Playground Environments with k8s

Xavier Orduña

DEVOPS Barcelona Meetup

March 2024

About me

- I learned to code in the last centry (Visual Basic and HTML)
- Computer Science and Engineering at UPC
- Founded DEXMA (2007 – 2015)
- Data Engineering Freelance (TESCO, ABI, Moonpay, …) (2015 – 2021)
- Team Lead at Circutor (2021 – present)
- And still freelancing for some companies …

OBSESSED ABOUT AUTOMATION!!!!

What is a Playground environment*?

- A Preview environment
- For each MR / PR
- With its own frontend
- With its own backend
- With its own database
- That is destroyed at merge

* Otherwise named feature environments

DEMO

Circutor / Cloud / myc-cloud / Issues / #1153

Close workspaces (front end)

Edit

Open Issue created 3 weeks ago by [redacted]

Overview

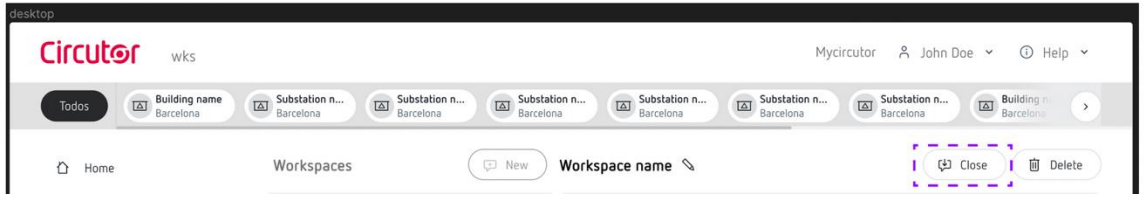
Users can close workspaces

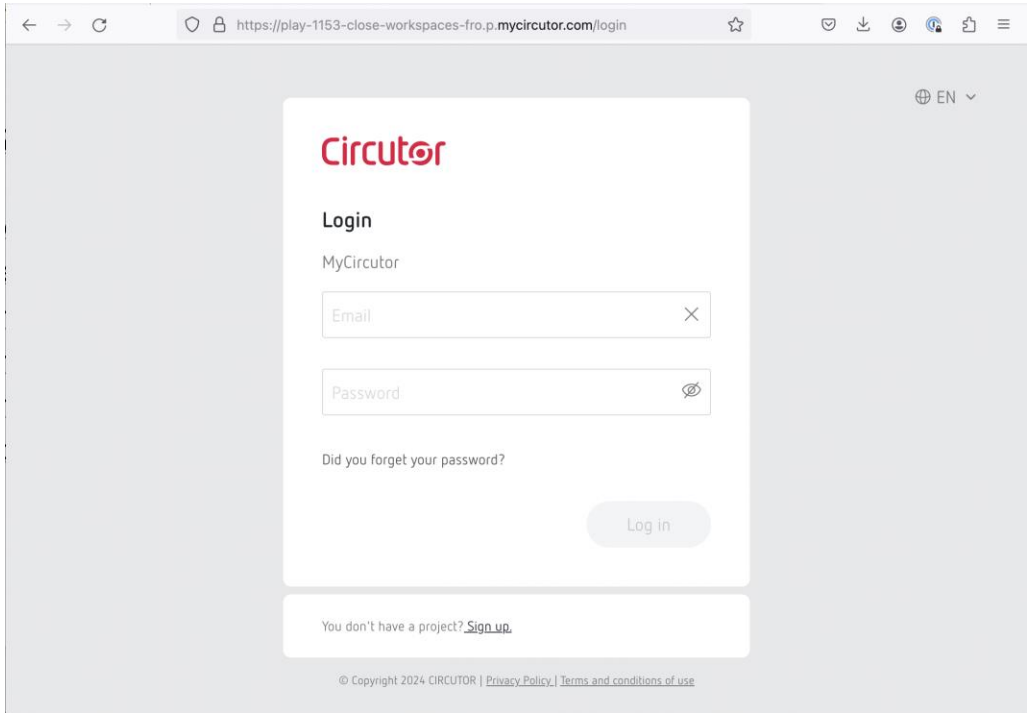
Tasks

- Add button to close the workspace
- Create the modal to validate that the user wants to close the workspace
- Create the message of who and when the message was closed and condition the message to be read-only.
- Add closed status to workspace list cards

Design

Figma link





Why?

