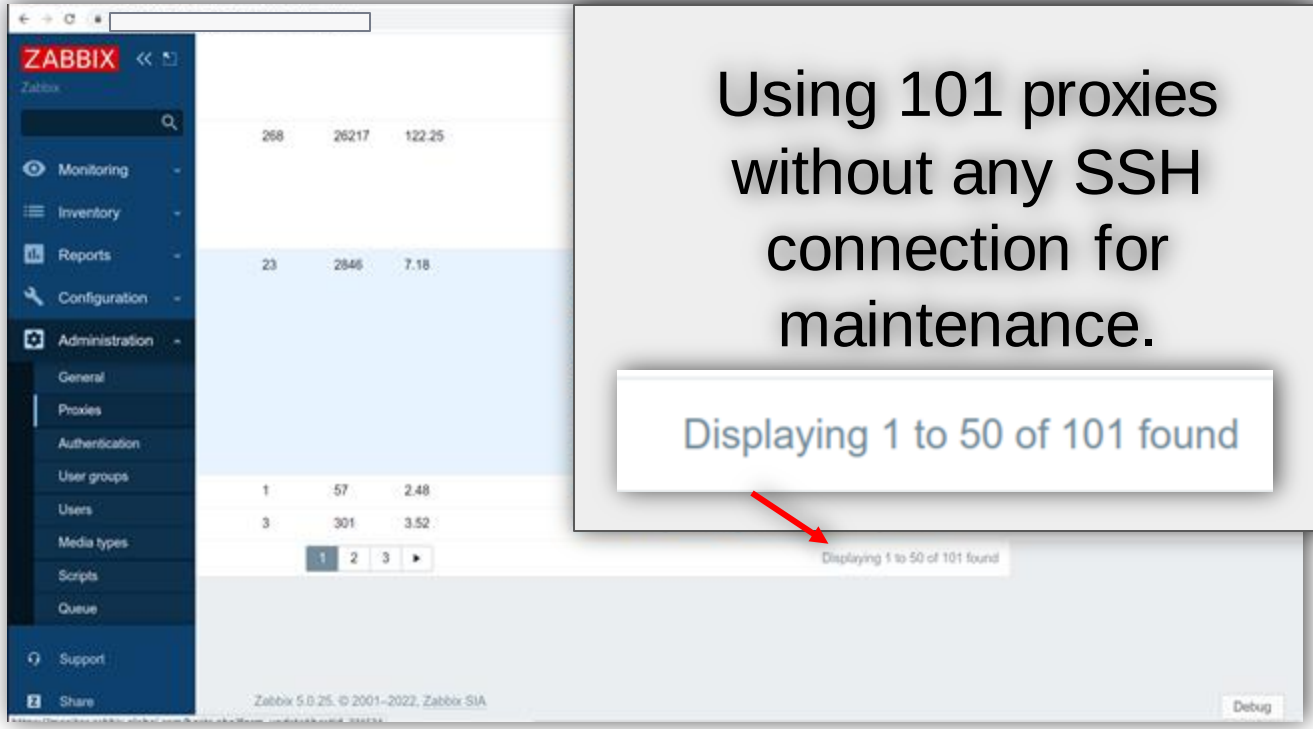
How this environment currently works

How this environment currently works

System information

Parameter	Value	Details
Zabbix server is running	Yes	10.143.224.31:10051
Number of hosts (enabled/disabled)	52494	45426 / 7068
Number of templates	689	
Number of items (enabled/disabled/not supported)	3080584	2579071 / 457540 / 43973
Number of triggers (enabled/disabled [problem/ok])	724126	485715 / 238411 [5095 / 480620]
Number of users (online)	1275	18
Required server performance, new values per second	11802.46	

How this environment currently works



Using 101 proxies without any SSH connection for maintenance.

Displaying 1 to 50 of 101 found

Yes, Zabbix works in containers

But, why do we still have environments in virtual machines?

- Maybe because of **culture** and mainly because of lack of knowledge
  - We can talk about it in other moment.

“

I won't talk about the benefits of Kubernetes.

“

My goal is to answer questions about how Zabbix works on Kubernetes.

