Build a CD Pipeline

Key Takeaways

TECHWORLD WITH NANA

All rights reserved to nnSoftware GmbH

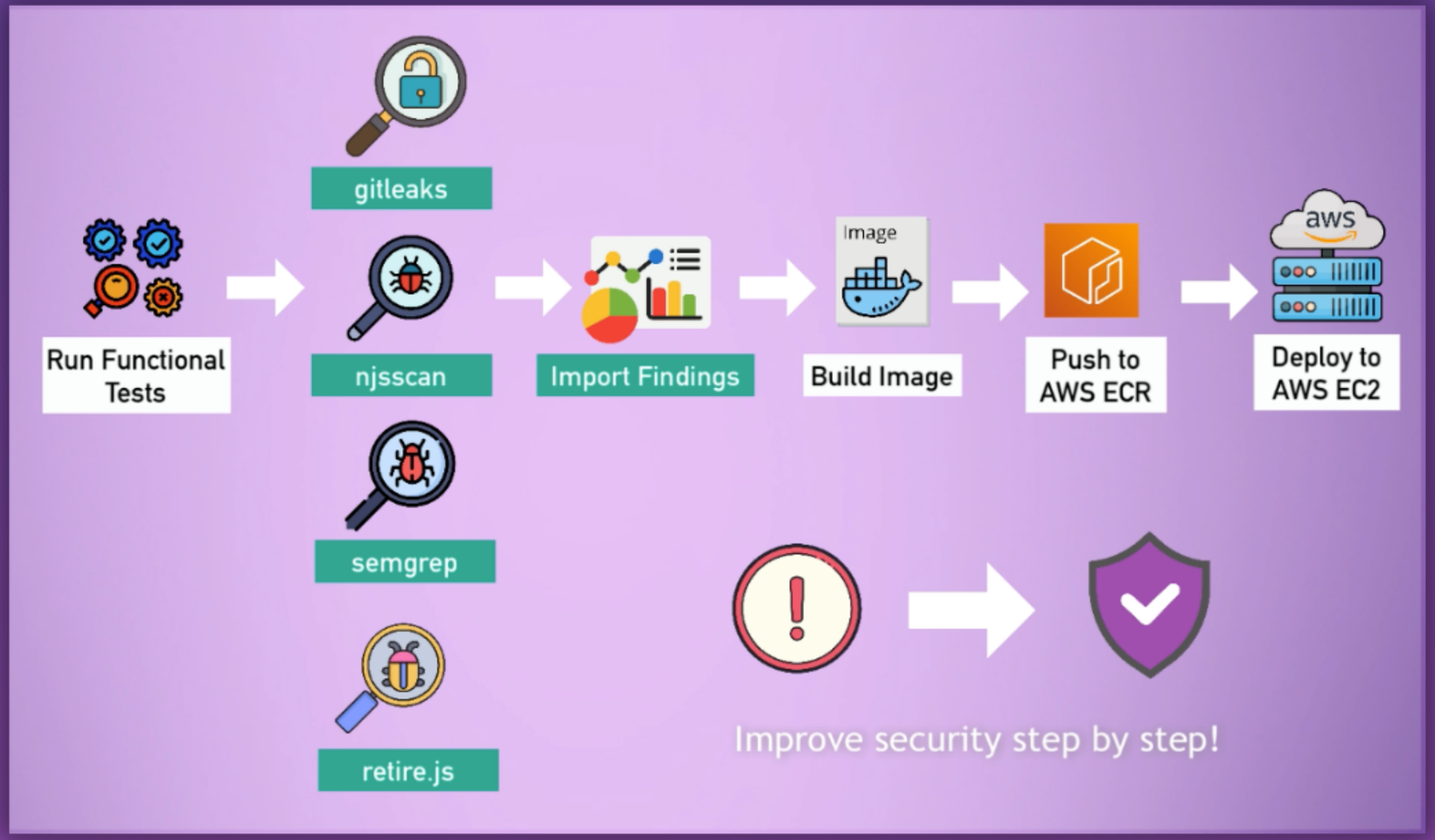
TECHWORLD WITH NANA

About TechWorld with Nana

TechWorld with Nana is an established name in the DevOps and Cloud industry, and it stands for the quality trainings helping 1,000s of engineers acquire the most in-demand skills in this field.

Our mission is enable individual engineers as well as companies to take advantage of the recent developments in Cloud and DevOps fields, to use technologies and concepts in order to create efficient, automated, streamlined DevSecOps processes in organisations.

