Continuous delivery

using GitLab, Docker and AWS Elastic Beanstalk

What is continuous delivery?

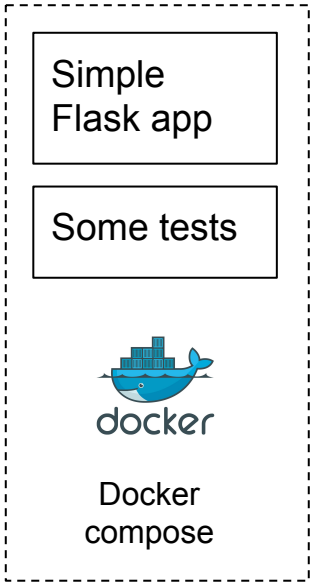
Approach of software teams to produce software in short cycles.

What we want to achieve?

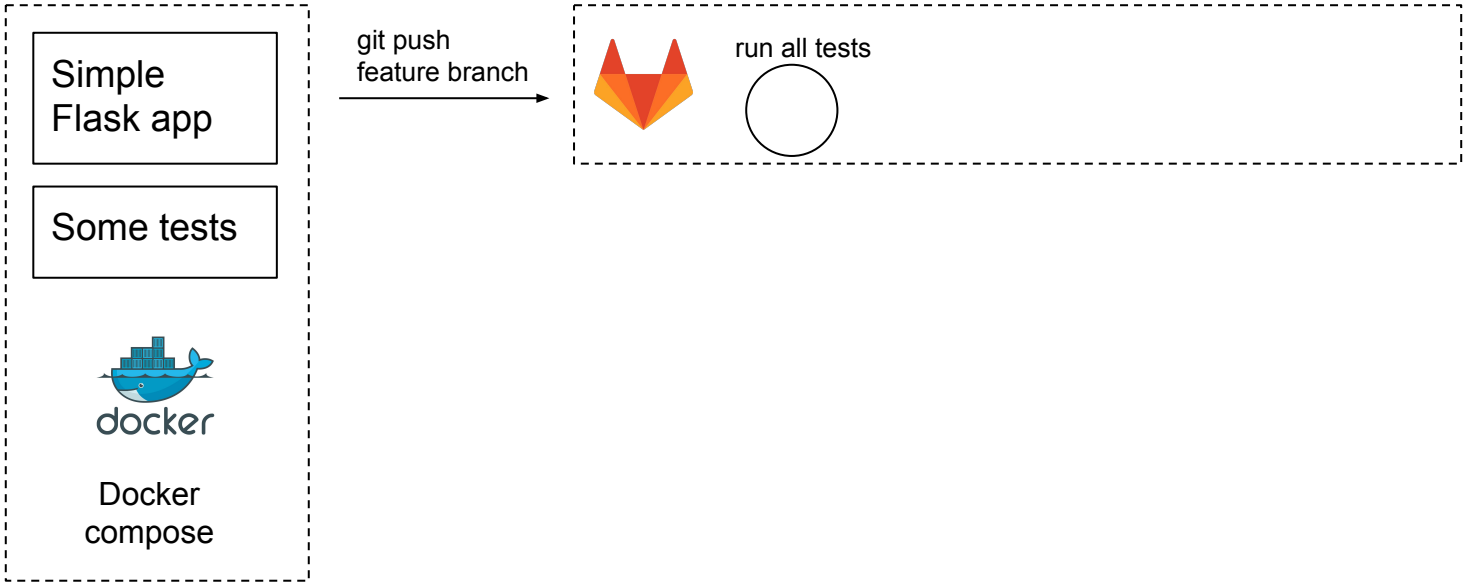
Simple
Flask app

Some tests

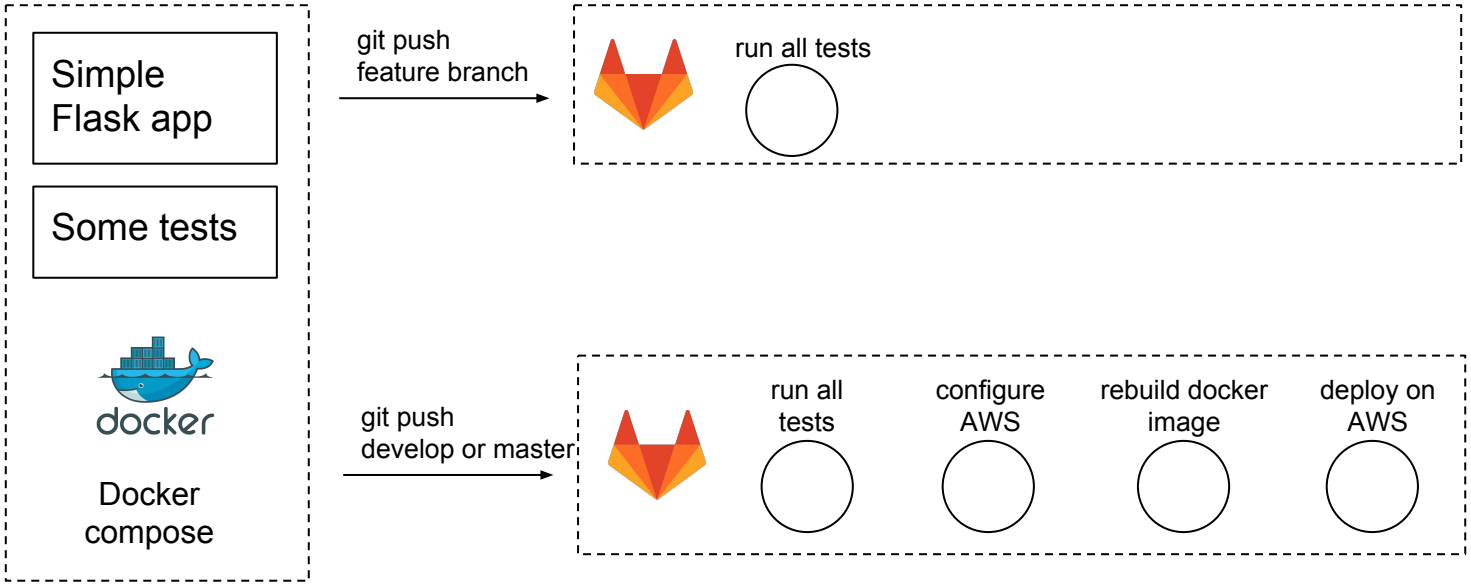
What we want to achieve?



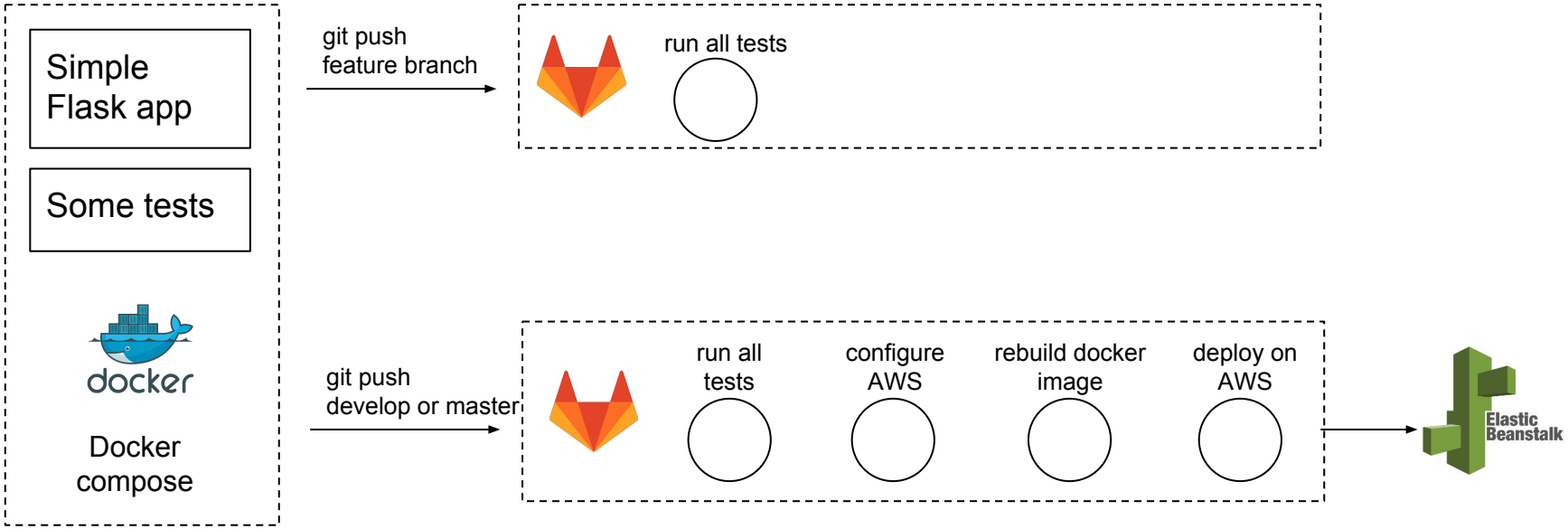
What we want to achieve?