- Quick product reviews
- Easy way to run isolated end 2 end tests
- Load tests
- Team colaboration

Advantages

- Quicker functional review times
- Enables developer to see how the whole system work, not only the service they are implemeting
- Frontend do not need to provision Backend
- Backend do not need to provision Frontend
- They can be used for many things (debugging, load testing, mobile, hardware integration)

Disadvantages

- They are memory hungry!!! its a whole system
- Developers rely too much on them, this leads to higher build times
- Its a “delighter” that defines your whole architecture
- Cost (K8S, …)

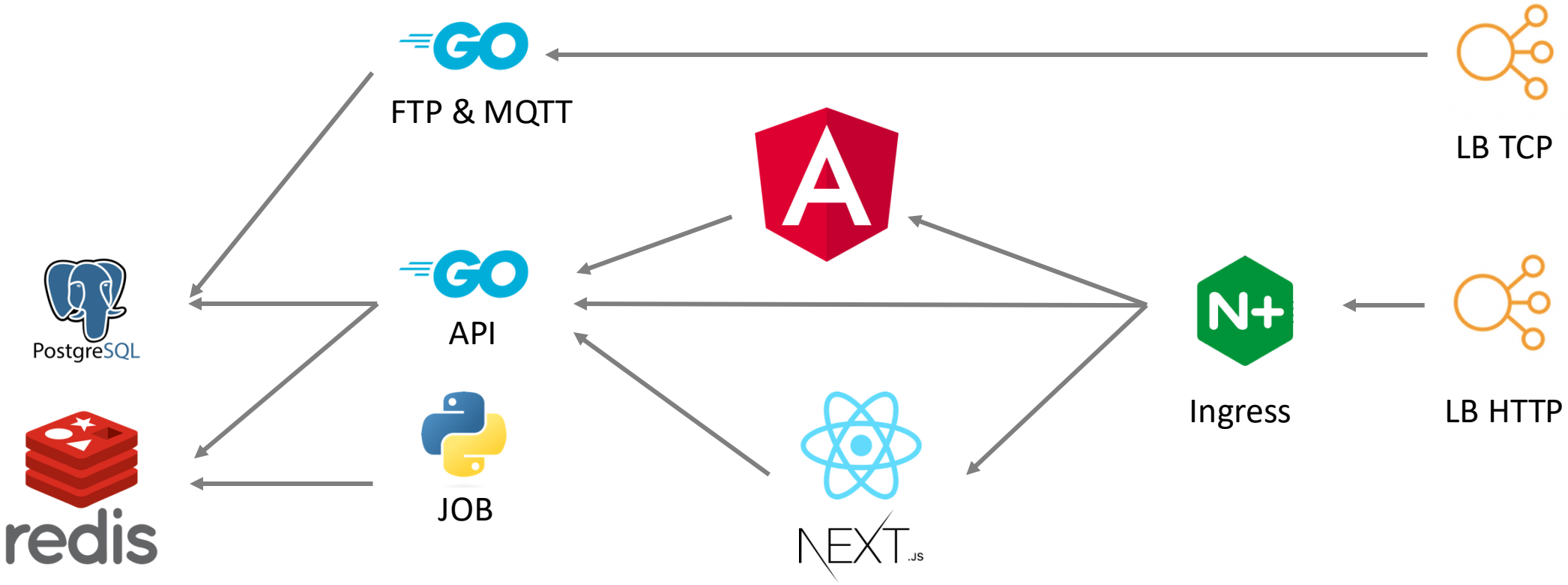
Recipe

- K8S Cluster
- Helm
- Gitlab “monorepo”