CI Part complete, CD Part still to be set up and secured



We will continue with CD part, securing step by step

But first, set up CD part **with no or basic security**

TECHWORLD WITH NANA

Introducing AWS Cloud

2 Authentication Layers with Cloud Platform

To access an AWS service, like AWS ECR, we need to authenticate twice

- 1st: Authenticate with AWS
- 2nd: Authenticate and access service within that account

TECHWORLD WITH NANA

AWS Access Types

- There are 2 types of access:

AWS Management Console

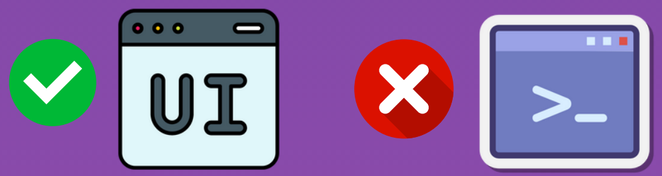
- Email and Password

AWS Command Line Interface

- Access Key Pair

- Select which access you allow for the user

AWS User has **separate credentials** for each type

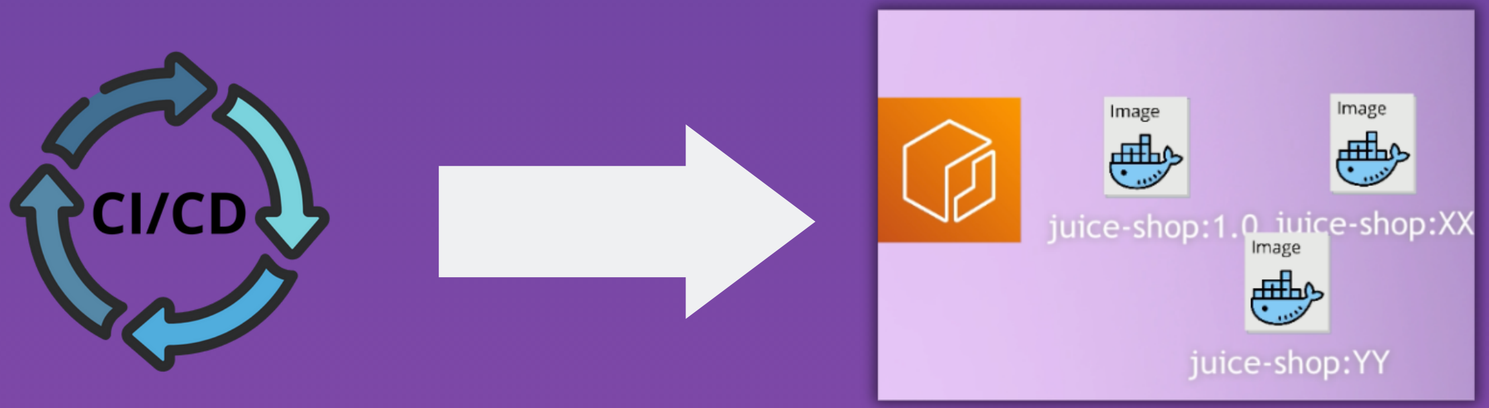


TECHWORLD WITH NANA

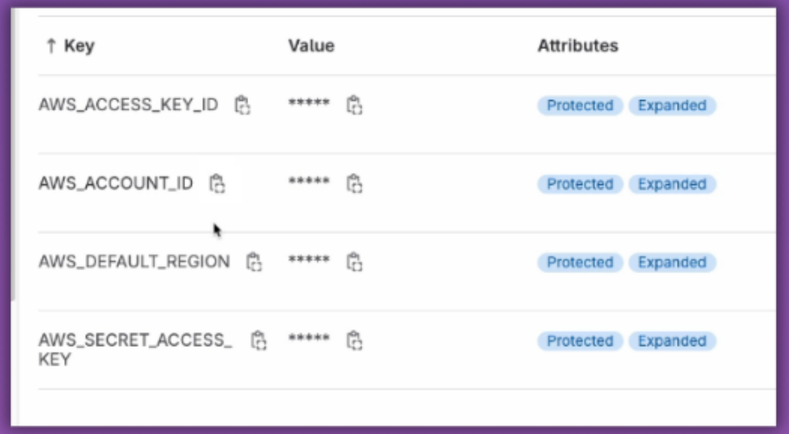
Deploy to AWS ECR

Push to AWS ECR

- In our pipeline, we want to push the images to an AWS ECR container repository



1 - Add variables to GitLab Platform for AWS Account Login

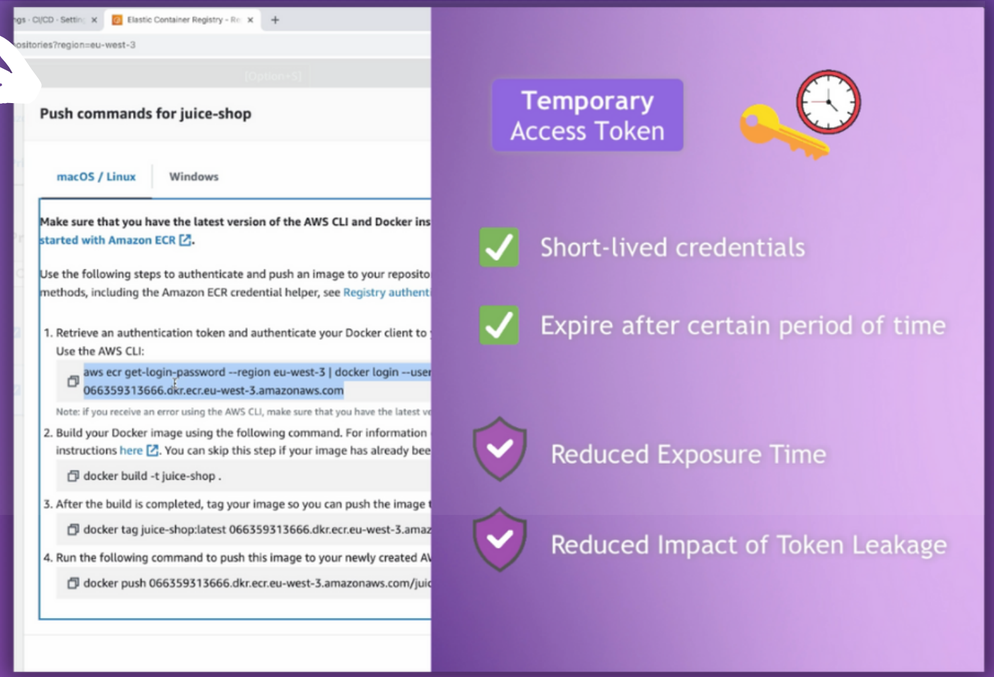