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
```

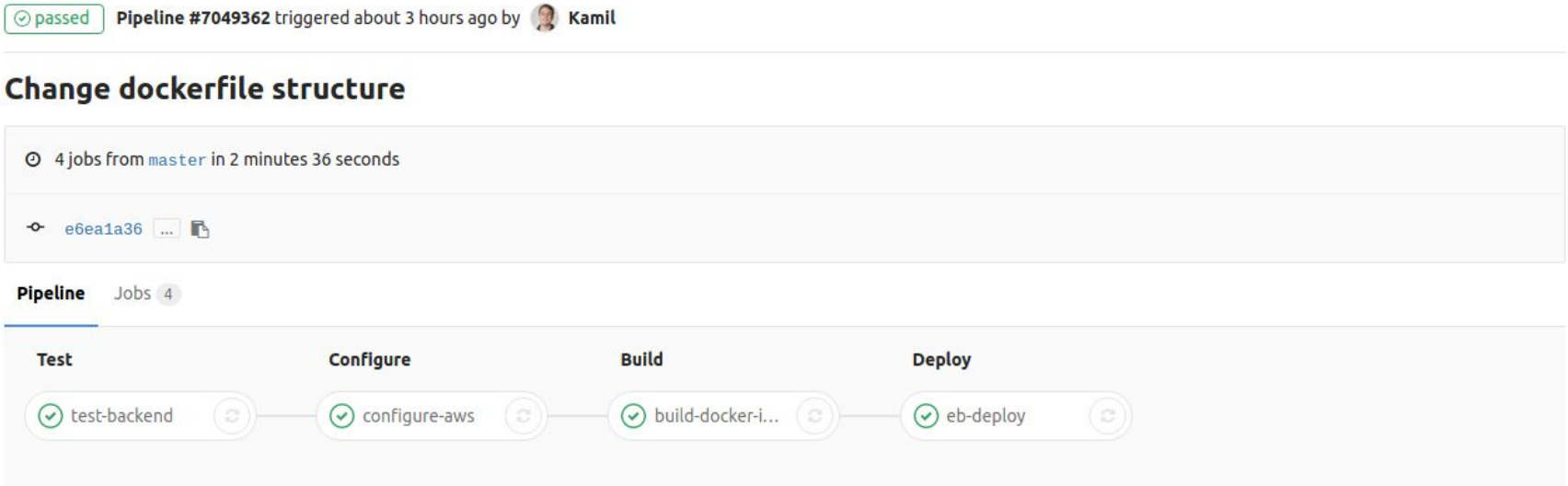
Step 1: Configure Docker & docker-compose

```
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



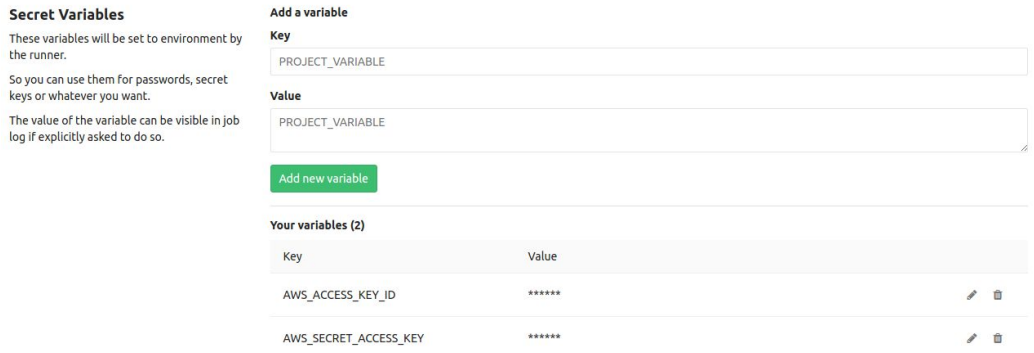
.gitlab-ci.yml

```
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
```