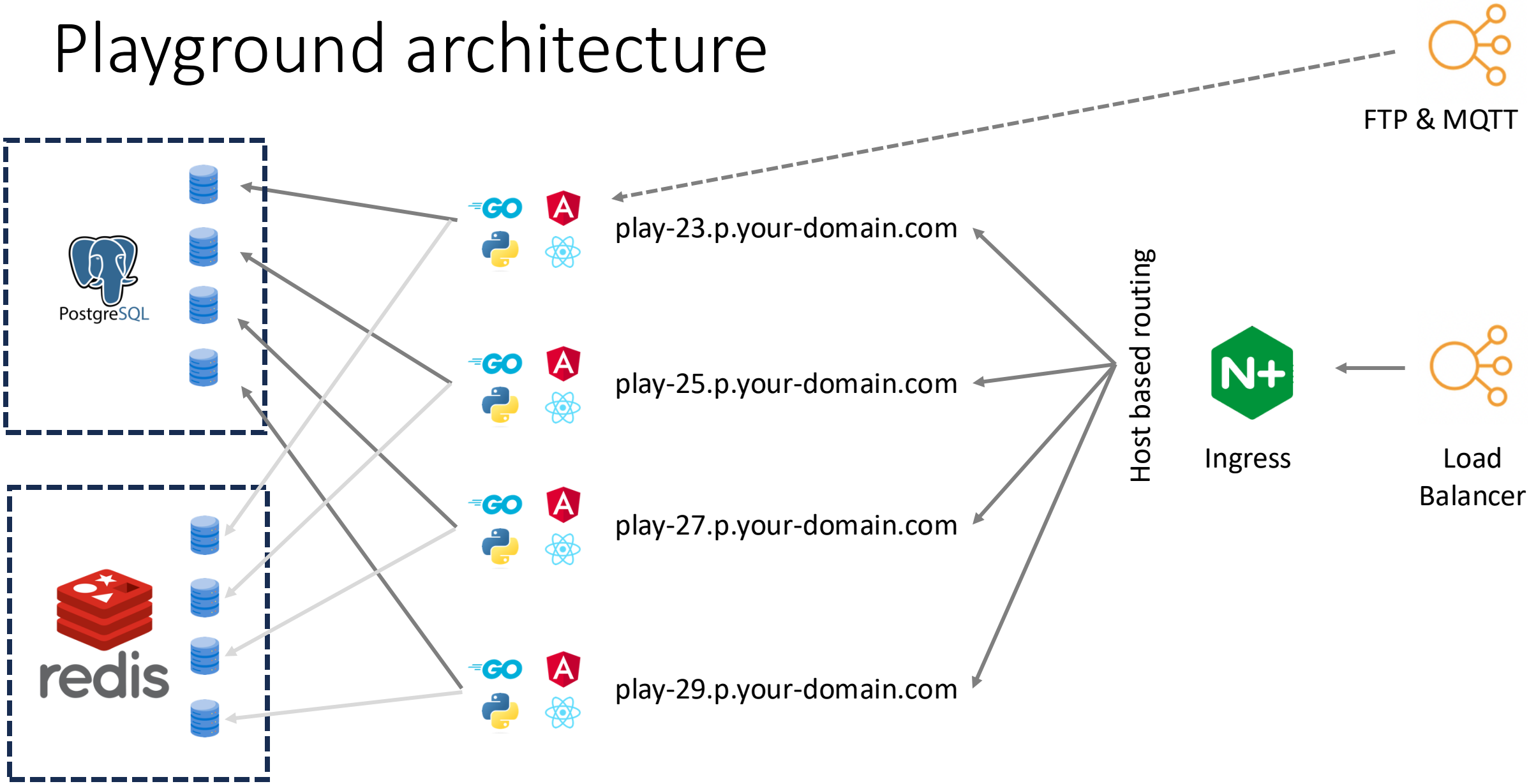
- And lots, lots of refinement …

Be carefull, some actions in this presentations require a lot of “faith” and involve some kind of goat sacrifices. Those moments are marked with this icon.

Example application



Playground architecture



Some examples

```
~/devel/playground-envs-demo  on 🌿 1-my-random-issue (⎈ |arn:aws:eks:eu-west-1:224392328862:cluster/dev-circutor:default) 🕒 16:07:51
$ helm list -n playground
NAME                            NAMESPACE       REVISION        UPDATED                                 STATUS          CHART                   APP VERSION
play-1160-associate-workspaces  playground      1               2024-02-29 12:33:06.177389115 +0000 UTC deployed        myc-cloud-0.0.23        1.0.0
play-1168-integrage-hydrators   playground      6               2024-03-05 13:48:54.803862768 +0000 UTC deployed        myc-cloud-0.0.23        1.0.0
play-1178-fix-wks-admin-action  playground      1               2024-03-05 13:01:26.0558326 +0000 UTC   deployed        myc-cloud-0.0.23        1.0.0
play-1181-create-eventalarm-ta  playground      12              2024-02-27 15:46:21.113302357 +0000 UTC deployed        myc-cloud-0.0.23        1.0.0
play-1197-waveforms-list-endpo  playground      3               2024-03-05 14:46:28.106130157 +0000 UTC deployed        myc-cloud-0.0.23        1.0.0
play-1198-fix-ts-from-gateways  playground      8               2024-03-05 14:25:54.491916498 +0000 UTC deployed        myc-cloud-0.0.23        1.0.0
play-1199-fix-asset-name-jn-la  playground      2               2024-03-04 07:34:15.477128789 +0000 UTC deployed        myc-cloud-0.0.23        1.0.0
play-1200-backend-architecture  playground      1               2024-03-01 15:40:28.295783179 +0000 UTC deployed        myc-cloud-0.0.23        1.0.0
play-1203-alarm-detail-layout   playground      2               2024-03-05 09:31:05.329947114 +0000 UTC deployed        myc-cloud-0.0.23        1.0.0
play-perform-account-asset-hyd  playground      3               2024-03-04 10:33:56.72623633 +0000 UTC  failed          myc-cloud-0.0.23        1.0.0
```

```
~/devel/playground-envs-demo  on 🌿 1-my-random-issue (⎈ |arn:aws:eks:eu-west-1:224392328862:cluster/dev-circutor:default) 🕒 16:07:59
$ kubectl get pods -n playground | grep 1203
play-1203-alarm-detail-layout-email-viewer-56f5d556df-xr7cd      1/1     Running     0          160m
play-1203-alarm-detail-layout-myc-api-785f68f9c-zfgkk            1/1     Running     0          160m
play-1203-alarm-detail-layout-myc-bg-667888bb57-jqqcw            1/1     Running     0          10h
play-1203-alarm-detail-layout-myc-frontend-v1-6db86c8b7f-zngjj   1/1     Running     0          11h
play-1203-alarm-detail-layout-myc-ftp-6b5699fbb4-z72ck           1/1     Running     0          3h28m
play-1203-alarm-detail-layout-myc-mqtt-broker-599d6fc65-8v76n    1/1     Running     0          10h
play-1203-alarm-detail-layout-myc-mqtt-broker-shelly-559fbbw7hp  1/1     Running     0          10h
play-1203-alarm-detail-layout-myc-report-engine-7df4f6dcdd8rlq5  1/1     Running     0          5h38m
play-1203-alarm-detail-layout-myc-wks-5787d65cfc-gwzrd           1/1     Running     0          5h38m
```