TECHWORLD WITH NANA

Push to AWS ECR

2 - Copy push command from AWS ECR

- Push command includes **retrieving an authentication token** and authenticating to AWS ECR
- After authentication and authorization the image is pushed to the repository



3 - Adjust Pipeline Code

```
build_image:
  stage: build
  image: docker:24
  services:
    - docker:24-dind
  before_script:
    - apk --no-cache add python3 py3-pip
    - pip3 install --no-cache-dir awscli
    - aws ecr get-login-password | docker login --username AWS --password-stdin $AWS_ACCOUNT_ID.dkr.ecr.$AWS_DEFAULT_REGION.amazonaws.com
  script:
    - docker build -t $IMAGE_NAME:$CI_COMMIT_SHA -t $IMAGE_NAME:latest .
    - docker push $IMAGE_NAME:$CI_COMMIT_SHA
    - docker push $IMAGE_NAME:latest
```

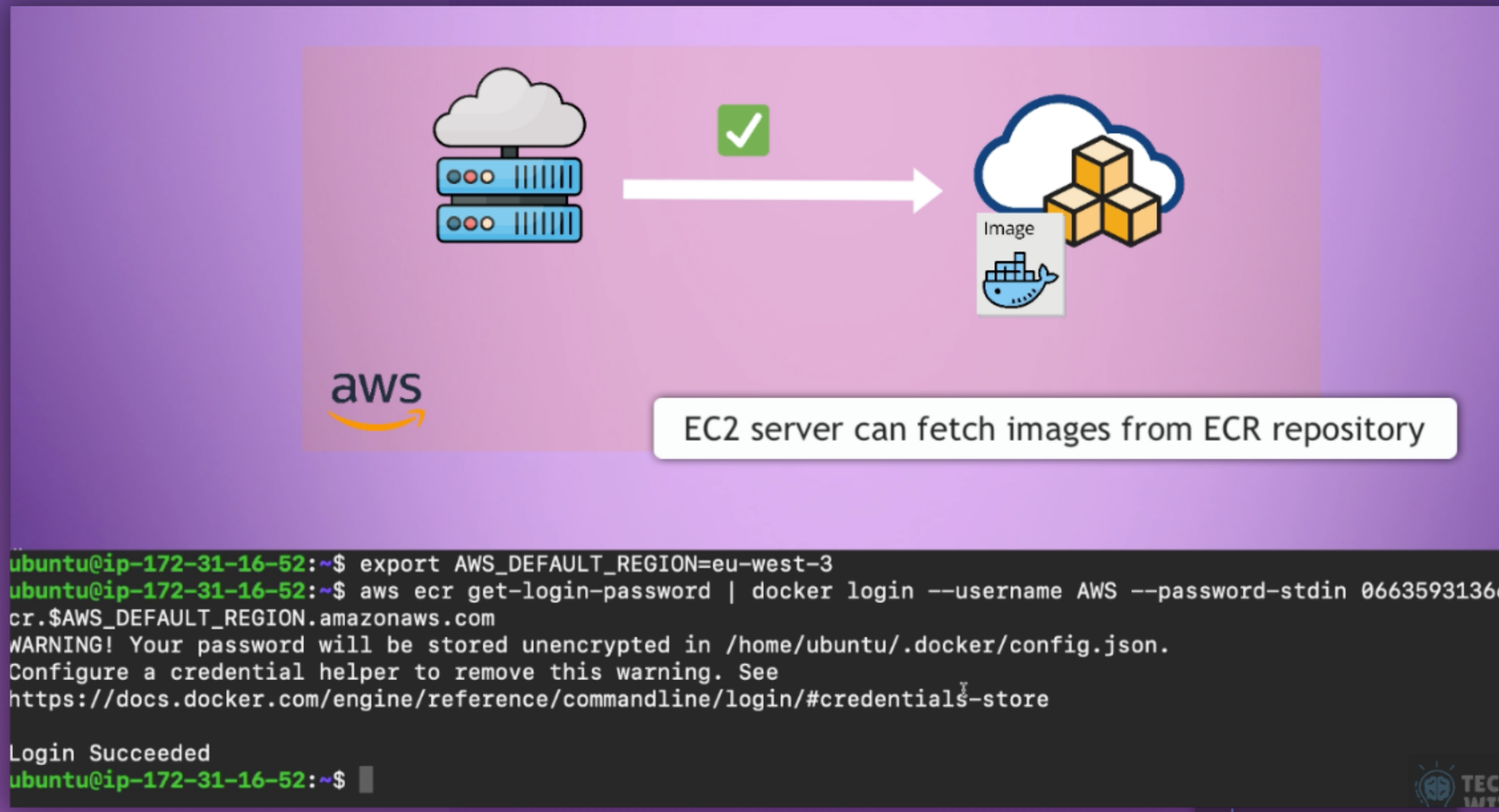
TECHWORLD WITH NANA

Deploy to AWS EC2

Provision EC2 Instance

How to

1. Create EC2 Instance
2. Open SSH port on EC2 Instance
3. Install Docker on EC2 Instance
4. Install AWS CLI
5. Export AWS environment variables