Our challenge

Zabbix has hundreds of integrations like:

- AWS;
- Azure;
- Redis;
- MongoDB;
- Kafka;
- APIs;
- Prometheus;
- Kubernetes/Openshift;
- Others.

Software Engineers, Cloud Engineers prefer Prometheus because it is "Cloud Native".

We Zabbix experts must show that Zabbix is prepared to work with modern technologies.

Service Discovery and Low Level Discovery working together is Awesome.

How to start working with kubernetes?

Required skills

- Linux;
- Networks;
- Basic shell script;
- Distributed architecture;
- High Availability Architecture;
- How containers work;
- Container orchestration;
  - Docker Swarm, Kubernetes, Openshift.
- CI/CD process.

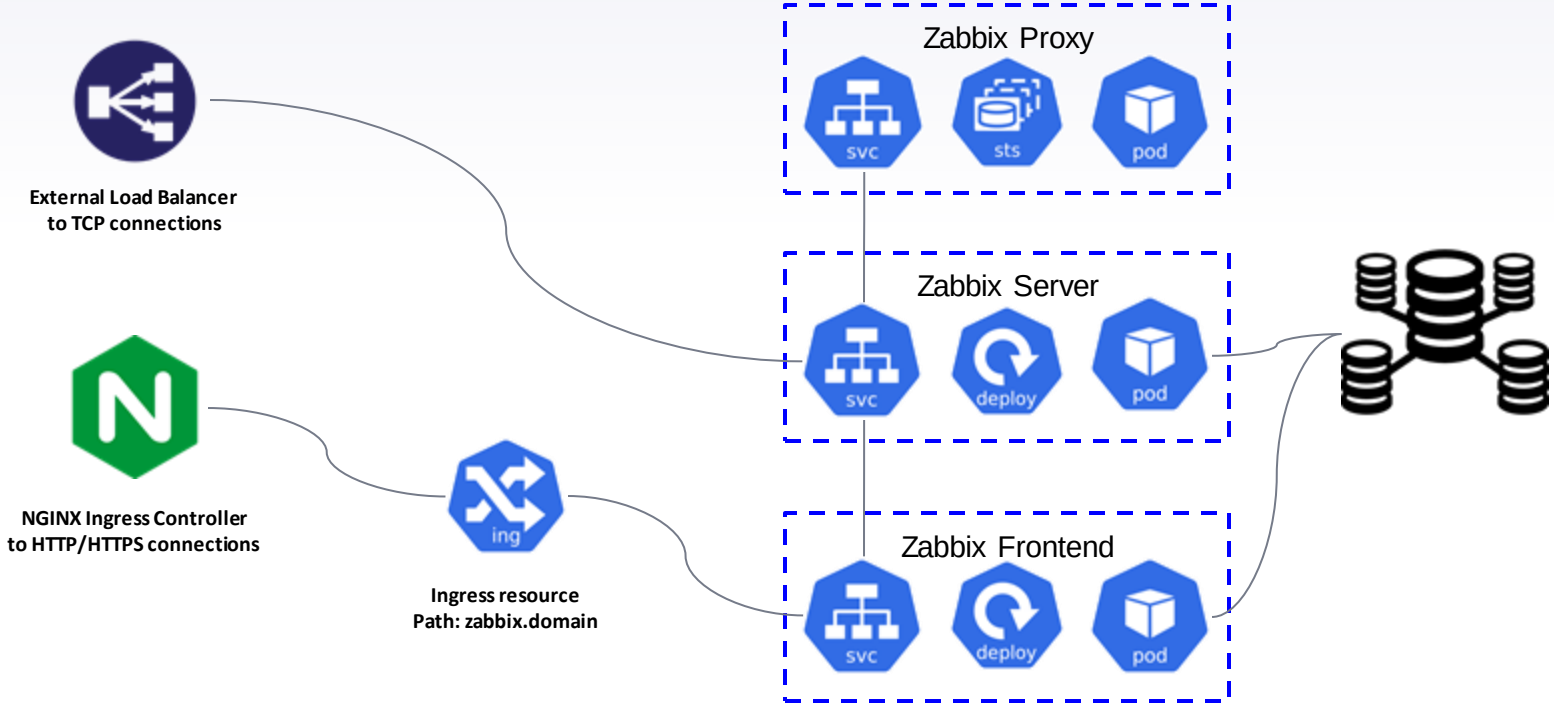
First steps

- Understand what resources we need to Deploy Zabbix in Kubernetes;
- Create an isolated environment for development, homolog and production;
- It is recommended to have 2 clusters:
  1. Development and homolog;
  2. Production.
- **You don't need a dedicated Zabbix cluster, just dedicated nodes.**
- Define how the Deploy will be done:
  - Gitlab CI/CD, ArgoCD, Harness, Azure Devops and others.