Deploy a playground step by step

- **Build Docker images**
- Define variables
- Setup database
- Prepare tags
- Install helm
- Run end2end
- Destroy

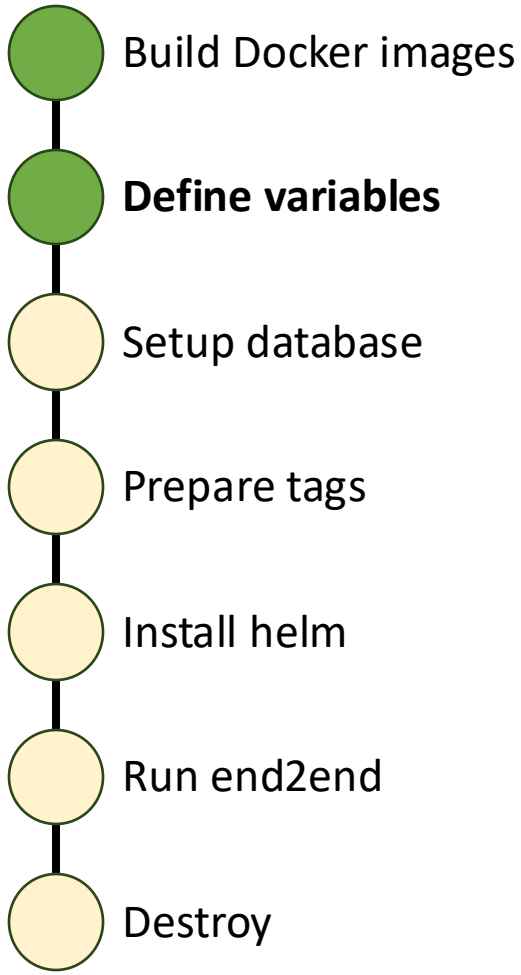
Build Docker image:

TAG: <branch_name>_<commit_hash>

When main_<commit_hash> has been deployed succesfully tag it as `stable`

It is very important that all parameters are passed as environment variables to be setup at RUNTIME

Deploy a playground step by step



Existing variables (CI/CD)

```
DEV_POSTGRES=<postgresq://user:pass@hostname>
DEV_REDIS=<redis://server>

CI_COMMIT_REF_SLUG // GITHUB_REF_NAME
```

Pipeline defined variables

```
PLAYGROUND_ENV=play-${CI_COMMIT_REF_SLUG:0:25}
BRANCH_DATABASE=`echo "$PLAYGROUND_ENV" | tr - _`
POSTGRES_URI=$PLAYGROUND_PG/$BRANCH_DATABASE
REDIS_URI=$DEV_REDIS/${CI_COMMIT_REF_SLUG%%-*}
BRANCH_HOSTNAME=$PLAYGROUND_ENV.p.mycircutor.com
DYNAMIC_ENVIRONMENT_URL=https://$BRANCH_HOSTNAME
```

Deploy a playground step by step

- Build Docker images
- Define variables
- **Setup database**
- Prepare tags
- Install helm
- Run end2end
- Destroy

```
$ echo "SELECT 'CREATE DATABASE $BRANCH_DATABASE' WHERE NOT EXISTS (SELECT FROM
      pg_database WHERE datname = '$BRANCH_DATABASE')\gexec" | psql
$PLAYGROUND_PG
```

