

Continuous delivery

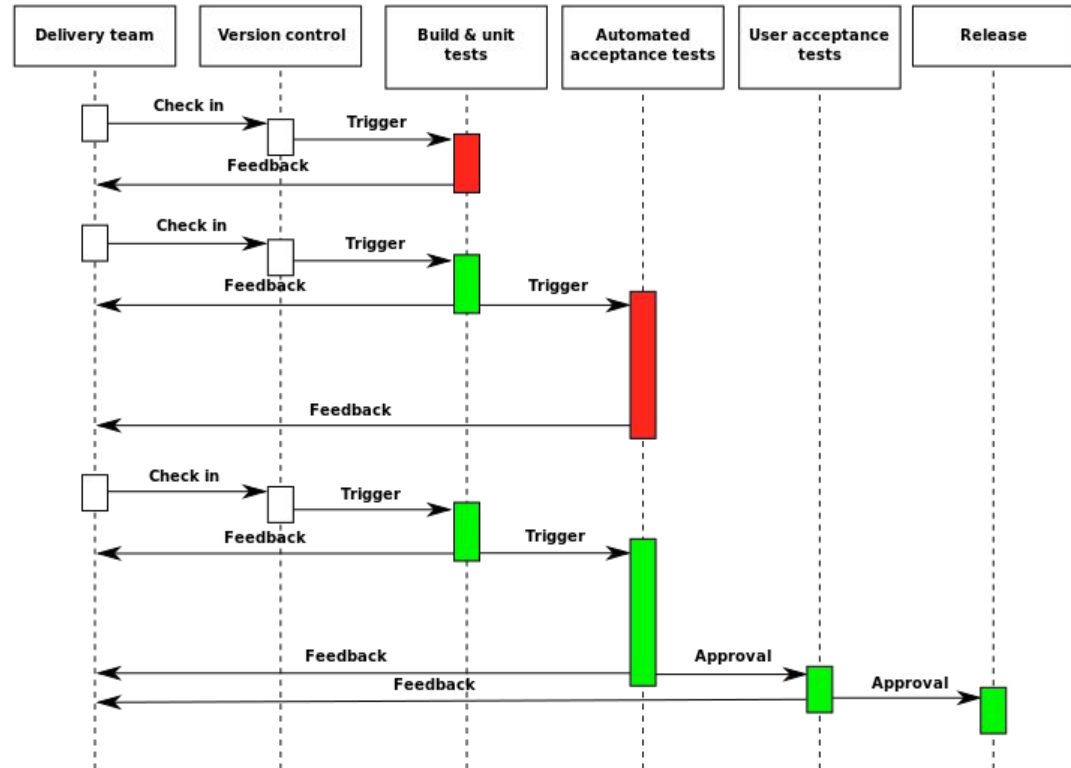
using GitLab, Docker and AWS Elastic Beanstalk

Kamil Rogalski

<https://github.com/drootnar/gitlab-cd-docker-aws-eb-example>

What is continuous delivery?

Approach of software teams to produce software in short cycles, ensuring that the software can be reliably released at any time.



What we want to achieve?

Simple
Flask app

Some tests

What we want to achieve?