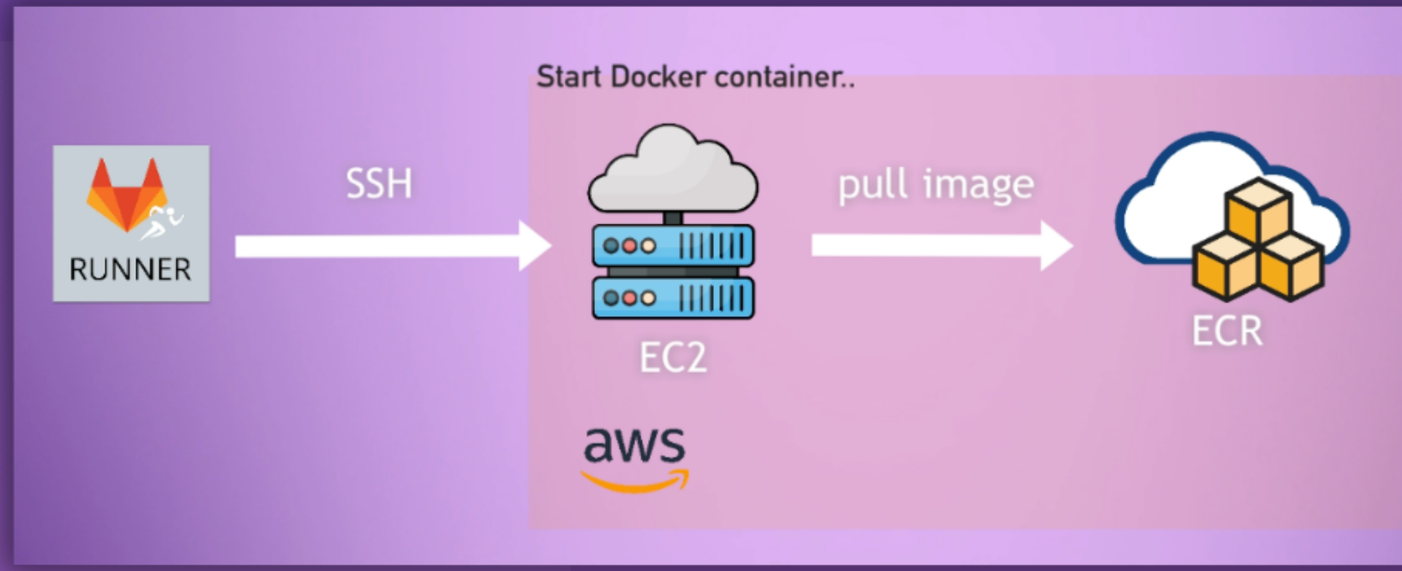


TECHWORLD WITH NANA

Add Deployment Step

How to

1. GitLab Runner needs to SSH into EC2 Instance
2. Pull image from ECR repository
3. Start Docker container



1 - Add SSH Key variables

AWS_SECRET_ACCESS_KEY	*****	Protected Expanded
SSH_PRIVATE_KEY	*****	File Protected Expanded