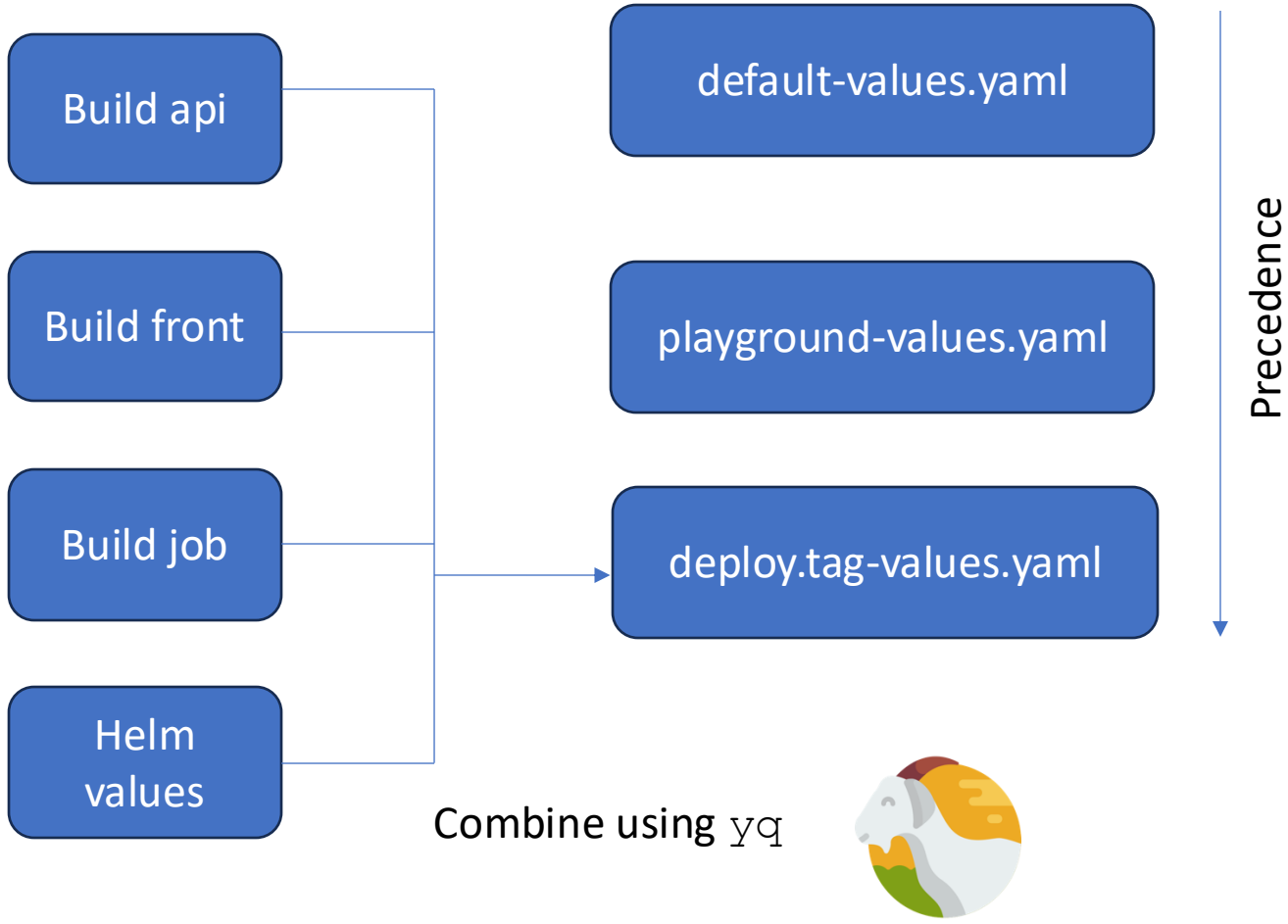
```
$ goose postgres up GOOSE_DBSTRING=$POSTGRES_URI
```

```
$ make upload-test-data DB_URI=$POSTGRES_URI
```