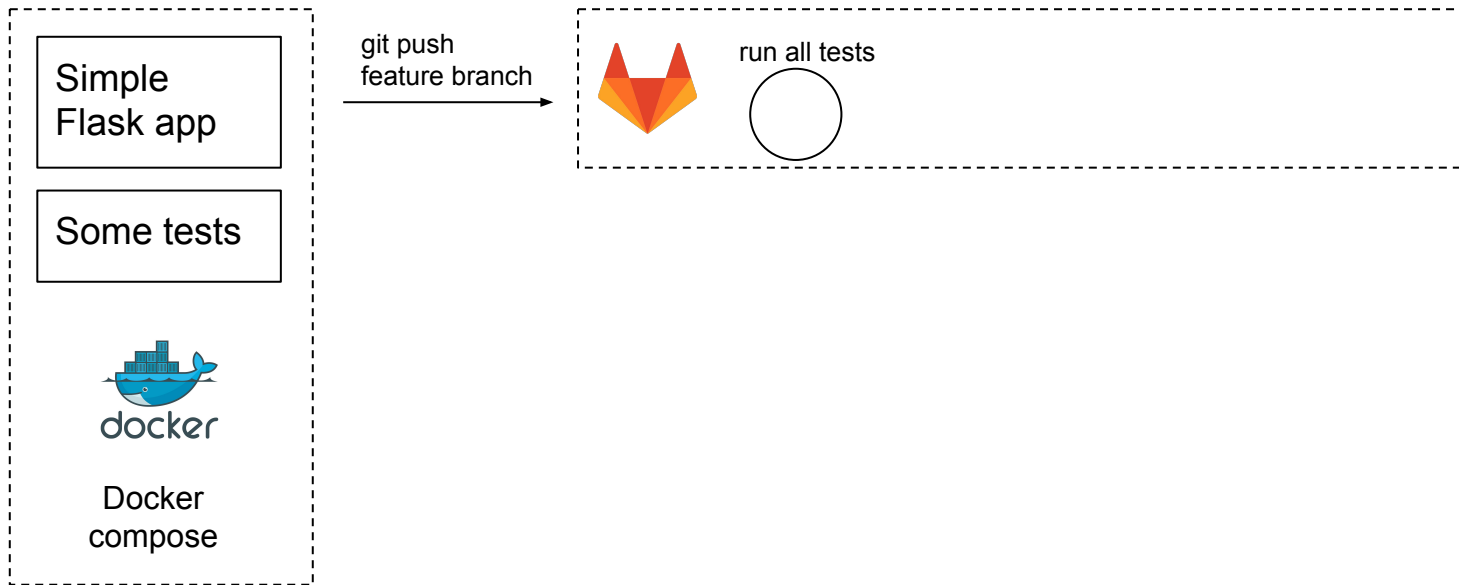
Simple
Flask app

Some tests

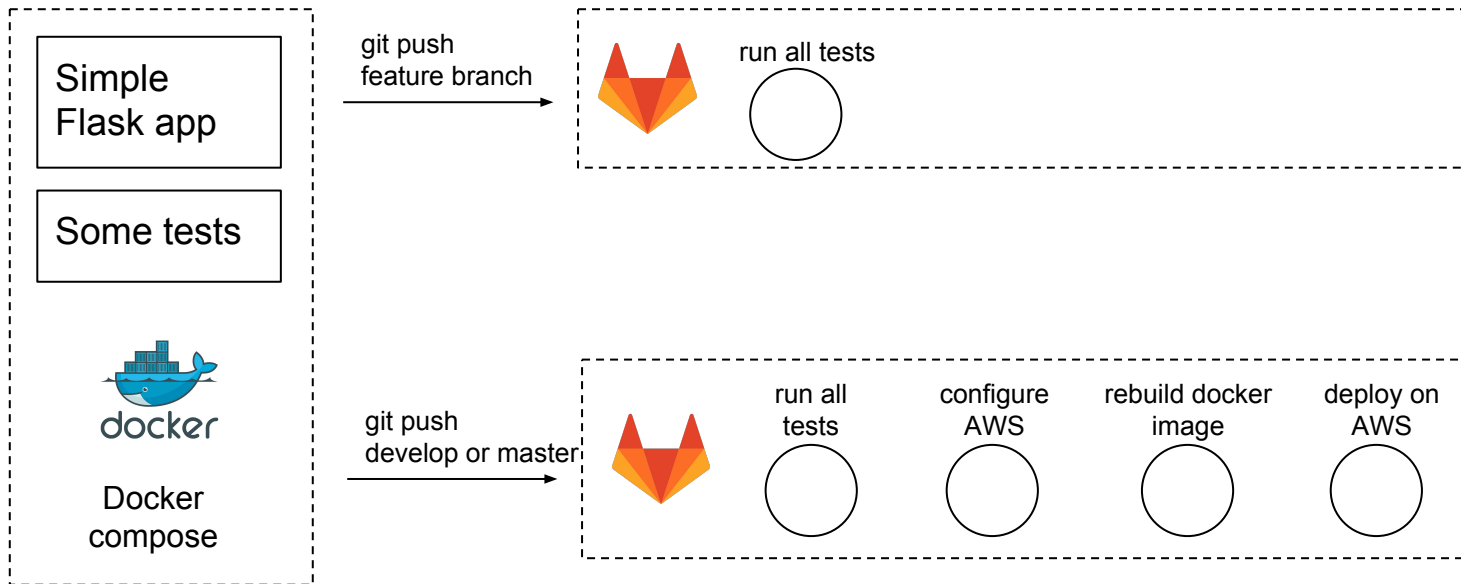


Docker
compose