TECHWORLD WITH NANA

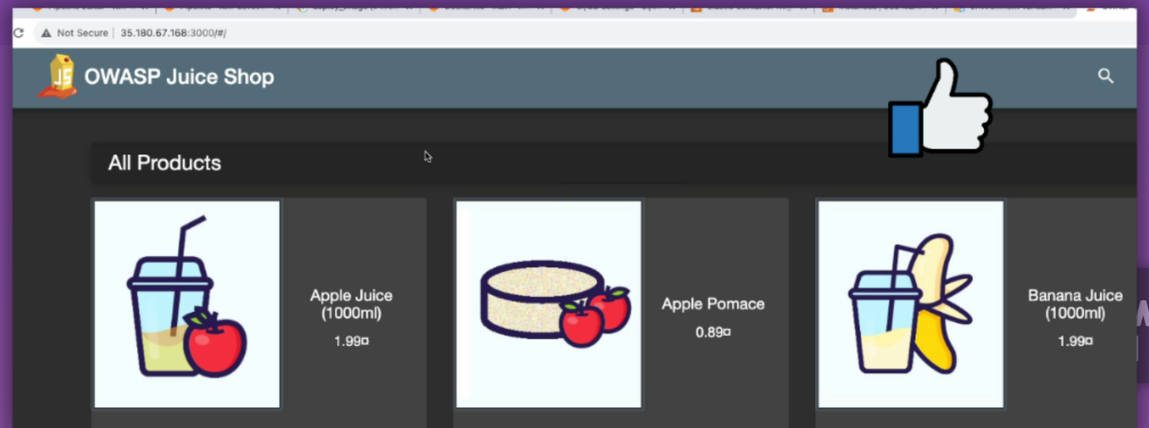
Add Deployment Step

2 - Adjust Pipeline Code

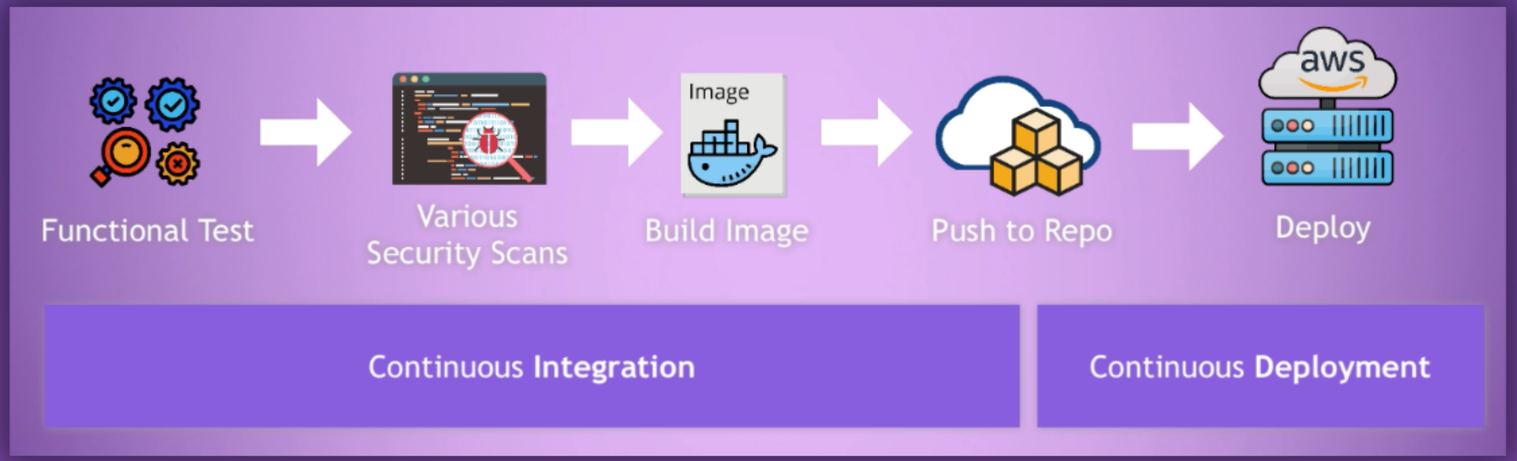
```
deploy_image:
  stage: deploy
  image: debian:bullseye-slim
  before_script:
    - apt update -y && apt install openssh-client -y
    - eval $(ssh-agent -s)
    - chmod 400 "$SSH_PRIVATE_KEY"
    - ssh-add "$SSH_PRIVATE_KEY"
    - mkdir -p ~/.ssh
    - chmod 700 ~/.ssh
  script:
    - ssh -o StrictHostKeyChecking=no $SERVER_USER@$SERVER_IP "docker pull $IMAGE_NAME:latest"
    - ssh -o StrictHostKeyChecking=no $SERVER_USER@$SERVER_IP "docker stop juice-shop || true && docker rm juice-shop || true"
    - ssh -o StrictHostKeyChecking=no $SERVER_USER@$SERVER_IP "docker run -d --name juice-shop -p 3000:3000 $IMAGE_NAME:latest"
```

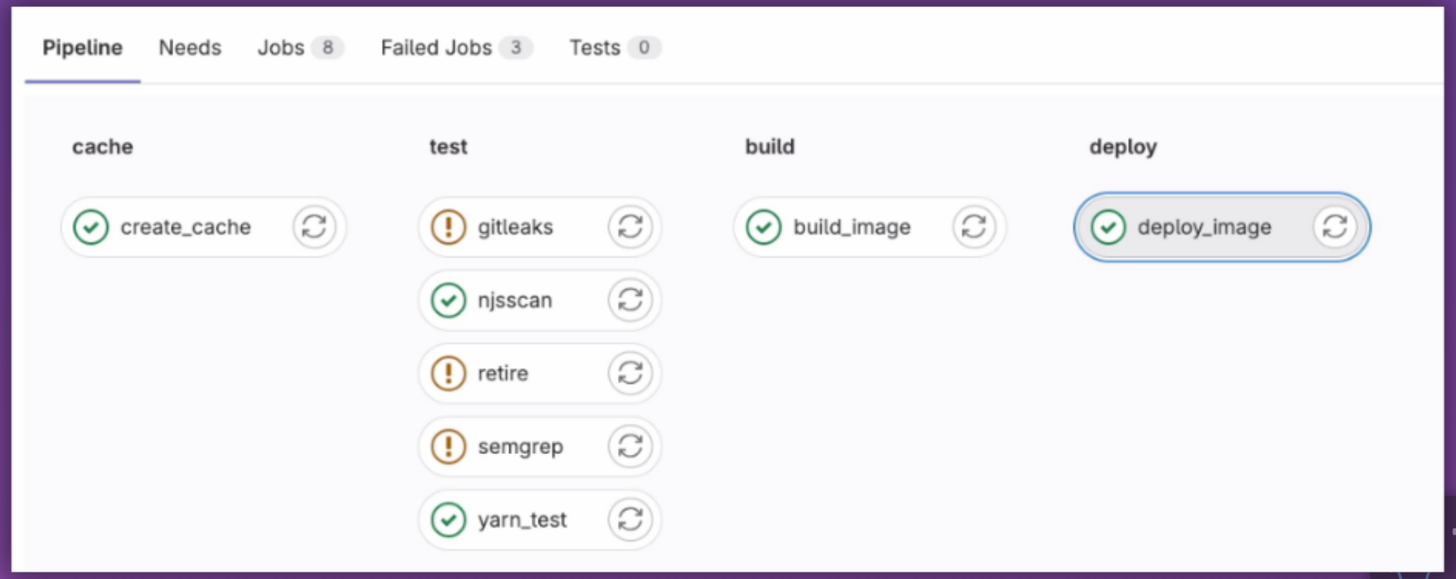
3 - Open Port to make application accessible