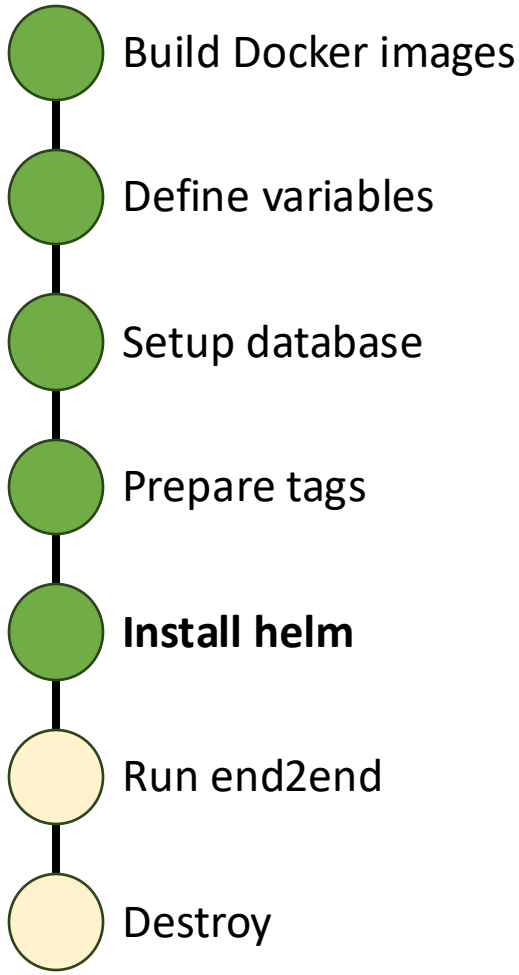
https://pressly.github.io/goose/

Deploy a playground step by step

- Build Docker images
- Define variables
- Setup database
- **Prepare tags**
- Install helm
- Run end2end
- Destroy



Deploy a playground step by step




```
$ helm upgrade --install $PLAYGROUND_ENV deployment/k8s/chart
    -f deployment/k8s/chart/values.yaml
    -f deployment/k8s/environments/playground-values.yaml
    -f deploy.tag-values.yaml
    --set global.hostname=$BRANCH_HOSTNAME
    --set global.environment=$PLAYGROUND_ENV
    --set global.baseUrl=$DYNAMIC_ENVIRONMENT_URL
    --set db.uri=$POSTGRES_URI
    --set redis.uri=$REDIS_URI
    --description "Deploy $buildVersion to $updatedTags"
    --debug --wait
```

Deploy a playground step by step

- Build Docker images
- Define variables
- Setup database
- Prepare tags
- Install helm
- **Run end2end**
- Destroy

Cypress runs all end2end tests in a clean environment each time

Deploy a playground step by step

- Build Docker images
- Define variables
- Setup database
- Prepare tags
- Install helm
- Run end2end
- **Destroy**

```
#!/bin/bash
regex="Merge branch '([A-Za-z0-9-]+)' into '([A-Za-z0-9-]+)'"
if [[ $CI_COMMIT_TITLE =~ $regex ]]; then
    echo ${BASH_REMATCH[1]} "->" ${BASH_REMATCH[2]}
    PLAYGROUND_ENV=${BASH_REMATCH[1]:0:25}
    if [[  "$PLAYGROUND_ENV" == *- ]]; then PLAYGROUND_ENV=${PLAYGROUND_ENV::-1}; fi
    export BRANCH_DATABASE=play_`echo "$PLAYGROUND_ENV" | tr - _`
    if helm list -n playground | grep play-$PLAYGROUND_ENV; then
        echo "Uninstalling playground-$PLAYGROUND_ENV"
        helm uninstall play-$PLAYGROUND_ENV -n playground
        # delete database ...
        echo "Deleting database"
        envsubst < deployment/scripts/delete-branch-database.sql > delete-database.sql
        cat delete-database.sql
        cat delete-database.sql | psql $PLAYGROUND_PG
        echo "Environment completely removed"
    else
        echo "No playground-$PLAYGROUND_ENV found"
    fi
else
    echo "No branch found in commit title"
fi
```

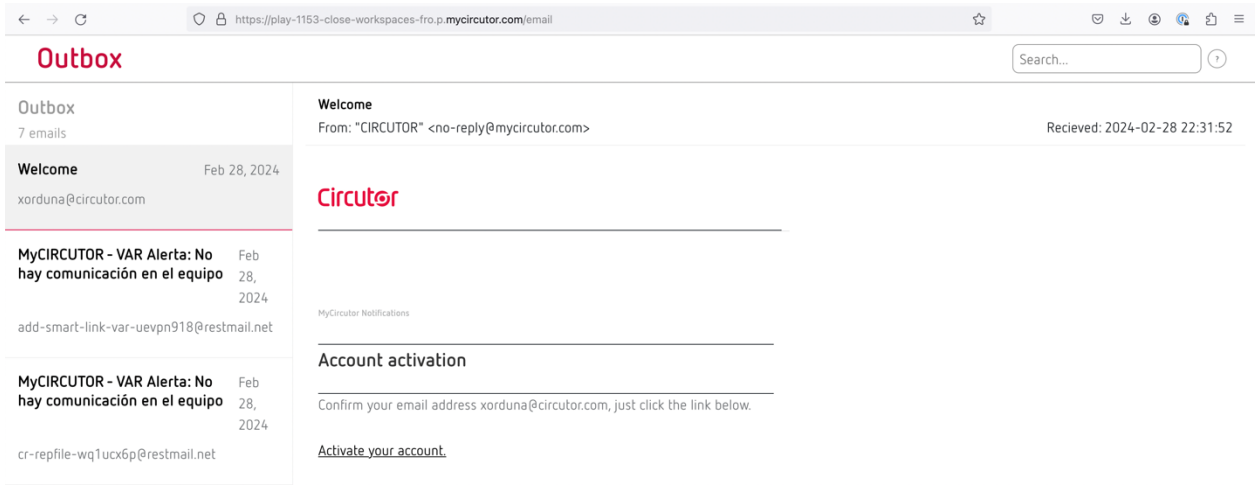
This complex script failed very often, so we ended up with a daily cleaning script looking for environments where the issue is closed