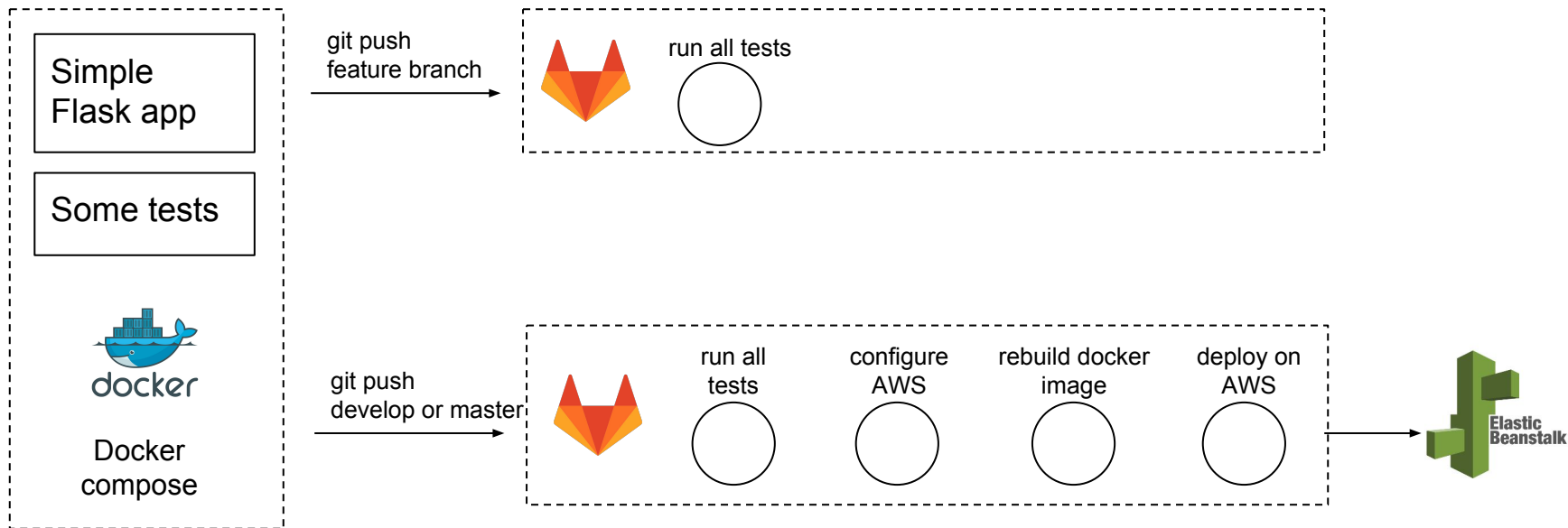
What we want to achieve?



What we want to achieve?



What we want to achieve?