- Add inbound firewall rule to open port



Our Basic Continuous Deployment





TECHWORLD WITH NANA

Use Self-Managed GitLab Runner

Why use a self-hosted Runner

GitLab.com's shared runners

GitLab Shared Runner

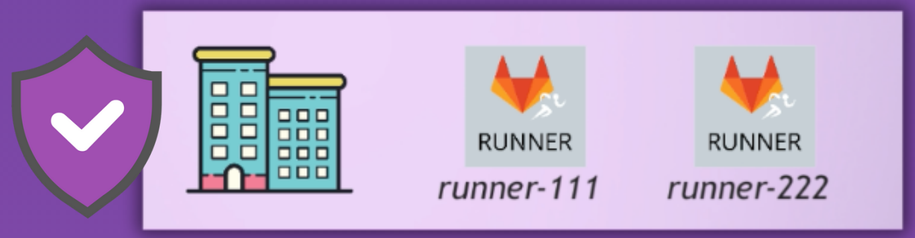
- Shared runners are available to all projects in a GitLab instance
- Shared runners on gitlab.com are available to **all users on the platform**

We have **no insight and control** over those shared runners

TECHWORLD WITH NANA

Have full control over your infrastructure

- For companies with higher security standards, it's security best practice to have full control over your infrastructure



Self-managed runners

- Have control over your runners, who execute your jobs

Self-managed GitLab instance

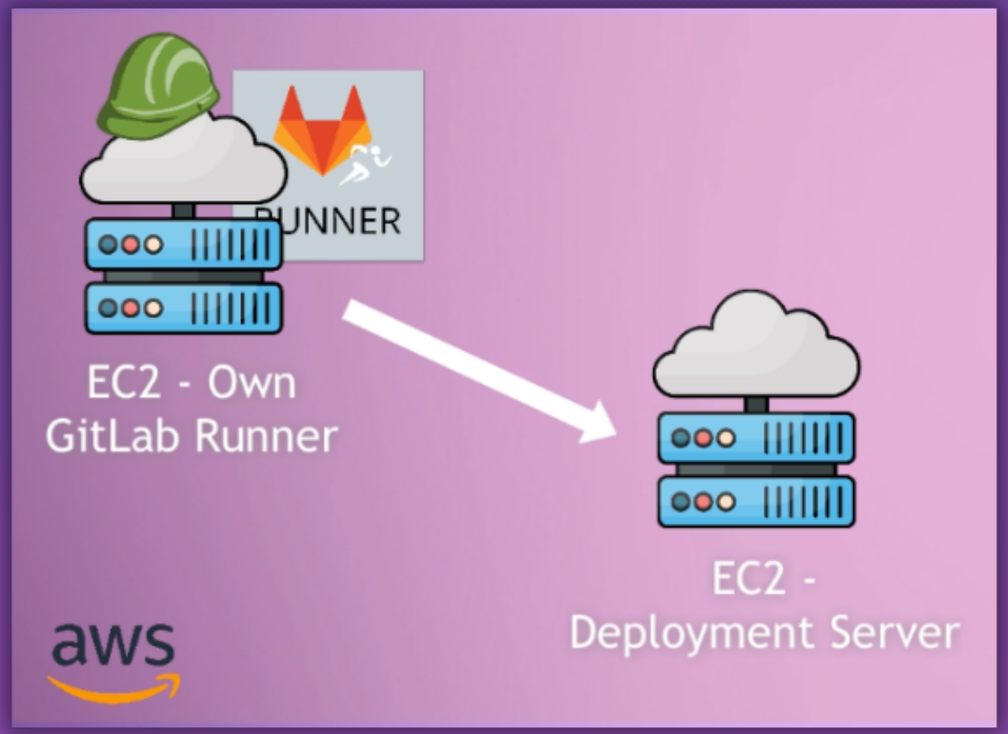
- Holds all your pipeline configuration data etc.

