EPAM Cloud&DevOps Fundamentals Autumn 2022

Create infrastructure and deploy website «Python Bytes Club's Blog»

Mykhailo Solomashenko

Kharkiv, Ukraine

Individual entrepreneur

Education:

• Higher technical in the specialty "Electric drive and automation of PU", Electrical Engineer, NTU "Kharkiv Polytechnic Institute".

Self-education:

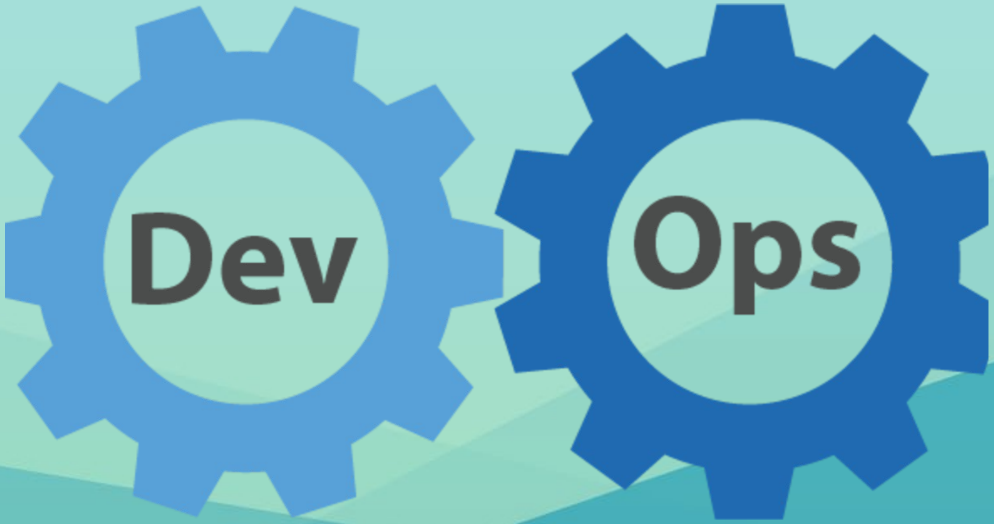
• Course "Cloud&DevOps Fundamentals" EPAM Systems.
• Course for beginners "IT Fundamentals" EPAM Systems.
• QAP at SkillFactory school.
• Some IT-courses at Prometheus.
• "Accounting consultant for small and medium businesses", Kharkiv National University of Economics.

Product build speed is an important competitive advantage in software development.

What used to be done in months is now done in a matter of days without loss of quality.

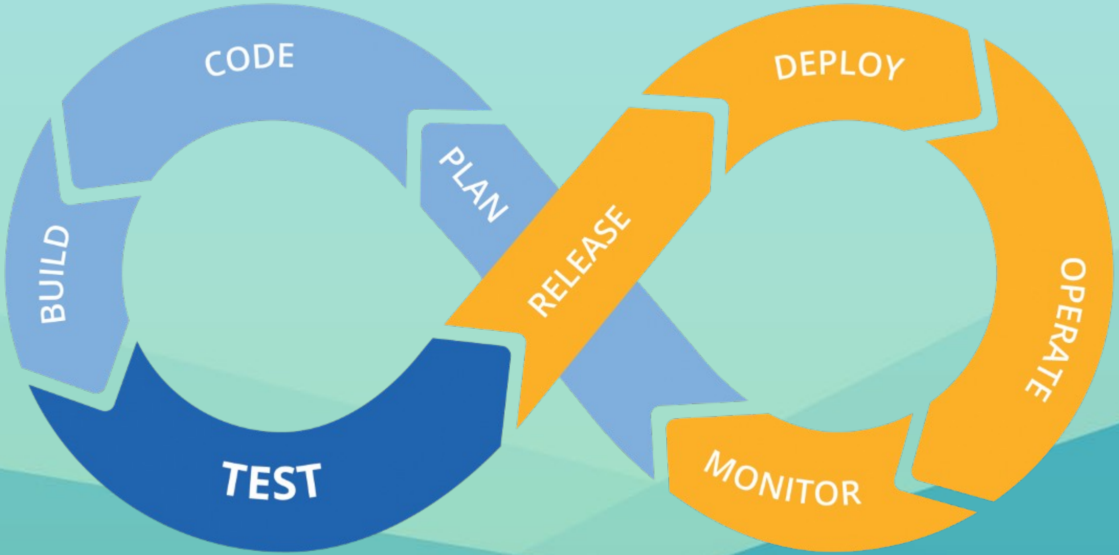
The path to faster releases is through automation and **CI/CD** implementation.

CI/CD is one of the DevOps practices that allows developers to deploy software changes more often and more reliably, minimize errors, increase build rates and improve the quality of the product being developed.



CI, or *continuous integration*, is the process of continuous software development with integration into the main branch. Automatically collects software, tests it and notifies you if something goes wrong.

CD, or *continuous delivery*, is the process of continuously delivering software to the consumer. Ensures the development of the project in small parts and ensures that it can be released at any time without additional manual checks.



CI/CD required:

1. To save time through the use of code and the rapid deployment of projects.
2. To get the expected result from the deployment.
3. To minimize the resulting errors.
4. That the project would not depend on the environment.
5. To carry out easy migration.

Result - acceleration of terms of an output of a product on the market.

For the implementation of the final project, a web app was chosen - the blog of the Internet club "Python Bytes", written on the Django Python framework.

The website with blog is deployed on AWS.

The system is also deployed there that builds the project into a Docker container and deploys the container on an EC2 instance.

