Pull Request Deployment

Test only YOUR changes

Motivation

- Code changes in a branch should be tested in isolation based on latest master
- Testing of not yet approved changes shouldn’t change shared datasource (DB)
- Multiple code changes shouldn’t share a datasource (DB) simultaniously
- Should be gone after merge without leftover junk

But how do you do it

What we need

- VM or some Server
- Own domain or sub subdomain (xxx.vizzlo.dev or xxx.dev.vizzlo.dev)
- GIT with PullRequests (github, gitlab…)
- PR build job (travis, CircleCi)
- Docker
- Docker registry (Sonatype Nexus, Quay)
- Database dump
- Cleanup scripts

What we do

1. Branch, commit, create PR
2. Build PR, build docker image with commit hast as TAG, push
3. Trigger reload
4. PR deployer work here….
5. Post info to PR as comment
6. After done: remove everything when PR is not open anymore

What is a deployent?

1 Postgres DB

1 Webservice with config

… Whatever other service you need without cyclical dependency like

(**web-service** needs port of **mail-service** and **mail-service** knows port of **web-service**)

What do we need to do

Deployer

- Provide trigger endpoint
- Have access to git service (github)
- Run internal webserver like Caddy as the reverse proxy
- Have Docker Lib connected to local docker.sock

Deployment Jobs Triggered

- Get all Open PRs
- Remove all containers not belonging to open PR
- Find already running container of latest state of PRs and pull all new Images
- Create DB deployment first
- Take port from first deployment and use it for config of next deployment
- Template and configure reverse proxy and restart

How to use Docker best? ---- Labels!!

```go
createdContainer, err := s.cli.ContainerCreate(context.Background(),
  &container.Config{
    ExposedPorts: nat.PortSet{
      nat.Port("5432/tcp"): {}, // Port inside the container
    },
    Env:   []string{"POSTGRES_PASSWORD=root", "POSTGRES_USER=root", "POSTGRES_DB=db"}, // ENV
    Image: "registry.hub.docker.com/library/postgres”,                    // docker image
    Labels: map[string]string{                                            // labels
      LabelKeyRepo:    “repo-name”,                                       // labels
      LabelKeyService: LabelService,                                      // labels
      LabelKeyType:    LabelDb,                                           // labels
      LabelKeyPrNr:    pr.GetPrID(),                                      // labels
    },
  },
  &container.HostConfig{
    PortBindings: nat.PortMap{
      natPort(PgPort): []nat.PortBinding{{HostIP: "0.0.0.0"}}, // Port mapping inside - outside (random port >
32k)
    },
    Mounts: []mount.Mount{                              // volume mount for start up from dump
      {
        Type:   mount.TypeBind,
        Source: s.dbHostDumpDir,
        Target: "/docker-entrypoint-initdb.d",
      },
    },
  },
  nil,
  pr.GetName(),                              // container name
)
```

Create Container (not start):

- Port inside container
- ENVs
- Labels to query later
- PortMapping (Inside -Outside)
- Volume mount for db dump
- Container Name

Start Container

```
err := s.cli.ContainerStart(context.Background(), createdContainer.ID, types.ContainerStartOptions{})
```

Find container and get assigned port

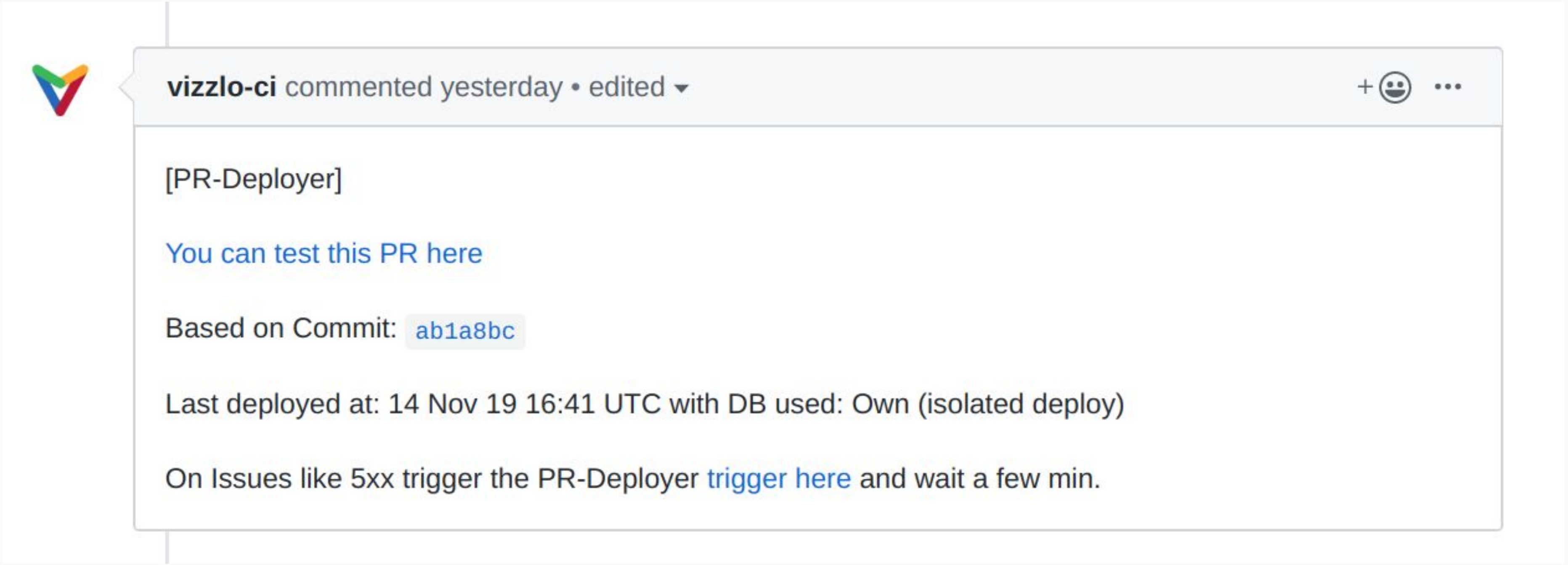
```
f := filters.NewArgs()
f.Add("id", id)
containers, err := s.cli.ContainerList(context.Background(), types.ContainerListOptions{
  Filters: f,
})

for _, port := range containers[0].Ports {
  if int(port.PrivatePort) == 5432 {
     hostPort := int(port.PublicPort) // assigned host port
  }
}
```

Find Container by labels

```
f := filters.NewArgs()
f.Add("label", fmt.Sprintf("%s=%s", LabelKeyRepo, repo))
f.Add("label", fmt.Sprintf("%s=%s", LabelKeyService, service))
f.Add("label", fmt.Sprintf("%s=%s", LabelKeyType, cType))
f.Add("label", fmt.Sprintf("%s=%s", LabelKeyPrNr, pr))
containers, err := s.cli.ContainerList(context.Background(), types.ContainerListOptions{
  Filters: f,
})
```

In the end…



vizzlo-ci commented yesterday • edited

[PR-Deployer]

You can test this PR here

Based on Commit: `ab1a8bc`

Last deployed at: 14 Nov 19 16:41 UTC with DB used: Own (isolated deploy)

On Issues like 5xx trigger the PR-Deployer trigger here and wait a few min.