Kubernetes objetcs

Kubernetes basics objects to Zabbix

How does connectivity work?



Two ways to deploy to Kubernetes

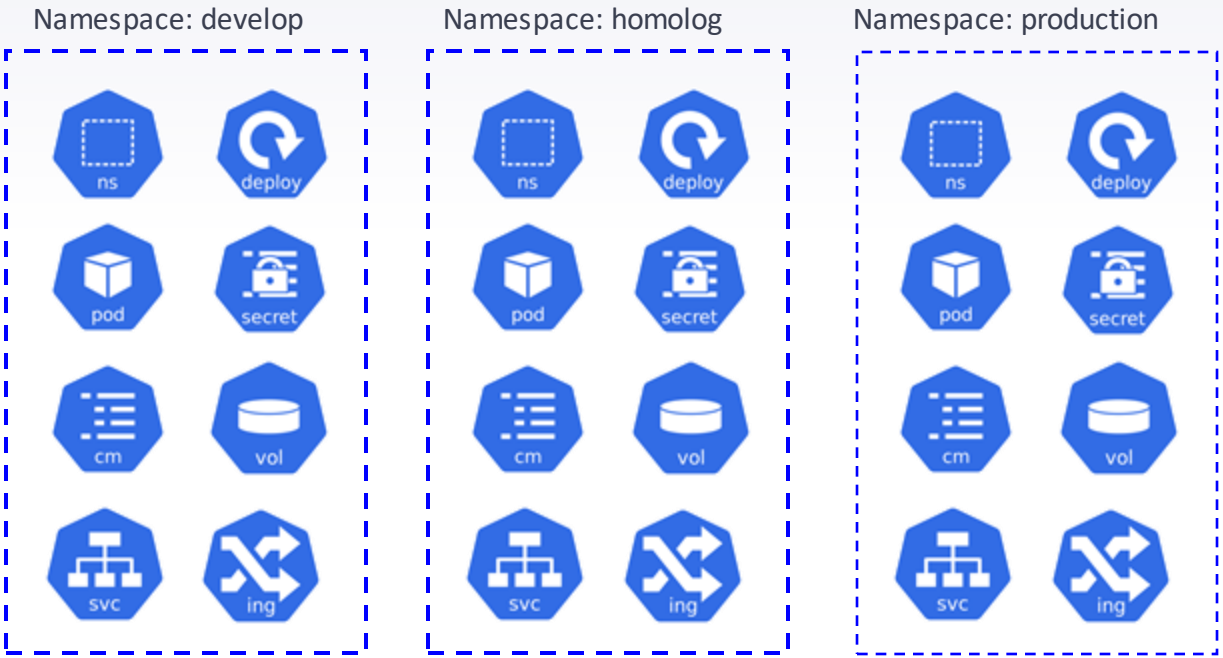
```
spec:
  nodeSelector:
    null
  containers:
  - name: zabbix-server-mysql
    image: zabbix/zabbix-server-mysql:5.0.28-ubuntu
    imagePullPolicy:
    ports:
    - containerPort: 10051
      protocol: TCP
    resources:
      limits:
        cpu: 2
        memory: 4Gi
      requests:
        cpu: 1
        memory: 2Gi
    readinessProbe:
      failureThreshold: 8
      initialDelaySeconds: 120
      periodSeconds: 10
      tcpSocket:
        port: 10051
    livenessProbe:
      initialDelaySeconds: 120
      tcpSocket:
        port: 10051
      timeoutSeconds: 1
```