The project can be divided into 4 stages:

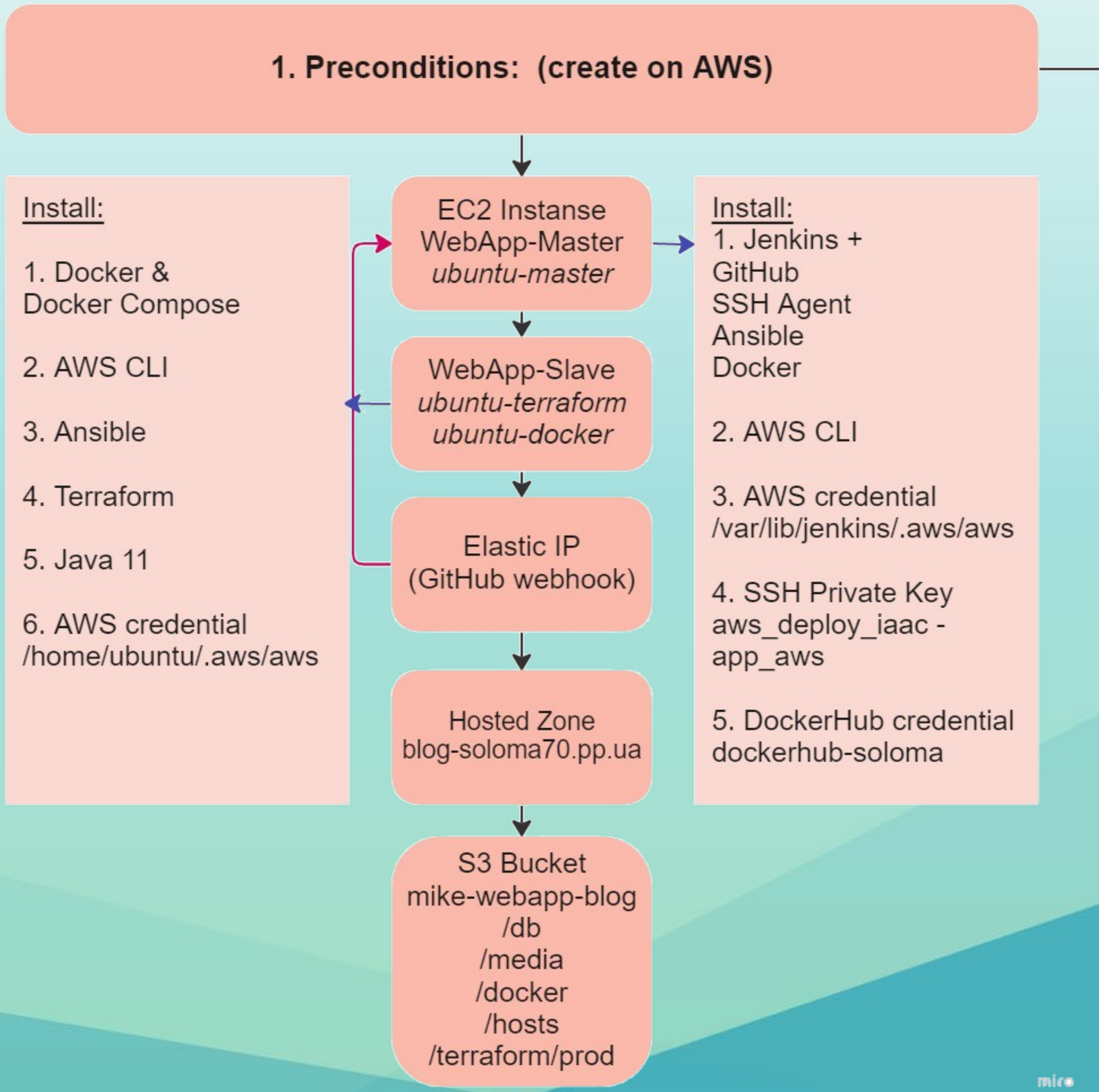
I. Preconditions stage

II. Deploying the infrastructure on AWS
Installing the necessary software

III. Initial installation of the site and database

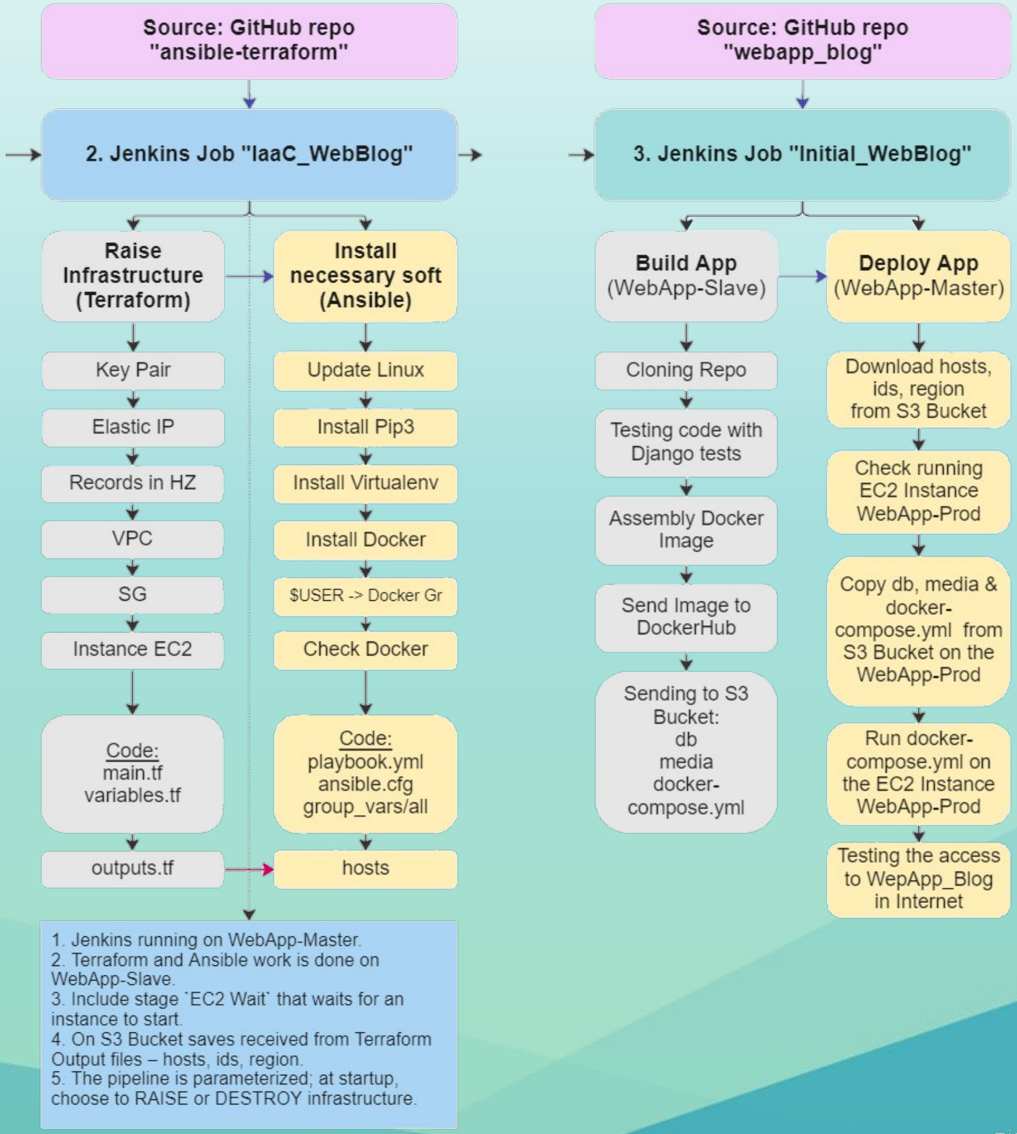
IV. CI/CD pipeline implements an automated code assembly system with new web app features

