SDLC

Table of Contents

Overview 1
- Architecture 1
- CI/CD flow 2
- Stack 2

Setup Cluster & Environment Repository 2
- Prerequisites 2
- Setup 2

Pipelines 6
- Pipelines catalog and pack 6

Large Tests 6
- Enable 6
- Selenium 7
- Slack notification 7

Keycloak 7

Metrics 7
- Alertmanager 7

Example Apps 7

Runbook 8
- Cluster delete 8
- Vault 8
- Keycloak 8

Other 8
- SDLC glossary 8
- GitHub project 8
- Diagrams 8

Overview

Architecture

Simple view

```
<iframe style="border:none" width="100%" height="550"
src="https://whimsical.com/embed/DGQcqfVJMY5CuP4z2QGU1s"></iframe>
```

Extended

```
<iframe style="border:none" width="100%" height="550"
src="https://whimsical.com/embed/8GKYgDts1C3ZKb3ccsrzpR"></iframe>
```

CI/CD flow

```
<iframe allowfullscreen frameborder="0" style="width: 100%; height: 550px;"
src="https://lucid.app/documents/embeddedchart/88013d3c-9451-45de-b97c-87e3bf8dff8a"
id="ZddTO1PVGjTd"></iframe>
```

Stack

- Jenkins X - CORE
- Tekton - CI/CD
- Kubernetes
- Terraform
- Keycloak - OpenId provider
- WebdriverIO - Large test framework

Setup Cluster & Environment Repository

Prerequisites

- Create a git bot user (different from your own personal user) and generate a personal access token, this will be used by Jenkins X to interact with git repositories.
- Terraform CLI
- Jenkins X CLI
- Kpt
- Google Cloud SDK
    - Enable workload identity for k8s cluster
- generate Sonar cloud token
- generate Slack Incoming Webhooks
- generate Snyk token

Setup

1. Create Infrastructure Repository

Create infrastructure repository for GKE: https://github.com/jx3-gitops-repositories/jx3-terraform-gke/generate

2. Create Environment Repository

Create environment repository: https://github.com/vitech-team/jx3-gke-vault

3. Prepare install script

install.sh

```
#!/usr/bin/env bash

export INFRA_REPO_NAME="demo-infra" ①
export ENV_REPO_NAME="demo-environment" ②

export INFRA_GIT="https://github.com/vitech-team/$INFRA_REPO_NAME.git" ③
export ENV_GIT="https://github.com/vitech-team/$ENV_REPO_NAME.git" ③

export TF_VAR_jx_bot_username=XXX ④
export TF_VAR_jx_bot_token=XXX ④

export CLUSTER_NAME="demo-time" ⑤
export GCP_PROJECT="XXX" ⑥
export ZONE="europe-west1-c" ⑦
export MIN_NODE_COUNT="4" ⑧
export FORCE_DESTROY="false" ⑨

export green="\e[32m"
export nrm="\e[39m"

git clone $INFRA_GIT
git clone $ENV_GIT

cd $INFRA_REPO_NAME || exit

rm values.auto.tfvars
⑩
cat <<EOF >>values.auto.tfvars
resource_labels = { "provider" : "jx" }
jx_git_url = "${ENV_GIT}"
gcp_project = "${GCP_PROJECT}"
cluster_name = "${CLUSTER_NAME}"
cluster_location = "${ZONE}"
force_destroy = "${FORCE_DESTROY}"
min_node_count = "${MIN_NODE_COUNT}"
EOF

git commit -a -m "fix: configure cluster repository and project"
git push

terraform init
terraform apply

echo -e "${green}Setup kubeconfig...${nrm}"
gcloud container clusters get-credentials "${CLUSTER_NAME}" --zone "${ZONE}" --project
"${GCP_PROJECT}"

echo "Taling logs..."
jx admin log

echo -e "${green}Okay, now we are creating new key for service account...${nrm}"
gcloud iam service-accounts keys create keyfile.json --iam-account "${CLUSTER_NAME}
-tekton@${GCP_PROJECT}.iam.gserviceaccount.com"
SECRETNAME=docker-registry-auth
kubectl create secret docker-registry $SECRETNAME \
  --docker-server=https://gcr.io \
  --docker-username=_json_key \
  --docker-email=sdlc@vitechteam.com \
  --docker-password="$(cat keyfile.json)" \
  --namespace=jx
kubectl label secret $SECRETNAME secret.jenkins-x.io/replica-source=true --namespace
=jx

jx namespace jx

echo -e "For vault root token use: ${green}kubectl get secrets vault-unseal-keys  -n
secret-infra -o jsonpath={.data.vault-root} | base64 --decode${nrm}"
```

① infrastructure repository name

② environment repository name

③ infra. and env. repo URLs

④ GitHub user name and token. **User should habe settings permission to all repositories**