.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

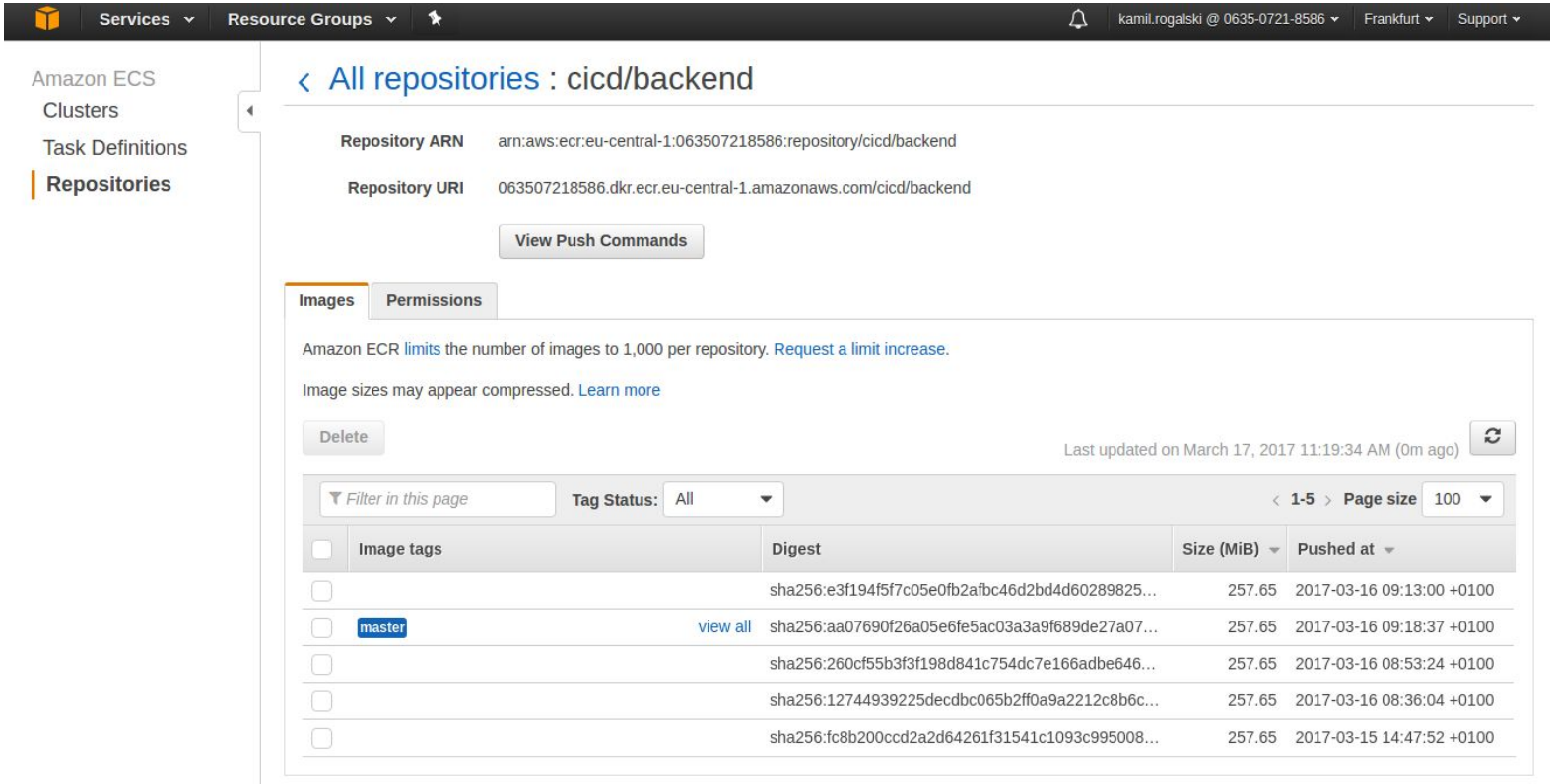
```
eb-deploy:
  image: kowalski0000/awscli:latest
  stage: deploy
  only:
    - master
  dependencies:
    - configure-aws
  before_script:
    - mkdir ~/.aws/
    - cp config/aws_credentials ~/.aws/credentials
    - cp config/aws_config ~/.aws/config