At the Preconditions stage, the tasks of deploying the necessary infrastructure in AWS, installing software related to the implementation of the pipeline: Docker, Terraform, Ansible and the system that will manage the pipeline - Jenkins are solved.

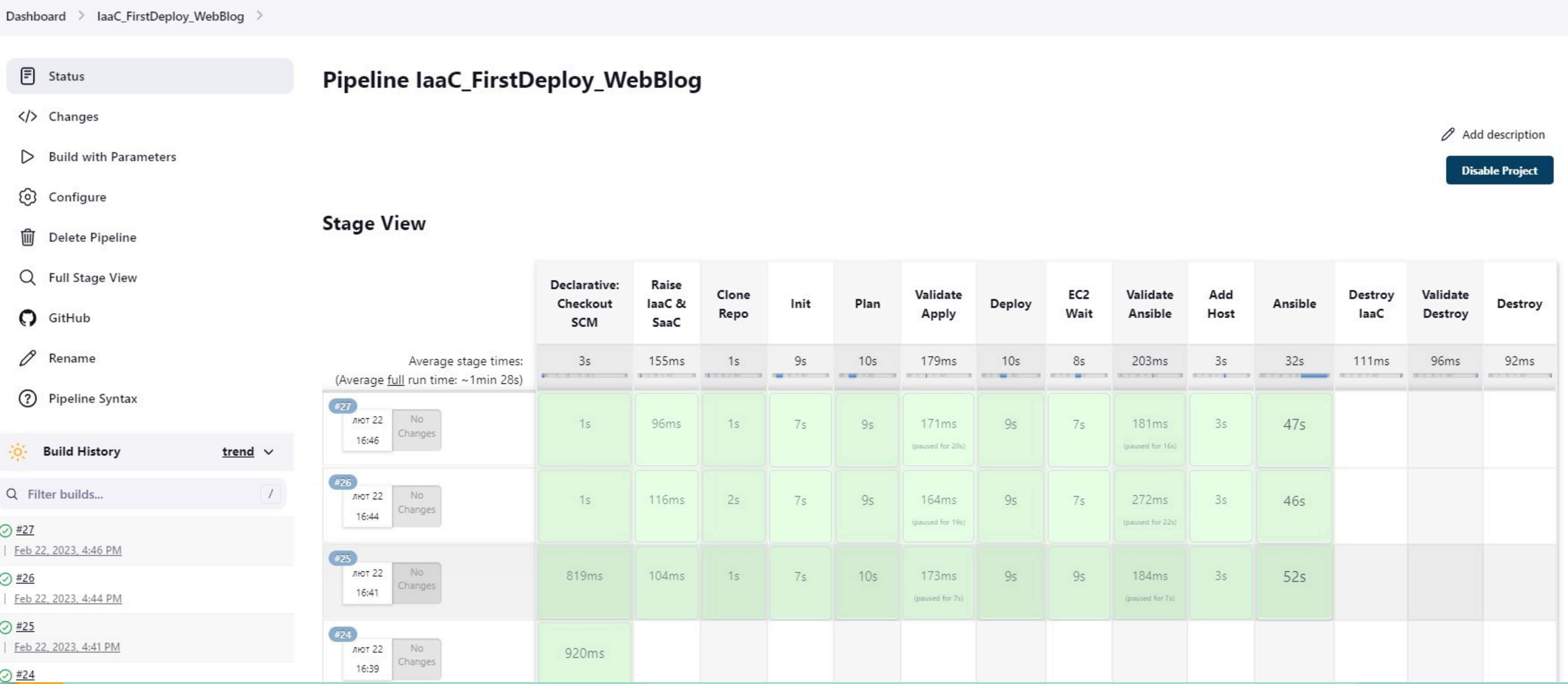


1. Preconditions: (create on AWS)
Install:
1. Docker & Docker Compose
2. AWS CLI
3. Ansible
4. Terraform
5. Java 11
6. AWS credential /home/ubuntu/.aws/aws
EC2 Instanse WebApp-Master ubuntu-master
WebApp-Slave ubuntu-terraform ubuntu-docker
Elastic IP (GitHub webhook)
Hosted Zone blog-soloma70.pp.ua
S3 Bucket mike-webapp-blog /db /media /docker /hosts /terraform/prod
Install:
1. Jenkins + GitHub SSH Agent Ansible Docker
2. AWS CLI
3. AWS credential /var/lib/jenkins/.aws/aws
4. SSH Private Key aws_deploy_iaac - app_aws
5. DockerHub credential dockerhub-soloma
miro

At stages II and III, the tasks of deploying the infrastructure on which the site will operate, installing the necessary software, and deploying the site on an EC2 instance in AWS are solved.