Using manifest file

```
containers:
- name: {{ .Release.Name }}
  image: {{ .Values.image.repository}}:{{ .Values.image.tag}}
  imagePullPolicy: {{ .Values.image.pullPolicy }}
  {{- if .Values.container.args }}
  args:
  {{- toYaml .Values.container.args | nindent 10 }}
  {{- end }}
  ports:
  - containerPort: {{ .Values.port }}
    protocol: TCP
  {{- if .Values.spec.resources }}
  resources:
  {{- toYaml .Values.spec.resources | nindent 10 }}
  {{- end }}
  {{- if .Values.spec.readinessProbe }}
  readinessProbe:
  {{- toYaml .Values.spec.readinessProbe | nindent 10}}
  {{- end }}
  {{- if .Values.spec.livenessProbe }}
  livenessProbe:
  {{- toYaml .Values.spec.livenessProbe | nindent 10}}
  {{- end }}
```

Using helm

Deploy using Kubernetes manifest files



It is not good. We need to manage 24 yaml file and just need to change some parameters.

How to improve it?

Package manager

apt

dnf

What is helm?

- The package manager for Kubernetes;
- Provide a reusable facility and serve as a single point of control;
- Use Helm Rollback to easily revert to an older version.

Deploy using Helm

Namespace: develop

Namespace: homolog

Namespace: production

How Helm works?

- Helm is based in template files;
- We need create our templates and define values.yaml;

Example Helm values and template

```yaml
spec:
  replicas: 1
  resources:
    requests:
      cpu: 1
      memory: 2Gi
    limits:
      cpu: 2
      memory: 4Gi
  readinessProbe:
    tcpSocket:
      port: 10051
    initialDelaySeconds:
    periodSeconds: 10
    failureThreshold: 8
  livenessProbe:
    tcpSocket:
      port: 10051
    initialDelaySeconds: 120
    timeoutSeconds: 1
```

values.yaml

```yaml
containers:
- name: {{ .Release.Name }}
  image: {{ .Values.image.repository}}:{{ .Values.image.tag}}
  imagePullPolicy: {{ .Values.image.pullPolicy }}
  {{- if .Values.container.args }}
  args:
  {{- toYaml .Values.container.args | nindent 10 }}
  {{- end }}
  ports:
  - containerPort: {{ .Values.port }}
    protocol: TCP
  {{- if .Values.spec.resources }}
  resources:
  {{- toYaml .Values.spec.resources | nindent 10 }}
  {{- end }}
  {{- if .Values.spec.readinessProbe }}
  readinessProbe:
  {{- toYaml .Values.spec.readinessProbe | nindent 10}}
  {{- end }}
  {{- if .Values.spec.livenessProbe }}
  livenessProbe:
  {{- toYaml .Values.spec.livenessProbe | nindent 10}}
  {{- end }}
```