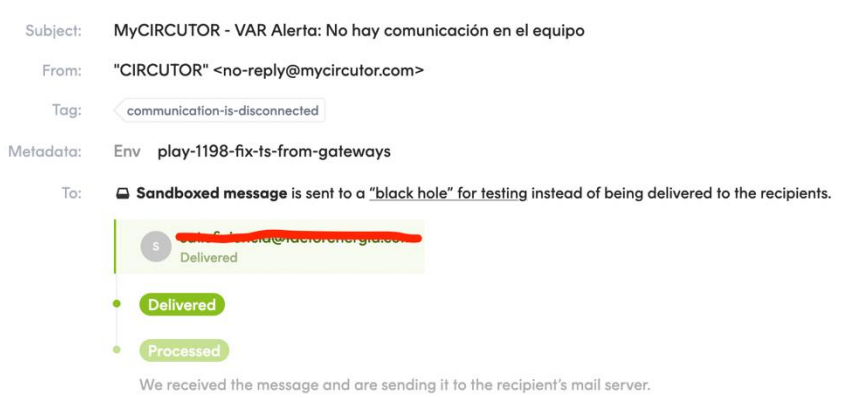
Tips and tricks - ID

- It is very important to define an ID for the environment (in our case is the issue ID)
- The ID number is used as database number in REDIS
- We use the first 25 characters of the issue for simplicity

Tips and tricks - Email

- We use postmark sandboxes to send email in playgrounds.
- All mails have metadata with the environment
- A custom email viewer (only for playgrounds) lets developer to see all mails.





Tips and Tricks – Environment Variables

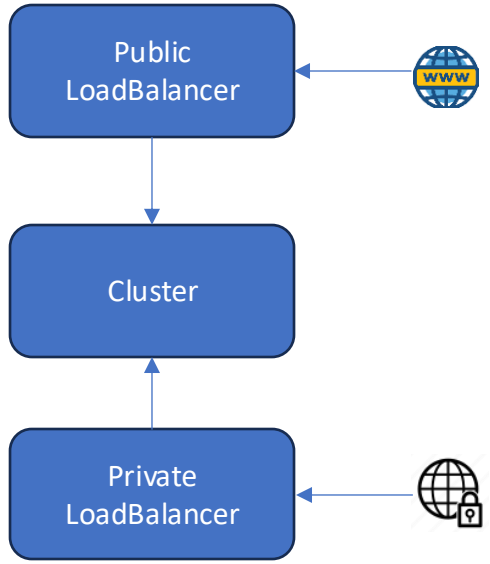
- All program variables should be defined at RUNTIME (not at build)
- Easy for Python, Go, ….
- Not so easy for Angular, React, NextJS, …

```
#!/bin/sh
envFilename='./.env.production'
nextFolder='.next/'

export
echo "Replace environment variables in Nextjs!"

while IFS='=' read -r configName configValue; do
    # no comment or not empty
    if [ "${configName#\#}" != "$configName" ] || [ -z "$configName" ]; then
        continue
    fi


    # get system env
    envValue=$(env | grep "^$configName=" | cut -d'=' -f2-)
    echo "configName: $configName configValue: $configValue envValue: $envValue"

    # if config found && configName starts with NEXT_PUBLIC
    if [ -n "$configValue" ] && [ -n "$envValue" ]; then
        # replace all
        echo "Replace: $configValue with $envValue"
        find "$nextFolder" \( -type d -name .git -prune \) -o -type f -print0 | xargs -0 sed -i '
    fi
done < "$envFilename"

echo "Starting Nextjs"
exec "$@"
```

```
CMD ["/bin/sh", "-c", "envsubst <
/usr/share/nginx/html/assets/env.template.js >
/usr/share/nginx/html/assets/env.js && exec nginx -g
'daemon off;'"]
```