Source: GitHub repo "ansible-terraform"
2. Jenkins Job "IaaC_WebBlog"
Raise Infrastructure (Terraform)
Key Pair
Elastic IP
Records in HZ
VPC
SG
Instance EC2
Code: main.tf variables.tf
outputs.tf
Install necessary soft (Ansible)
Update Linux
Install Pip3
Install Virtualenv
Install Docker
$USER -> Docker Gr
Check Docker
Code: playbook.yml ansible.cfg group_vars/all
hosts
Source: GitHub repo "webapp_blog"
3. Jenkins Job "Initial_WebBlog"
Build App (WebApp-Slave)
Cloning Repo
Testing code with Django tests
Assembly Docker Image
Send Image to DockerHub
Sending to S3 Bucket: db media docker-compose.yml
Deploy App (WebApp-Master)
Download hosts, ids, region from S3 Bucket
Check running EC2 Instance WebApp-Prod
Copy db, media & docker-compose.yml from S3 Bucket on the WebApp-Prod
Run docker-compose.yml on the EC2 Instance WebApp-Prod
Testing the access to WepApp_Blog in Internet

1. Jenkins running on WebApp-Master.
2. Terraform and Ansible work is done on WebApp-Slave.
3. Include stage `EC2 Wait` that waits for an instance to start.
4. On S3 Bucket saves received from Terraform Output files – hosts, ids, region.
5. The pipeline is parameterized; at startup, choose to RAISE or DESTROY infrastructure.


Dashboard > IaaC_FirstDeploy_WebBlog >

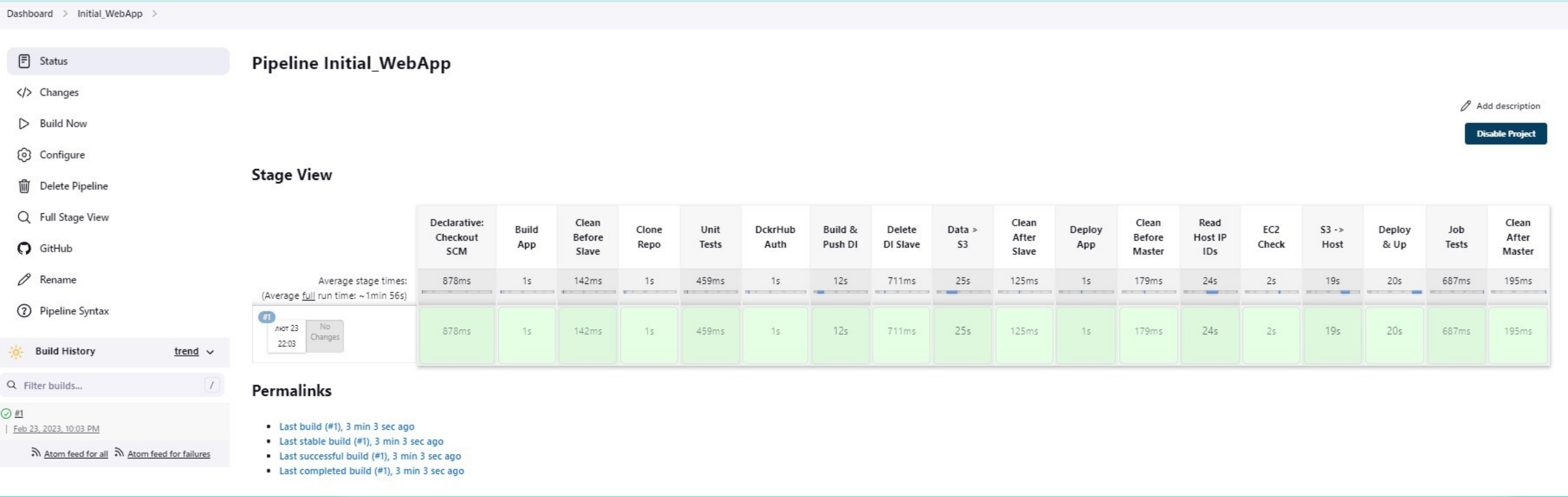
Pipeline IaaC_FirstDeploy_WebBlog

Stage View

	Declarative: Checkout SCM	Raise IaaC & SaaC	Clone Repo	Init	Plan	Validate Apply	Deploy	EC2 Wait	Validate Ansible	Add Host	Ansible	Destroy IaaC	Validate Destroy	Destroy
Average stage times: (Average full run time: ~1min 28s)	3s	155ms	1s	9s	10s	179ms	10s	8s	203ms	3s	32s	111ms	96ms	92ms
#27 лют 22 16:46 No Changes	1s	96ms	1s	7s	9s	171ms (paused for 20s)	9s	7s	181ms (paused for 16s)	3s	47s			
#26 лют 22 16:44 No Changes	1s	116ms	2s	7s	9s	164ms (paused for 19s)	9s	7s	272ms (paused for 22s)	3s	46s			
#25 лют 22 16:41 No Changes	819ms	104ms	1s	7s	10s	173ms (paused for 7s)	9s	9s	184ms (paused for 7s)	3s	52s			
#24 лют 22 16:39 No Changes	920ms													

Build History:

- #27 Feb 22, 2023, 4:46 PM
- #26 Feb 22, 2023, 4:44 PM
- #25 Feb 22, 2023, 4:41 PM
- #24