- Host your own GitLab instance to keep all your data in private network

TECHWORLD WITH NANA

Register a self-managed runner

- We can set up an EC2 Instance as GitLab Runner
- Switching to self-managed runner allows us to leverage security best practices on AWS



With that change, we can leverage AWS IAM Best Practices

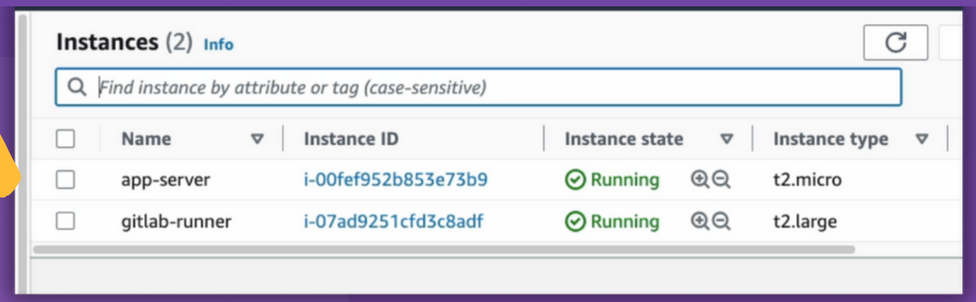
- Give EC2 Instance AWS Role
- Use that role to access any services within AWS that we need