Tips and Tricks - Domains

- Playgrounds are only accesible via VPN
- Dev cluster has public and private LB
- And public and private ingress
- First we had *.p.mycircutor.com pointed to private LB
- Now certs and domains are managed automatically with Lets Encrypt, Cloudflare DNS and Nginx Ingress. And we can expose playgrounds temporary to the internet

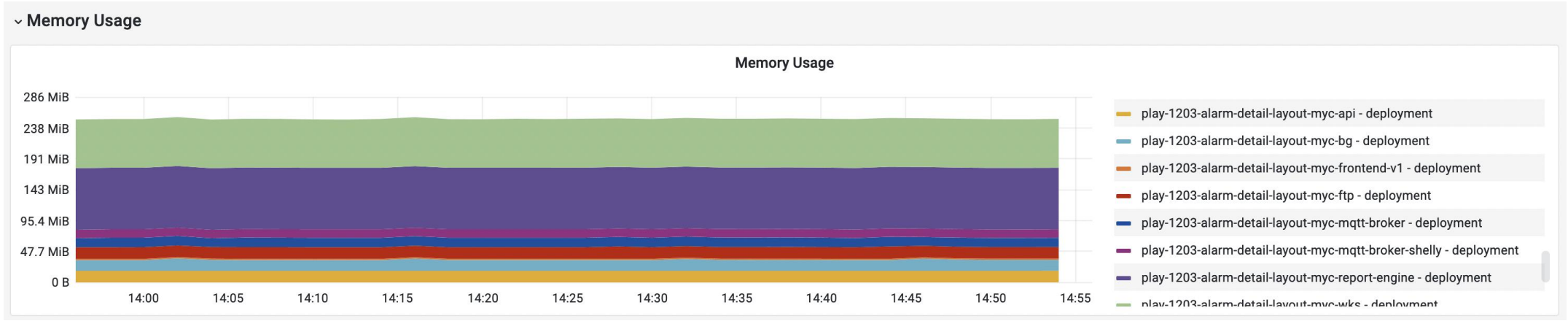


Tips and Tricks - Database

- Keep in mind that each playground can have a lot of connections to a DB
- Use a database only for playgrounds
- Use ID as DB num in Redis
- Use schema migration tools as goose
- Create a small subset of data that is loaded on each Playground
- We have a job to copy a snapshot of production database into a playground.

Tips and Tricks - Cost

- Keep memory and CPU requests very low since most of playgrounds will be iddle
- Add a “cleanup” job that is executed daily and looks for closed issues with still a playground
- Only for HTTP Services
- Our current memory footprint is ~ 240Mb



Tips and Tricks – TCP based services

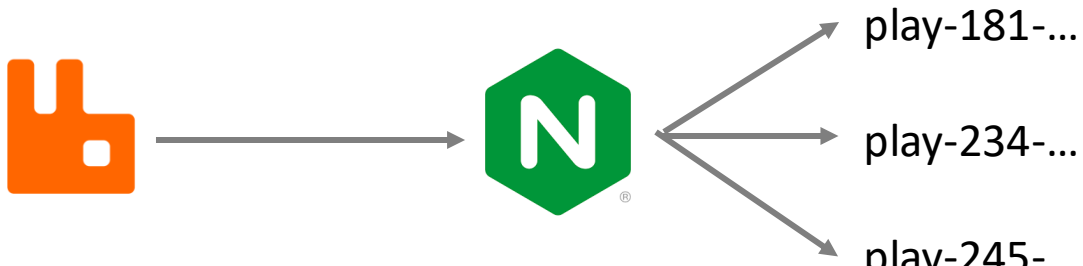
- TCP services cannot be routed using “hostname” like HTTP
- Right now we need to create a load balancer for each TCP services exposed.
- Trick: we expose the TCP service depending on a flag: `exposeMQTT` which by default is false.
- Trick: use Kong Ingress and SNI header based routing: https://docs.konghq.com/kubernetes-ingress-controller/latest/guides/services/tcp/

Tips and Tricks - RabbitMQ

- We want to use RabbitMQ with STOMP plugin and custom Auth API
- RabbitMQ supports virtual hosts but only a single Authentication endpoint
- This is an example of how playgrounds impact your arquitecture

```
server {
    listen 80;
    server_name my-hostname.com;

    location /api/resource {
        # Extreu el valor de 'vhost' de la URL
        if ($arg_vhost ~ ^(.+)$) {
            set $target_host $1;
        }

        # Afegeix els paràmetres original a la nova URL
        set $target_uri /api/resource?vhost=$arg_vhost&resource=$arg_resource;

        # Utilitza 'proxy_pass' per enviar la petició al servidor corresponent amb la nova URL
        proxy_pass http://$target_host$target_uri;
    }
```

Reverse proxy based on query parameter



Thank you so much!

xavier.orduna@gmail.com