Dashboard > Initial_WebApp >

Pipeline Initial_WebApp

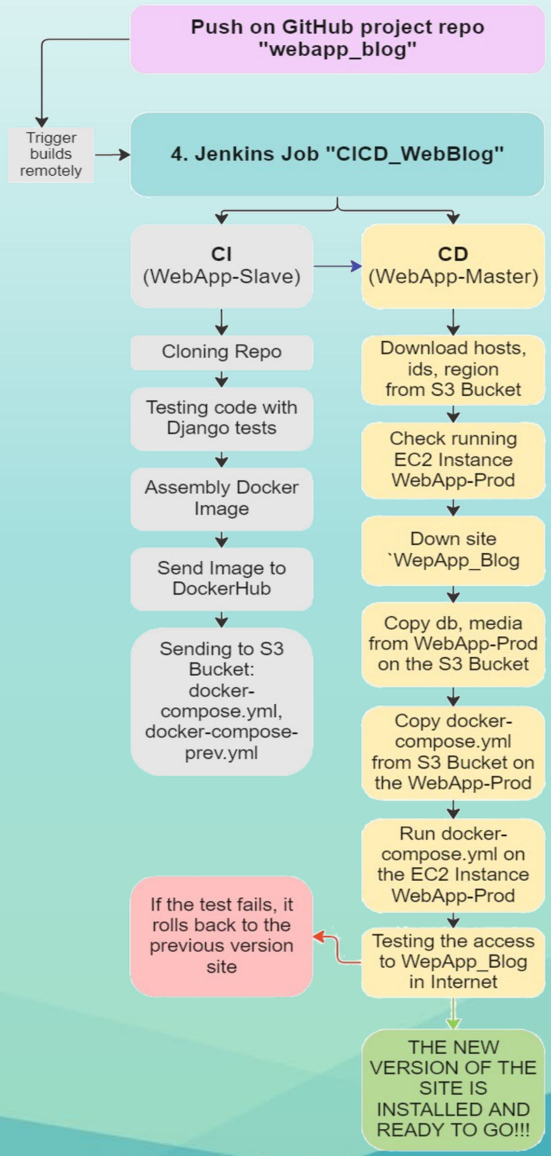
Stage View

	Declarative: Checkout SCM	Build App	Clean Before Slave	Clone Repo	Unit Tests	DckrHub Auth	Build & Push DI	Delete DI Slave	Data > S3	Clean After Slave	Deploy App	Clean Before Master	Read Host IP IDs	EC2 Check	S3 -> Host	Deploy & Up	Job Tests	Clean After Master
Average stage times: (Average full run time: ~1min 56s)	878ms	1s	142ms	1s	459ms	1s	12s	711ms	25s	125ms	1s	179ms	24s	2s	19s	20s	687ms	195ms
#1 лют 23 22:03 No Changes	878ms	1s	142ms	1s	459ms	1s	12s	711ms	25s	125ms	1s	179ms	24s	2s	19s	20s	687ms	195ms

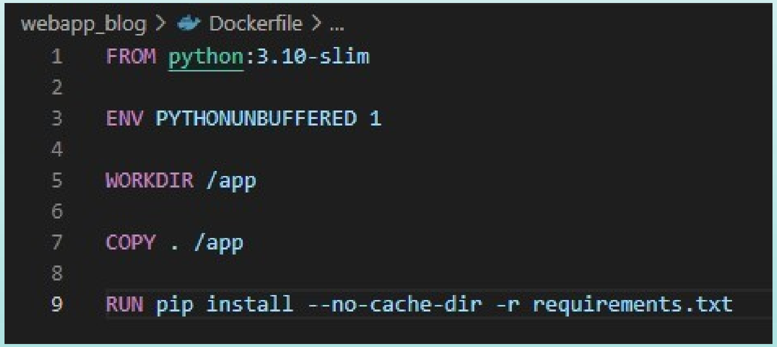
Permalinks

- Last build (#1), 3 min 3 sec ago
- Last stable build (#1), 3 min 3 sec ago
- Last successful build (#1), 3 min 3 sec ago
- Last completed build (#1), 3 min 3 sec ago

At the IV stage, the job of building a CI/CD pipeline is solved with changes in the source code associated with the implementation of various features.


Push on GitHub project repo "webapp_blog"
Trigger builds remotely
4. Jenkins Job "CICD_WebBlog"
CI (WebApp-Slave)
CD (WebApp-Master)
Cloning Repo
Testing code with Django tests
Assembly Docker Image
Send Image to DockerHub
Sending to S3 Bucket: docker-compose.yml, docker-compose-prev.yml
Download hosts, ids, region from S3 Bucket
Check running EC2 Instance WebApp-Prod
Down site `WepApp_Blog
Copy db, media from WebApp-Prod on the S3 Bucket
Copy docker-compose.yml from S3 Bucket on the WebApp-Prod
Run docker-compose.yml on the EC2 Instance WebApp-Prod
Testing the access to WepApp_Blog in Internet
If the test fails, it rolls back to the previous version site
THE NEW VERSION OF THE SITE IS INSTALLED AND READY TO GO!!!

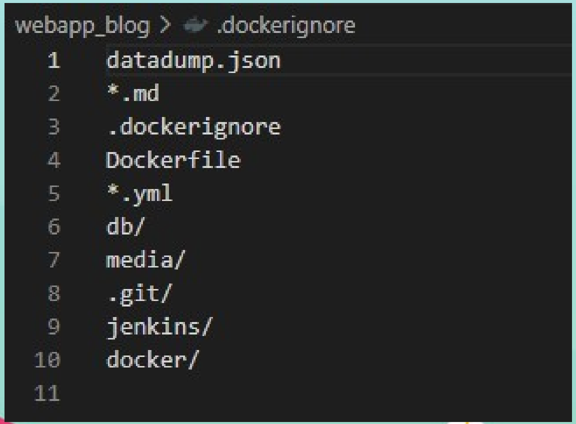
Dockerfile



webapp_blog > Dockerfile > ...

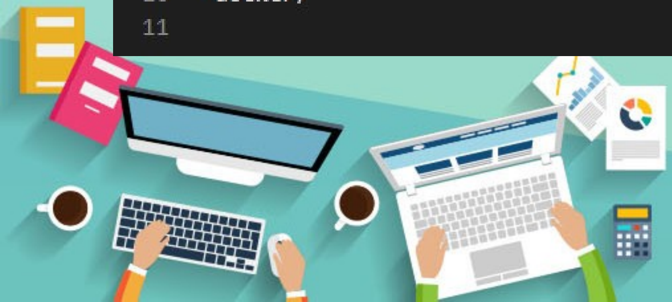
```
FROM python:3.10-slim

ENV PYTHONUNBUFFERED 1

WORKDIR /app

COPY . /app

RUN pip install --no-cache-dir -r requirements.txt
```