How Flask app looks like?

```
app = Flask(__name__)
```

```
@app.route('/')
```

```
def greetings():
```

```
    name = request.args.get('name', 'anonymous')
```

```
    return "Hello {}".format(name)
```


Some tests using Pytest

```
@pytest.fixture
def client(request):
    client = backend.app.test_client()
    return client

def test_main_endpoint(client):
    response = client.get('/')
    assert b'Hello anonymous!' in response.data

def test_main_endpoint_with_param(client):
    response = client.get('/?name=John')
    assert b'Hello John!' in response.data
```

Step 1: configure Docker & docker-compose

backend-app:

build:

context: ./backend

dockerfile: Dockerfile.dev

restart: unless-stopped

ports:

- "8000:8000"

stdin_open: true

command: bash -c "pip install -r requirements.txt &&
gunicorn --chdir backend backend:app -w 2 --bind 0.0.0.0:8000 --reload"

Step 1: configure Docker & docker-compose

backend-tests:

build:

context: ./backend

dockerfile: Dockerfile.dev

stdin_open: true

command: bash -c "pip install -r requirements.txt -r requirements-test.txt &&
py.test"

Step 2: configure GitLab pipelines

 **passed** Pipeline #7049362 triggered about 3 hours ago by  Kamil


Change dockerfile structure

🕒 4 jobs from `master` in 2 minutes 36 seconds

🔗 `e6ea1a36` ... 

Pipeline Jobs 4

Test

 test-backend

Configure

 configure-aws

Build

 build-docker-i...

Deploy

 eb-deploy

.gitlab-ci.yml

stages:

- test
- configure
- build
- deploy

test-backend:

image: python:3.5.2

stage: test

before_script:

- cd backend
- pip install -r requirements.txt -r requirements-test.txt

script:

- py.test

.gitlab-ci.yml

configure-aws:

image: kowalski0000/awscli:latest

stage: configure

only:

- develop
- master

artifacts:

paths:

- config/

script:

- mkdir config
- printf "[eb-cli]\naws_access_key_id = %s\naws_secret_access_key = %s\n" "\$AWS_ACCESS_KEY_ID" "\$AWS_SECRET_ACCESS_KEY" > config/aws_credentials
- printf "[profile eb-cli]\nregion=eu-central-1\noutput=json" > config/aws_config
- aws ecr **get**-login --region eu-central-1 > config/docker_login

Secret Variables

These variables will be set to environment by the runner.

So you can use them for passwords, secret keys or whatever you want.

The value of the variable can be visible in job log if explicitly asked to do so.





Add a variable

Key

Value

Add new variable

Your variables (2)

Key	Value	
AWS_ACCESS_KEY_ID	*****	 
AWS_SECRET_ACCESS_KEY	*****	 

.gitlab-ci.yml

script:

- source config/docker_login
- bash -ex eb/build_and_push_images.sh \${CI_BUILD_REF_NAME}

```
#!/bin/bash
```

```
TAG=${1-latest}
```

```
REPO=063507218586.dkr.ecr.eu-central-1.amazonaws.com
```

```
function upload() {  
    echo "Building image $2 from directory $1"  
    docker build -t $2:$TAG $1  
    docker tag $2:$TAG $REPO/$2:$TAG  
    docker push $REPO/$2:$TAG  
    echo "Done"  
}
```

```
upload backend cicd/backend
```

.gitlab-ci.yml

eb-deploy:

image: kowalski0000/awscli:latest

stage: deploy

only:

- master

dependencies:

- configure-aws

before_script:

- mkdir ~/.aws/
- cp config/aws_credentials ~/.aws/credentials
- cp config/aws_config ~/.aws/config

script:

- cd eb/app
- cat Dockerrun.aws.json.template | TAG=\${CI_BUILD_REF_NAME} envsubst > Dockerrun.aws.json
- git checkout -B "\$CI_BUILD_REF_NAME" "\$CI_BUILD_REF"
- git add Dockerrun.aws.json
- eb deploy --staged