templates/deployment.yaml

Solution flow

Technologies involved

Git

Gitlab

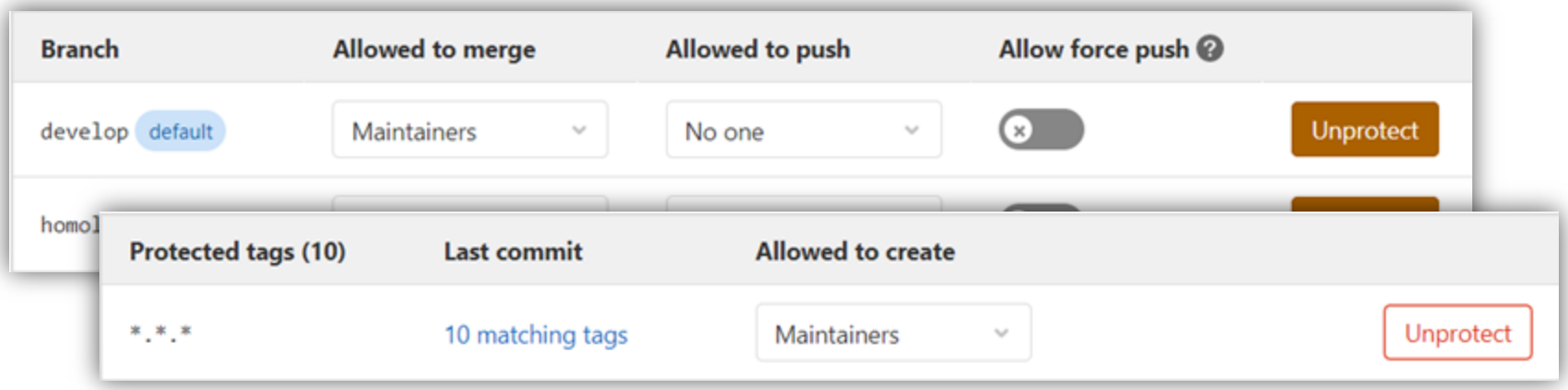
CI/CD

ArgoCD

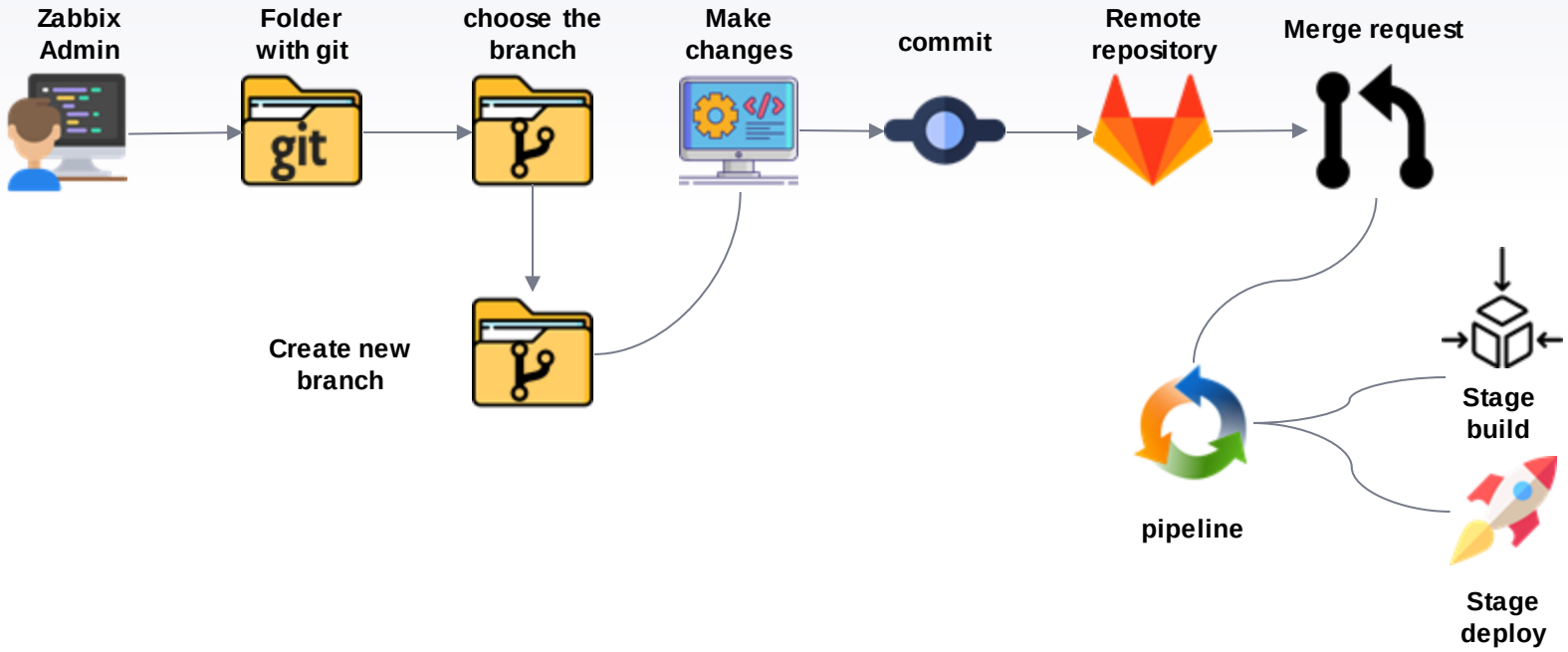
Kubernetes OR OpenShift

Best practices for working with branches