.dockerignore

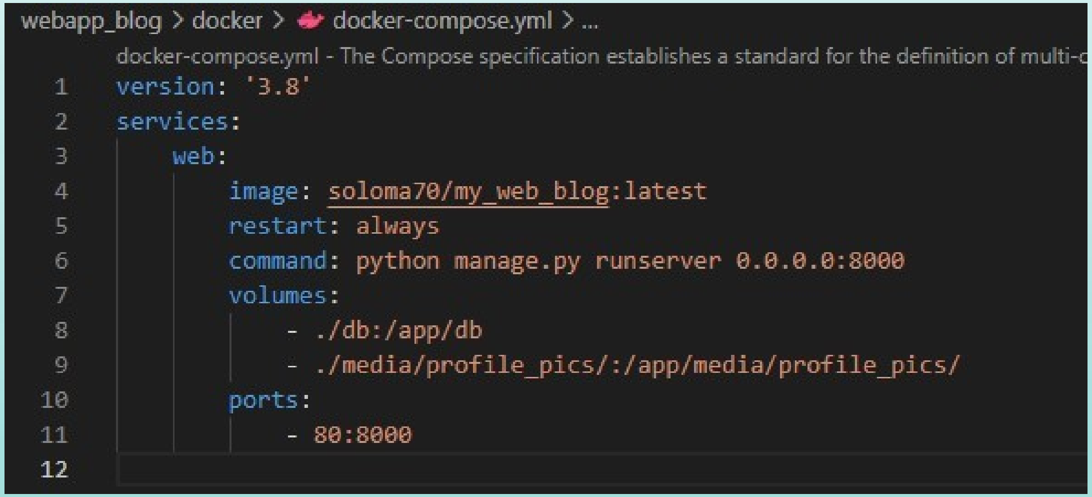


webapp_blog > .dockerignore

```
datadump.json
*.md
.dockerignore
Dockerfile
*.yml
db/
media/
.git/
jenkins/
docker/

```



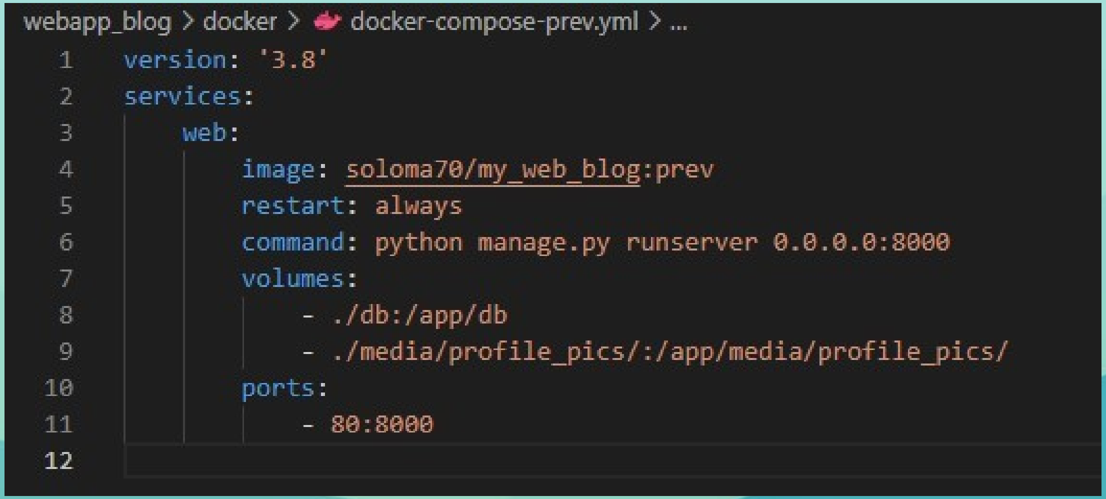
Docker-compose.yml



webapp_blog > docker > docker-compose.yml > ...

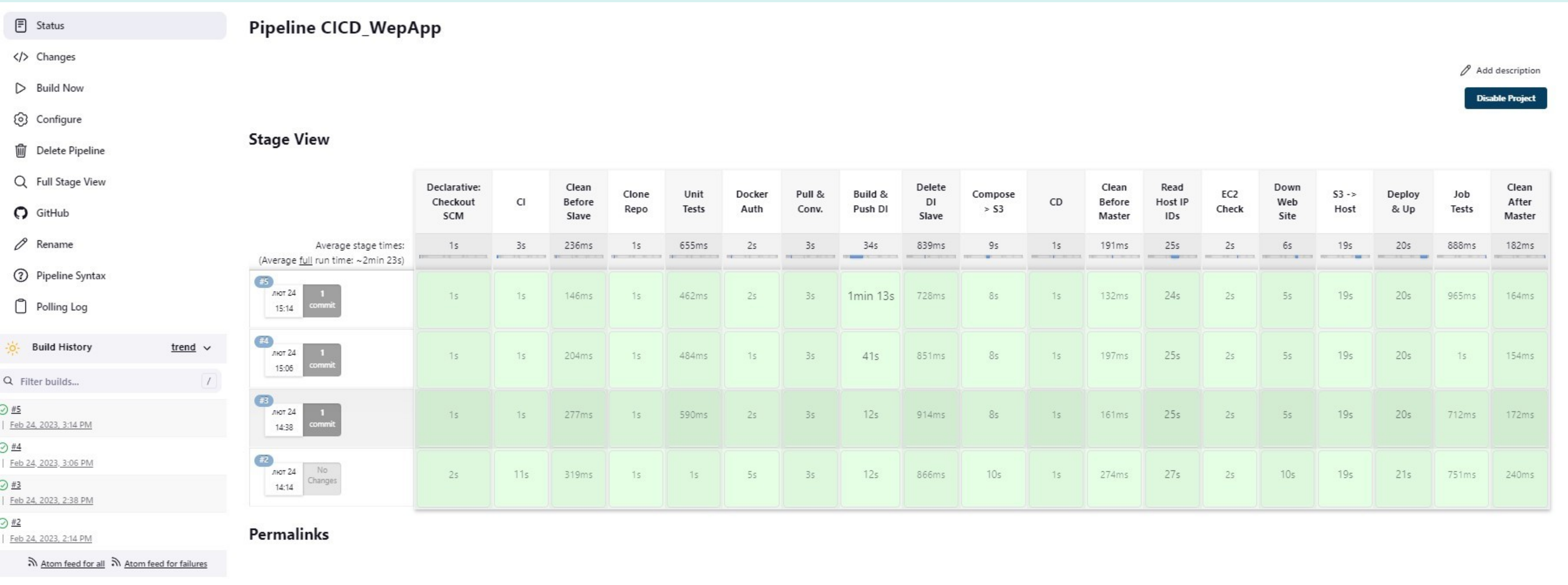
docker-compose.yml - The Compose specification establishes a standard for the definition of multi-c

```
version: '3.8'
services:
  web:
    image: soloma70/my_web_blog:latest
    restart: always
    command: python manage.py runserver 0.0.0.0:8000
    volumes:
      - ./db:/app/db
      - ./media/profile_pics/:/app/media/profile_pics/
    ports:
      - 80:8000

```



webapp_blog > docker > docker-compose-prev.yml > ...

```
version: '3.8'
services:
  web:
    image: soloma70/my_web_blog:prev
    restart: always
    command: python manage.py runserver 0.0.0.0:8000
    volumes:
      - ./db:/app/db
      - ./media/profile_pics/:/app/media/profile_pics/
    ports:
      - 80:8000

```



Status
Changes
Build Now
Configure
Delete Pipeline
Full Stage View
GitHub
Rename
Pipeline Syntax
Polling Log

Build History trend

Filter builds...