Step 3: Configure Gitlab Runner

- Shared vs specific runners
- In order to build images you need to run own Gitlab runners
- Gitlab runner as docker container: <https://docs.gitlab.com/runner/install/docker.html>

```
root@ip-172-31-20-213:~# docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
PORTS	NAMES			
3776b439be70	312cc2d315a3	"sh -c 'if [-x /u..."	12 seconds ago	Up 7 seconds
runner-729af211-project-2822304-concurrent-0-build				
77da8e2be8ec	quay.io/gitlab/gitlab-runner-docker-cleanup	"go-wrapper run"	8 days ago	Up 8 days
root_gitlab-runner-docker-cleanup_1				
3d339ff53eaf	gitlab/gitlab-runner	"/usr/bin/dumb-ini..."	8 days ago	Up 8 days
root_gitlab-runner_1				

Step 4: configure EC2 Container Service

Services

Resource Groups

kamil.rogalski @ 0635-0721-8586

Frankfurt

Support

Amazon ECS

Clusters

Task Definitions

Repositories

< All repositories : cicd/backend

Repository ARN

arn:aws:ecr:eu-central-1:063507218586:repository/cicd/backend

Repository URI

063507218586.dkr.ecr.eu-central-1.amazonaws.com/cicd/backend

View Push Commands

Images

Permissions

Amazon ECR [limits](#) the number of images to 1,000 per repository. [Request a limit increase.](#)

Image sizes may appear compressed. [Learn more](#)

Delete

Last updated on March 17, 2017 11:19:34 AM (0m ago)

Filter in this page

Tag Status: All

< 1-5 > Page size 100

<input type="checkbox"/>	Image tags	Digest	Size (MiB)	Pushed at
<input type="checkbox"/>		sha256:e3f194f5f7c05e0fb2afbc46d2bd4d60289825...	257.65	2017-03-16 09:13:00 +0100
<input type="checkbox"/>	master	view all sha256:aa07690f26a05e6fe5ac03a3a9f689de27a07...	257.65	2017-03-16 09:18:37 +0100
<input type="checkbox"/>		sha256:260cf55b3f3f198d841c754dc7e166adbe646...	257.65	2017-03-16 08:53:24 +0100
<input type="checkbox"/>		sha256:12744939225decdb065b2ff0a9a2212c8b6c...	257.65	2017-03-16 08:36:04 +0100
<input type="checkbox"/>		sha256:fc8b200ccd2a2d64261f31541c1093c995008...	257.65	2017-03-15 14:47:52 +0100

Step 4: configure EC2 Container Service

Services ▾

Resource Groups ▾

★

🔔

kamil.rogalski @ 0635-0721-8586 ▾

Frankfurt ▾

Support ▾

Amazon ECS

Clusters

Task Definitions

Repositories

< All repositories : cicd/backend

Repository ARN arn:aws:ecr:eu-central-1:063507218586:repository/cicd/backend

Repository URI **063507218586.dkr.ecr.eu-central-1.amazonaws.com/cicd/backend**

View Push Commands

Images Permissions

Amazon ECR [limits](#) the number of images to 1,000 per repository. [Request a limit increase.](#)

Image sizes may appear compressed. [Learn more](#)

Delete

Last updated on March 17, 2017 11:19:34 AM (0m ago) ↻

Filter in this page

Tag Status: All ▾

< 1-5 > Page size 100 ▾

<input type="checkbox"/>	Image tags	Digest	Size (MiB) ▾	Pushed at ▾
<input type="checkbox"/>		sha256:e3f194f5f7c05e0fb2afbc46d2bd4d60289825...	257.65	2017-03-16 09:13:00 +0100
<input type="checkbox"/>	master	view all sha256:aa07690f26a05e6fe5ac03a3a9f689de27a07...	257.65	2017-03-16 09:18:37 +0100
<input type="checkbox"/>		sha256:260cf55b3f3f198d841c754dc7e166adbe646...	257.65	2017-03-16 08:53:24 +0100
<input type="checkbox"/>		sha256:12744939225decdb065b2ff0a9a2212c8b6c...	257.65	2017-03-16 08:36:04 +0100
<input type="checkbox"/>		sha256:fc8b200ccd2a2d64261f31541c1093c995008...	257.65	2017-03-15 14:47:52 +0100