⑤ cluster name

⑥ GCP project id

⑦ cluster zone: https://cloud.google.com/compute/docs/regions-zones#available

⑧ default node count

⑨ if buckets and PVCs should be deleted in case of `terraform destroy` command

⑩ store cluster configs in file. for more configs see: https://github.com/jx3-gitops-repositories/jx3-terraform-gke#terraform-inputs

4. Populate secrets

4.1 Create vault proxy

Fist we need start vault proxy

sec-vault-start.sh

```
jx secret vault portforward
```

4.2 Auto populate secrets

sec-auto-populate.sh

```
jx secret populate
```

4.3 Populate required secrets

sec-required-populate.sh

```
jx secret edit -f slack

jx secret edit -f snyk

jx secret edit -f sonar
```

Secrets also can be populated via Vault UI see: Vault

4.4 Verify secrets

Execute `jx secret verify` and check if all needed secrets are populated like: `sonar`, `slack`, etc…

5. Create application based on SDLC quickstart

5.1 Spring ?

If you need some REST API backend service use template with name: `vitech-sdlc-backend`

quick-start-backend.sh

```
YOUR_ORG_NAME="vitech-team"
jx project quickstart --pack="spring-gradle" --org="${YOUR_ORG_NAME}"
```

5.2 Angular ?

If you need frontend application on Angular use: `vitech-sdlc-frontend` *quickstart-forntend.sh*

```
YOUR_ORG_NAME="vitech-team"
jx project quickstart --pack="angular" --org="${YOUR_ORG_NAME}"
```

After setup you need edit default configs in `environments` folder.

- keycloak url: `kubectrl get ingress -n keycloak`
- change backend service name in `nginx.conf`

Pipelines

Pipelines catalog and pack

All shared tasks and packs stored in: https://github.com/vitech-team/tekton-pipelines-catalog

Packs

Custom packs: https://github.com/vitech-team/tekton-pipelines-catalog/tree/master/packs

All tasks, packs and pipelines are in sync with environment repository via Kpt.

more information about tasks and pipelines check Tekton docs

more information about pipelines on JX see JX Pipeline Docs

Large Tests

Currently, we have only Large Tests implementation based WebdriverIO. We added a few steps to `release` and `pullrequest` pipelines:

- Check if large test been executed on **Staging** before promote it on Production **environment**
- Execute **Large Tests** after changes been applied on environment like Production.

Enable

- open `.lighthouse/large-test/triggers.yaml` and change: `always_run: false`, `optional: false` to `true`.
- open `.lighthouse/jenkins-x/release.yaml` and uncomment commented tasks: ` large-test-prepare-and-check` and `large-test-execute`
  - change large test image name property: `LARGE_REPORTS_IMAGE`
  - change your app URLs properties: `APP_URL_STAGING`, `APP_URL_PRODUCTION`, if you have more environments just add additional property like: `APP_URL_XXX`

Selenium

For selenium hub config use:

- `charts/dev/largetests/values.yaml.gotmpl`
- `charts/dev/largetests/values.yaml`

List of all configs: https://github.com/helm/charts/tree/master/stable/selenium#configuration

Slack notification

If you wanna change Large test execution message to slack, open and change: `charts/dev/secret/templates/slack-messages.yaml`

> You can use next variables what can be populated/replaced: `$idea`, `${STATUS}`, `${REPORT_URL}`, `${DETAILS}` and `${GIT_SHA}`

Keycloak

For Keycloak configuration use: `charts/dev/keycloak/values.yaml.gotmpl` file in env. repository list of keycloak configs: https://github.com/codecentric/helm-charts/tree/master/charts/keycloak#configuration

Metrics

Metrics chart kube-prometheus-stack

- For custom monitors and gradana dashboard use folder: `charts/dev/prometheusmonitors/templates`
- For metrics stack configuration use
  - `charts/prometheus-community/kube-prometheus-stack/values.yaml.gotmpl`
  - `charts/prometheus-community/kube-prometheus-stack/values.yaml`

Alertmanager

Configure Slack Notifications

- In vault find `alertmanager.yaml` secret and replace `${SLACK_HOOK_URL}` with your hook URL. See example: `charts/prometheus-community/kube-prometheus-stack/secret-schema.yaml`

Example Apps

- Frontend app example
- Backend example

- Large test example

Runbook

Cluster delete

- for the cluster delete `cd` to your infra. repository and execute `terraform destroy`

Vault

- For port forward Vault type: `jx secret vault portforward` - after that you can rich Vault at https://localhost:8200
- Vault root token can be found in secret: `vault-unseal-keys`, key: `vault-root`

Keycloak

- keycloak url: `kubectrl get ingress -n keycloak`

Other

SDLC glossary

https://github.com/vitech-team/SDLC/wiki/SDLC

GitHub project

https://github.com/vitech-team/SDLC/projects

Diagrams

- https://lucid.app/lucidchart/invitations/accept/c7c8be31-1804-4a2c-8db6-1300d64974ee
- https://whimsical.com/sdlc-9iJvu6pNAXzUQBYYR61qAM