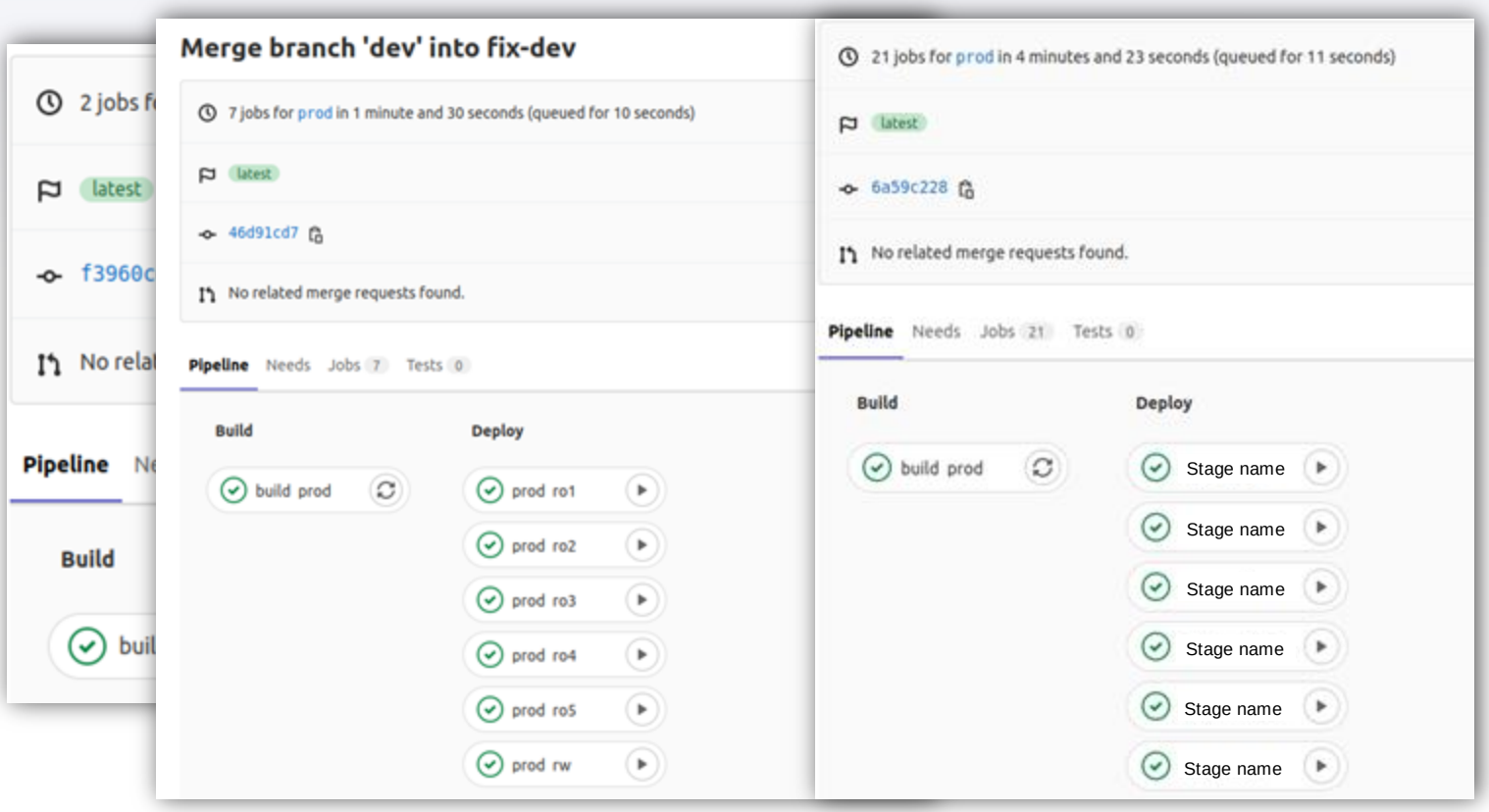
- Enable protected branches and tags;
- Keep stable branches secure, and force developers to use merge requests;
- Gitlab example:
  - Repo > Settings > Repository > Protected branches or tags.