Step 4: configure AWS ECS and Beanstalk

Services ▾

Resource Groups ▾

★

🔔

kamil.rogalski @ 0635-0721-8586 ▾

Frankfurt ▾

Support ▾

Elastic Beanstalk

botxo ▾

cicd ▾

gitlab-ci ▾

gitlab-runner ▾

py-fb-bot ▾

Create New Application

All Applications > cicd > cicd-production (Environment ID: e-mp8m89pxzy, URL: cicd-production.eu-central-1.elasticbeanstalk.com)

Actions ▾

Dashboard

Configuration

Logs

Health

Monitoring

Alarms


Managed Updates

Events

Tags

Overview

Refresh



Health


Ok

Causes

Running Version

app-5345-170316_082328-stage-170316_082328

Upload and Deploy



Configuration

64bit Amazon Linux 2016.09
v2.5.0 running Multi-container
Docker 1.12.6 (Generic)

Change

Recent Events

Show All

Time	Type	Details
2017-03-16 09:26:00 UTC+0100	INFO	Environment health has transitioned from Info to Ok. Application update completed 47 seconds ago and took 54 seconds.
2017-03-16 09:24:27 UTC+0100	INFO	Environment update completed successfully.
2017-03-16 09:24:27 UTC+0100	INFO	New application version was deployed to running EC2 instances.
2017-03-16 09:24:22 UTC+0100	INFO	ECS task: arn:aws:ecs:eu-central-1:063507218586:task/40f1fa6e-76ff-4afe-b903-2374ea00ee94 is RUNNING.
2017-03-16 09:24:18 UTC+0100	INFO	Starting new ECS task with awseb-cicd-production-mp8m89pxzy:9.

Step 4: configure AWS ECS and Beanstalk

The screenshot displays the AWS Management Console interface for configuring an Elastic Beanstalk application. The top navigation bar shows the user is logged in as 'kamil.rogalski @ 0635-0721-8586' in the 'Frankfurt' region. The breadcrumb trail indicates the path: 'All Applications > cicd > cicd-production'. The environment details are: (Environment ID: e-mp8m89pczy, URL: cicd-production.eu-central-1.elasticbeanstalk.com).

The left sidebar contains a navigation menu with the following items: Dashboard, Configuration (highlighted), Logs, Health, Monitoring, Alarms, Managed Updates, Events, and Tags.

The main content area is titled 'Web Tier' and contains six configuration panels, each with a settings gear icon:

- Scaling**: Environment type: Single instance; Custom Availability Zones: blank.
- Instances**: Instance type: t2.micro; Availability Zones: Any.
- Notifications**: Notifications: Off.
- Software Configuration**: Log publication: Off; Log streaming: disabled.
- Updates and Deployments**: Rolling updates are disabled.
- Health**: Application health check URL: blank; Health reporting: Enhanced.
- Managed Updates**: Managed updates are disabled.

Step 4: configure AWS ECS and Beanstalk

```
{
  "AWSEBDockerrunVersion": 2,
  "containerDefinitions": [
    {
      "essential": true,
      "image": "063507218586.dkr.ecr.eu-central-1.amazonaws.com/cicd/backend:${TAG}",
      "memory": 128,
      "mountPoints": [],
      "name": "backend",
      "portMappings": [
        {
          "containerPort": 8000,
          "hostPort": 80
        }
      ]
    }
  ]
}
```

... come back to last stage of pipeline

```
cd eb/app
cat Dockerrun.aws.json.template | TAG=${CI_BUILD_REF_NAME} envsubst > Dockerrun.aws.json
git checkout -B "$CI_BUILD_REF_NAME" "$CI_BUILD_REF"
git add Dockerrun.aws.json
eb deploy --staged
```

Result

- automatic deploy
- continuous integration
- continuous delivery
- separate AWS configuration and code
- scalable AWS resources