  script:
    - cd eb/app
    - cat Dockerrun.aws.json.template | 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

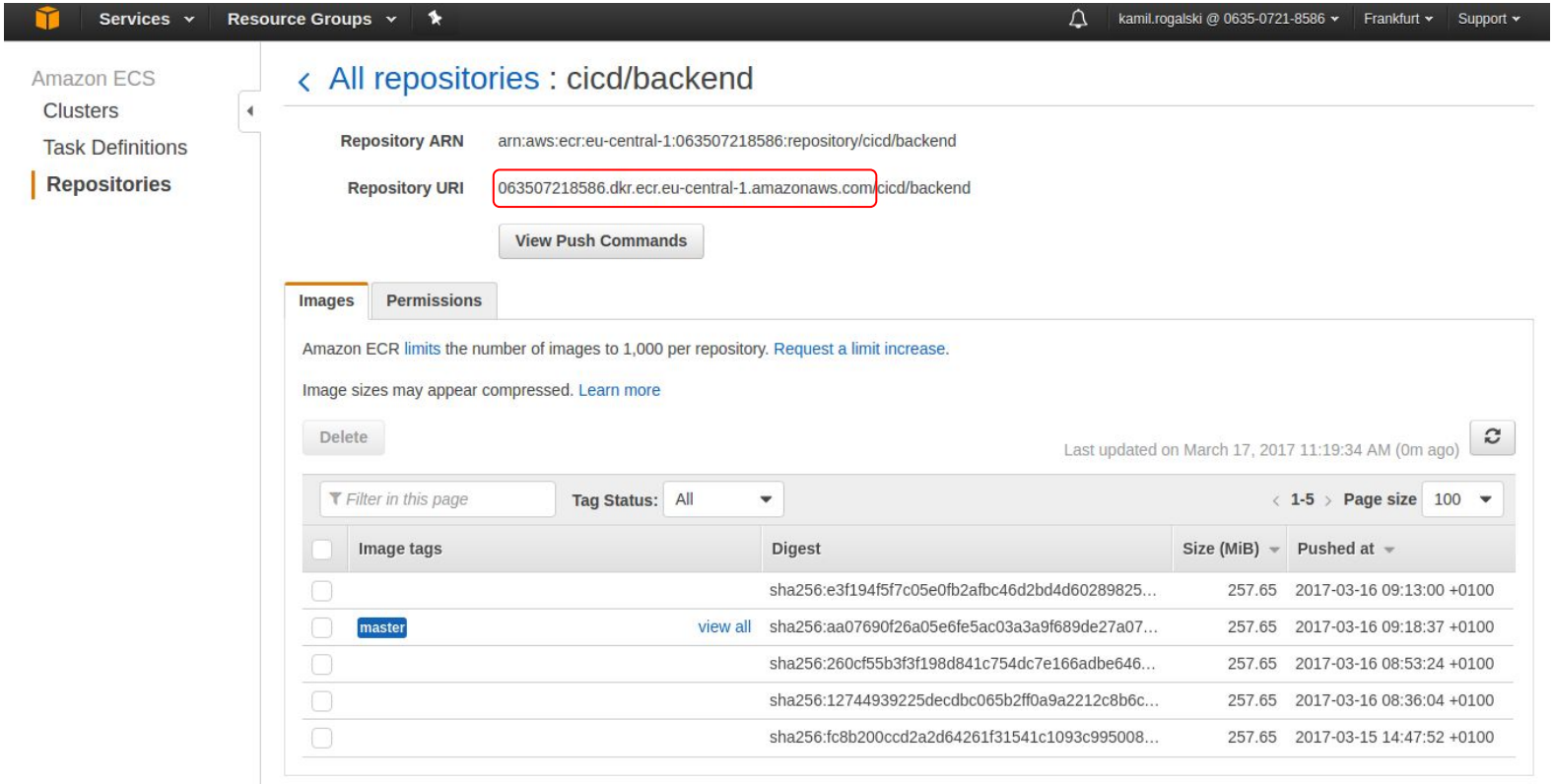
- Shared vs specific runners
- In order to build images you need to run own Gitlab runners
- Gitlab runner as docker container: https://docs.gitlab.com/runner/install/docker.html