Basic CI/CD Flow using Git and Gitlab with secure



CI/CD Flow using Git and Gitlab



Gitlab CI sample code using branches

Become dynamic CI/CD process
$CI_PROJECT_PATH: Repository name
$CI_COMMIT_REF_SLUG: Branch name
$CI_PIPELINE_ID: Pipeline ID

```
docker-build-dev-hom:
  stage: build_container
  script:
    - env
    - echo $CI_COMMIT_BRANCH
    - docker login -u gitlab-ci-token -p $CI_JOB_TOKEN $CI_REGISTRY
    - docker build --build-arg BUILD_DATE=`date -u +"%Y-%m-%dT%H:%M:%SZ"` \
      --build-arg BUILD_PIPELINE_ID=$CI_PIPELINE_ID \
      -t $CI_REGISTRY/$CI_PROJECT_PATH:${CI_COMMIT_REF_SLUG}-${CI_PIPELINE_ID} .
    - docker push $CI_REGISTRY/$CI_PROJECT_PATH:${CI_COMMIT_REF_SLUG}-${CI_PIPELINE_ID}
  only:
    refs:
      - develop
      - homolog
```

Define when this stage should run
In this case only branch is **develop** or **homolog**

Gitlab CD sample code

```yaml
# Using YAML Anchors
.default-deploy: &default-deploy
  stage: deploy
  image: "[redacted]/gold-openshift-cli:latest"
  variables:
    ENVIRONMENT_STAGE: # Define "develop" ou "homolog"
  environment:
    name: ${CI_COMMIT_REF_SLUG}
    url: https://${PROJECT_NAME_SLUG}.${INGRESS_DOMAIN}
    kubernetes:
      namespace: ${KUBE_NAMESPACE}
  before_script: ···
  script: ···
  except:
    variables:
      - $DEPLOY_ONLY # Não executa caso esta variável exista, pula o passo no fluxo de rollback ou redeploy
  only:
    - develop #Definir individualmente "develop" ou "homolog"
```

Gitlab CD sample code

```yaml
deploy-develop:
  <<: *default-deploy
  variables:
    ENVIRONMENT_STAGE: "develop"
  before_script:
```

Reference to YAML anchor

```yaml
deploy-homolog:
  <<: *default-deploy
  variables:
    ENVIRONMENT_STAGE: "homolog"
  before_script:
```

Reference to YAML anchor

Variable	KUBE_NAMESPACE	**********	✗	✗	All (default)	
Variable	INGRESS_DOMAIN	**********	✗	✗	All (default)	
Variable	INGRESS_DOMAIN	**********	✓	✗	develop	
Variable	INGRESS_DOMAIN	**********	✗	✗	homolog	

Gitlab CD sample code

```
deploy-develop:
  <<: *default-deploy
  variables:
    ENVIRONMENT_STAGE: "develop"
  before_script:
    - if [ -z "${OCP_TOKEN_[redacted]}" ]; then echo "Variavel de TOKEN nao encontrada" && exit 1; fi
    - oc login --server=${OPENSHIFT_[redacted]}:6443 --token=${OCP_TOKEN_[redacted]}
  script:
    - chmod +x deploy_helm.sh
    - ./deploy_helm.sh
  only:
    - develop
```

Reference to YAML anchor

Run script to deploy

Result from CI/CD process

Deploy

passed	_prod	#5338257 runner manual	00:00:28 1 month ago	
passed	_prod	#5338258 runner manual	00:00:22 1 month ago	
passed	_prod	#5338259 runner manual	00:00:26 1 month ago	
passed	_prod	#5338260 runner manual	00:00:26 1 month ago	
passed	prod	#5338261 runner manual	00:00:25 1 month ago	
passed	_prod	#5338262 runner manual	00:00:22 1 month ago	
passed	_prod	#5338263 runner manual	00:00:15 1 month ago	
passed	_prod	#5338264 runner manual	00:00:29 1 month ago	

Benefits of using Git

- All centralized environment settings, such as:
  - External scripts and AlertScripts;
  - odbc.ini and odbcinst.ini;
  - All configuration parameters:
    - CacheSize, ValueCache, Pollers and others.
- View all changes:
  - Who changed;
  - What changed;
  - Because it changed.

Benefits of using CI/CD

- No manual intervention;
- Automatic docker image build and update;
- Automatic setup process.

THANKS!

Any questions?

You can find me at:

Credits

Special thanks to all the people who made and released these awesome resources for free:

- Presentation template by SlidesCarnival
- Illustrations by Sergei Tikhonov
- Photographs by Unsplash