#5 Feb 24, 2023, 3:14 PM
#4 Feb 24, 2023, 3:06 PM
#3 Feb 24, 2023, 2:38 PM
#2 Feb 24, 2023, 2:14 PM

Atom feed for all Atom feed for failures

Pipeline CICD_WepApp

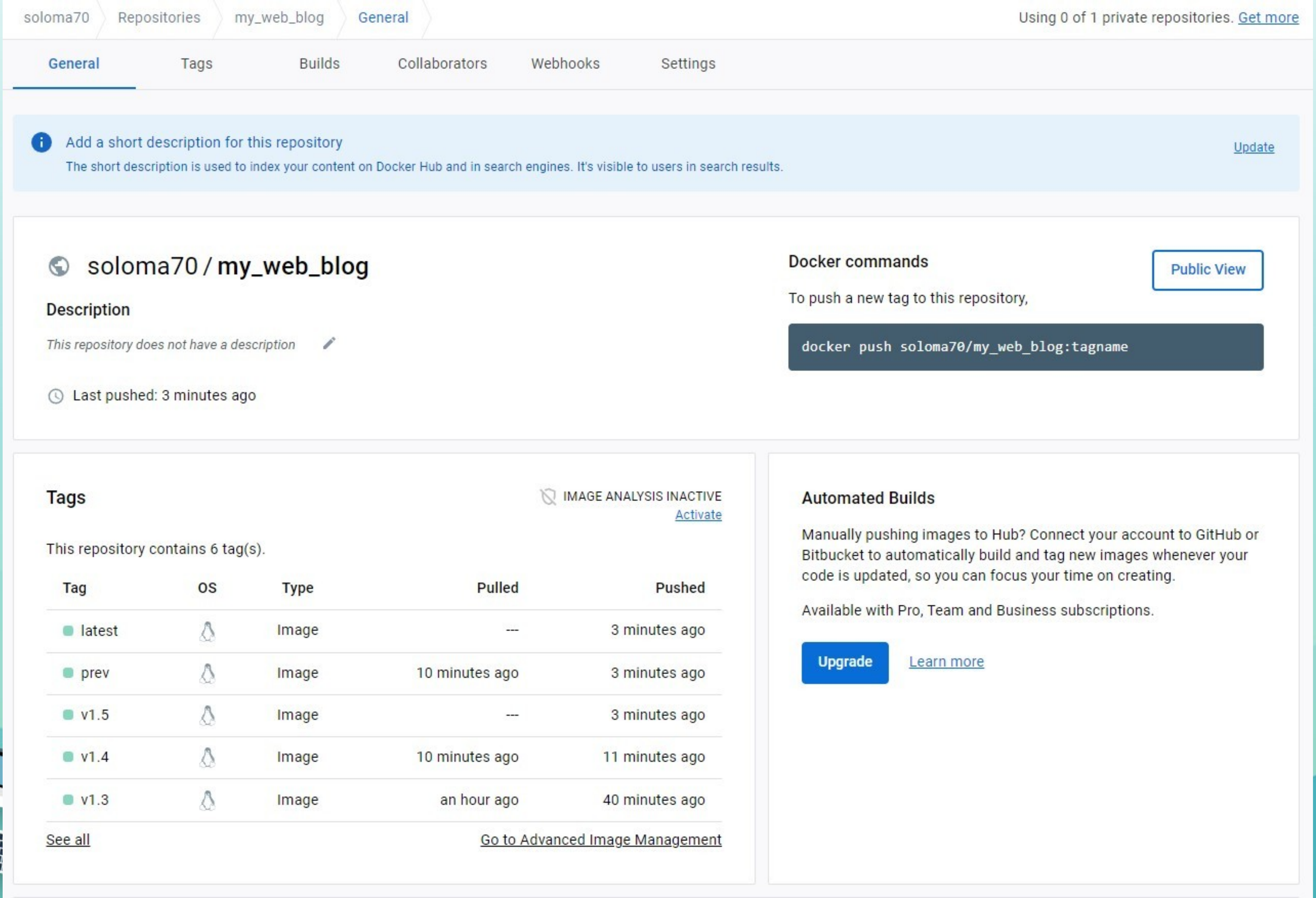
Add description

Disable Project

Stage View

	Declarative: Checkout SCM	CI	Clean Before Slave	Clone Repo	Unit Tests	Docker Auth	Pull & Conv.	Build & Push DI	Delete DI Slave	Compose > S3	CD	Clean Before Master	Read Host IP IDs	EC2 Check	Down Web Site	S3 -> Host	Deploy & Up	Job Tests	Clean After Master
Average stage times: (Average full run time: ~2min 23s)	1s	3s	236ms	1s	655ms	2s	3s	34s	839ms	9s	1s	191ms	25s	2s	6s	19s	20s	888ms	182ms
#5 лют 24 15:14 1 commit	1s	1s	146ms	1s	462ms	2s	3s	1min 13s	728ms	8s	1s	132ms	24s	2s	5s	19s	20s	965ms	164ms
#4 лют 24 15:06 1 commit	1s	1s	204ms	1s	484ms	1s	3s	41s	851ms	8s	1s	197ms	25s	2s	5s	19s	20s	1s	154ms
#3 лют 24 14:38 1 commit	1s	1s	277ms	1s	590ms	2s	3s	12s	914ms	8s	1s	161ms	25s	2s	5s	19s	20s	712ms	172ms
#2 лют 24 14:14 No Changes	2s	11s	319ms	1s	1s	5s	3s	12s	866ms	10s	1s	274ms	27s	2s	10s	19s	21s	751ms	240ms

Permalinks


soloma70 Repositories my_web_blog General
Using 0 of 1 private repositories. Get more
General Tags Builds Collaborators Webhooks Settings

Add a short description for this repository

The short description is used to index your content on Docker Hub and in search engines. It's visible to users in search results.