```
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
```

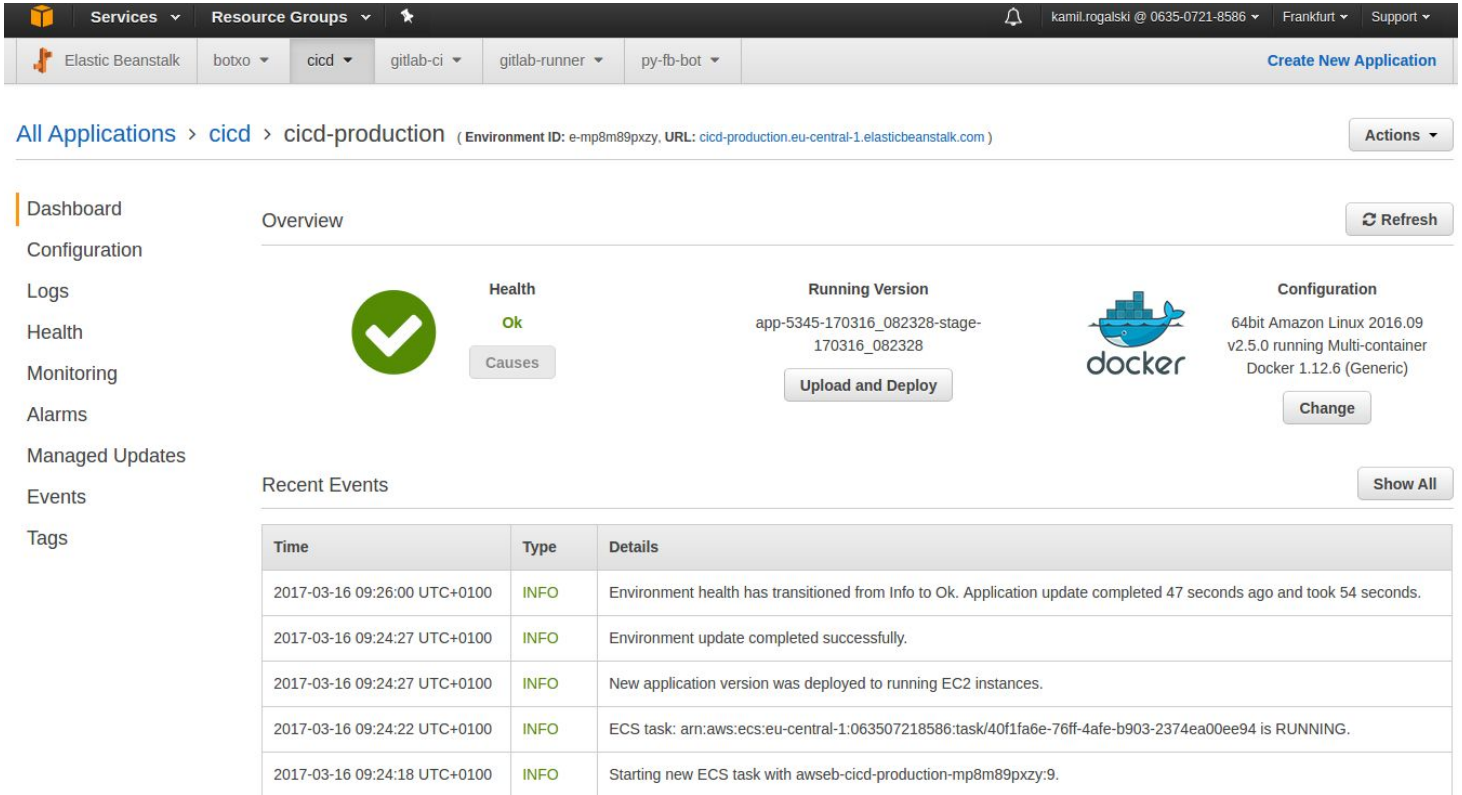
Step 4: configure EC2 Container Service



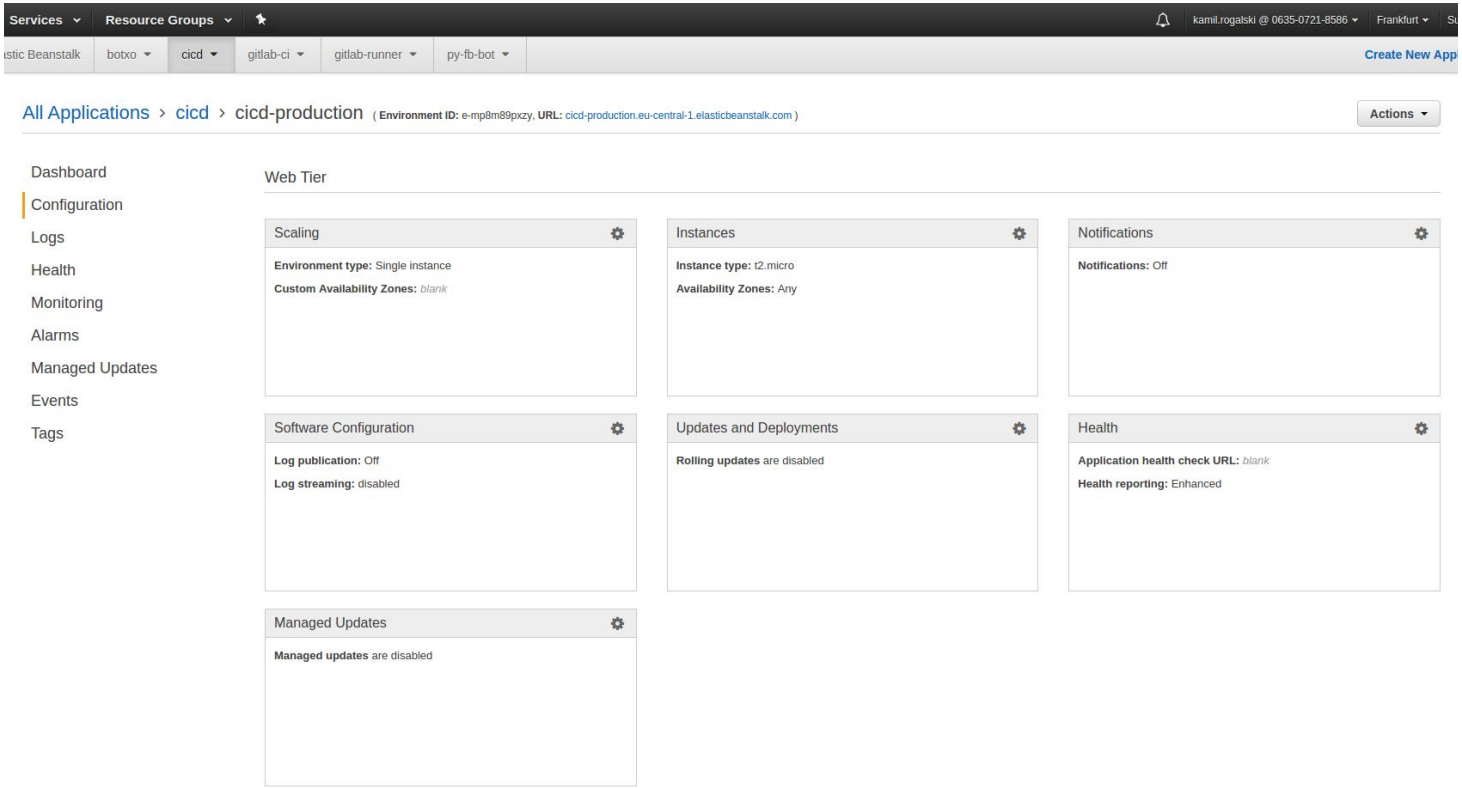
Step 4: configure EC2 Container Service



Step 4: configure AWS ECS and Beanstalk



Step 4: configure AWS ECS and Beanstalk



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
                }
            ]
        }
    ]
}
```

```
eb deploy --staged
```

Result

- automatic deploy
- continuous integration
- scalable CI/CD machinery
- separate AWS configuration and code