Update

soloma70 / **my_web_blog**

Description

This repository does not have a description

Last pushed: 3 minutes ago

Docker commands

To push a new tag to this repository,

```
docker push soloma70/my_web_blog:tagname
```

Public View

Tags

IMAGE ANALYSIS INACTIVE
Activate

This repository contains 6 tag(s).

Tag	OS	Type	Pulled	Pushed
latest		Image	---	3 minutes ago
prev		Image	10 minutes ago	3 minutes ago
v1.5		Image	---	3 minutes ago
v1.4		Image	10 minutes ago	11 minutes ago
v1.3		Image	an hour ago	40 minutes ago

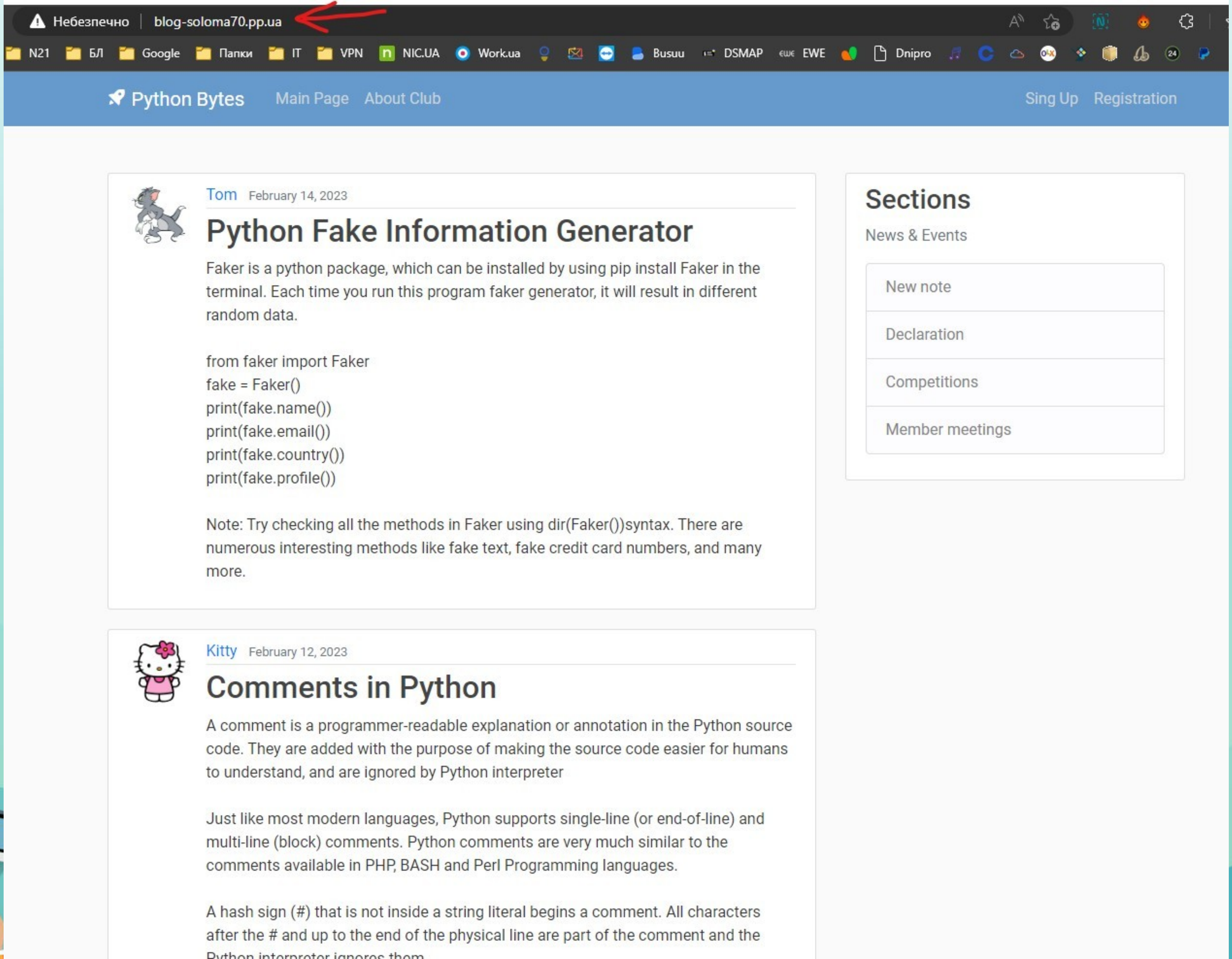
See all Go to Advanced Image Management

Automated Builds

Manually pushing images to Hub? Connect your account to GitHub or Bitbucket to automatically build and tag new images whenever your code is updated, so you can focus your time on creating.

Available with Pro, Team and Business subscriptions.

Upgrade Learn more



Python Bytes Main Page About Club Sing Up Registration

Tom February 14, 2023

Python Fake Information Generator

Faker is a python package, which can be installed by using pip install Faker in the terminal. Each time you run this program faker generator, it will result in different random data.

```
from faker import Faker
fake = Faker()
print(fake.name())
print(fake.email())
print(fake.country())
print(fake.profile())
```

Note: Try checking all the methods in Faker using dir(Faker())syntax. There are numerous interesting methods like fake text, fake credit card numbers, and many more.

Kitty February 12, 2023

Comments in Python

A comment is a programmer-readable explanation or annotation in the Python source code. They are added with the purpose of making the source code easier for humans to understand, and are ignored by Python interpreter

Just like most modern languages, Python supports single-line (or end-of-line) and multi-line (block) comments. Python comments are very much similar to the comments available in PHP, BASH and Perl Programming languages.

A hash sign (#) that is not inside a string literal begins a comment. All characters after the # and up to the end of the physical line are part of the comment and the Python interpreter ignores them.

Sections

News & Events

- New note
- Declaration
- Competitions
- Member meetings

Thank You!