TECHWORLD WITH NANA

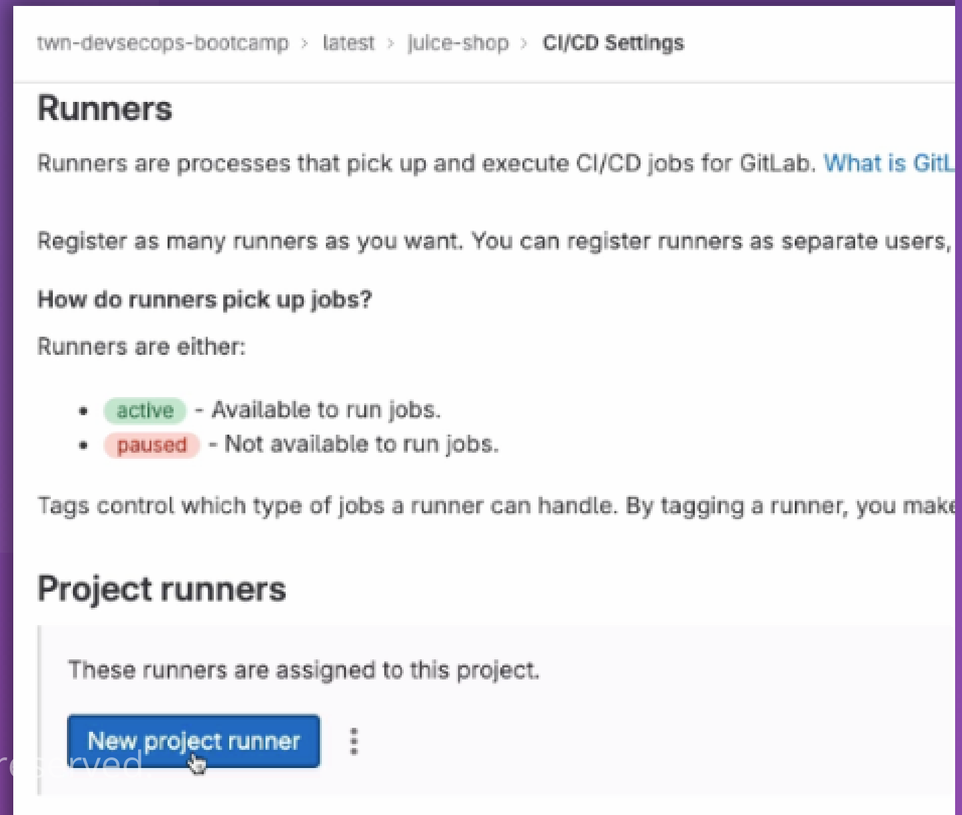
Register a self-managed runner

How to - Part 1

1. **Create EC2 Instance** for Self-Hosted Runner



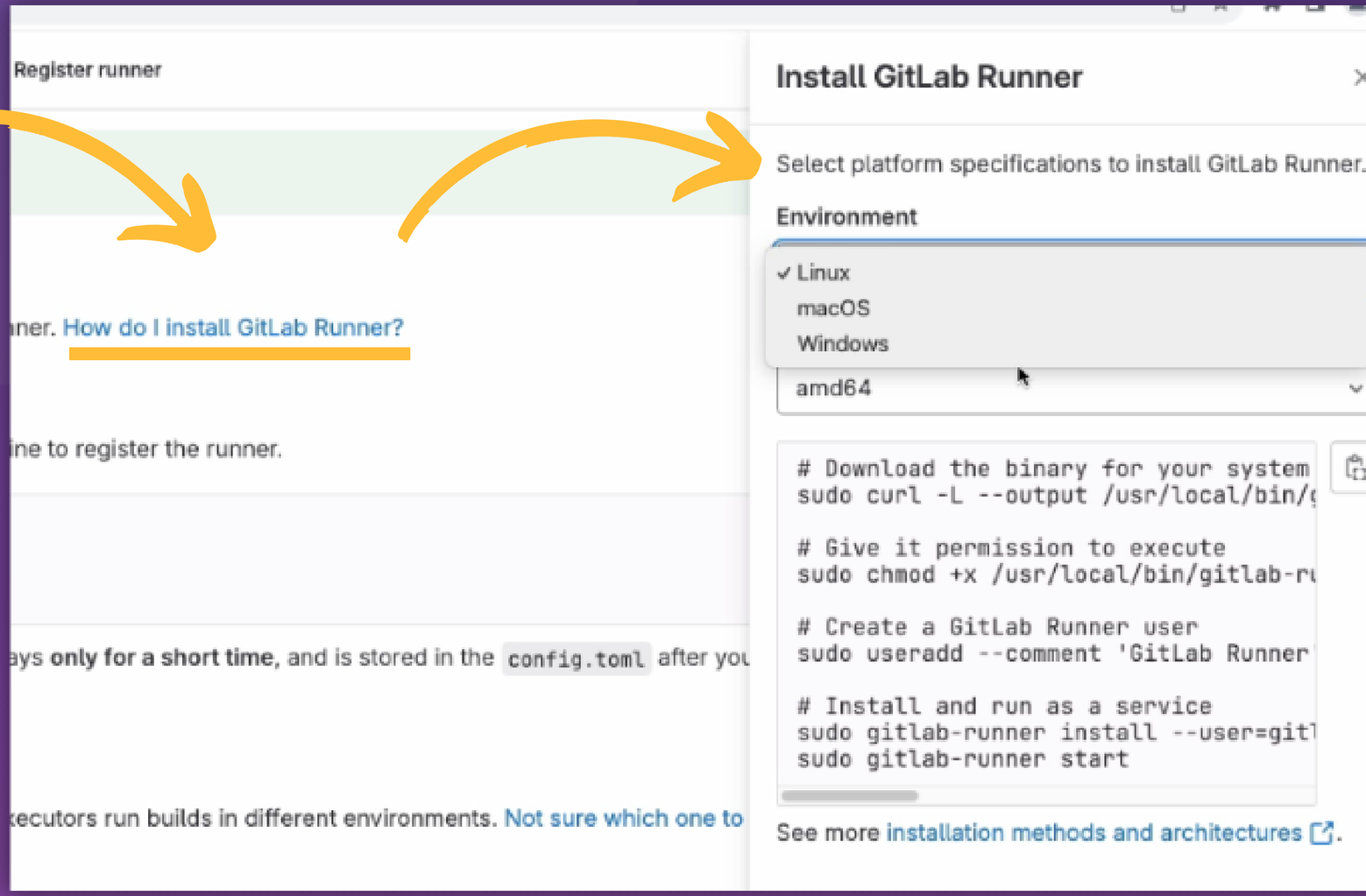
2. Create **new project runner** on GitLab
    a. This generates a command that registers the runner with all its configurations



Register a self-managed runner

How to - Part 2

3. **Install GitLab Runner**

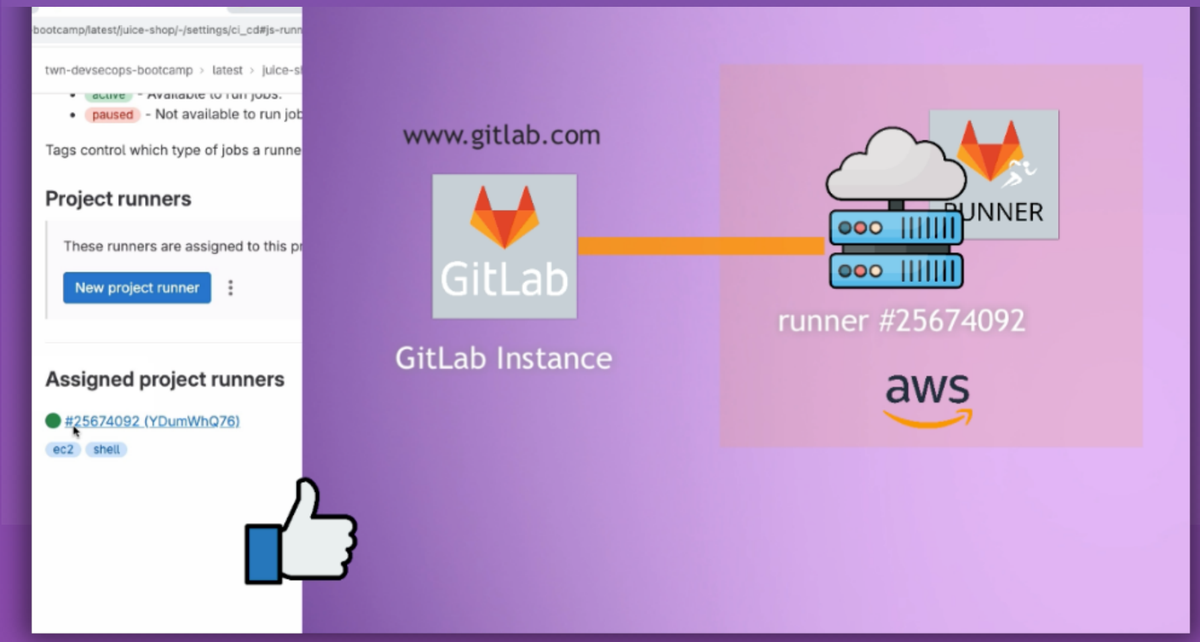


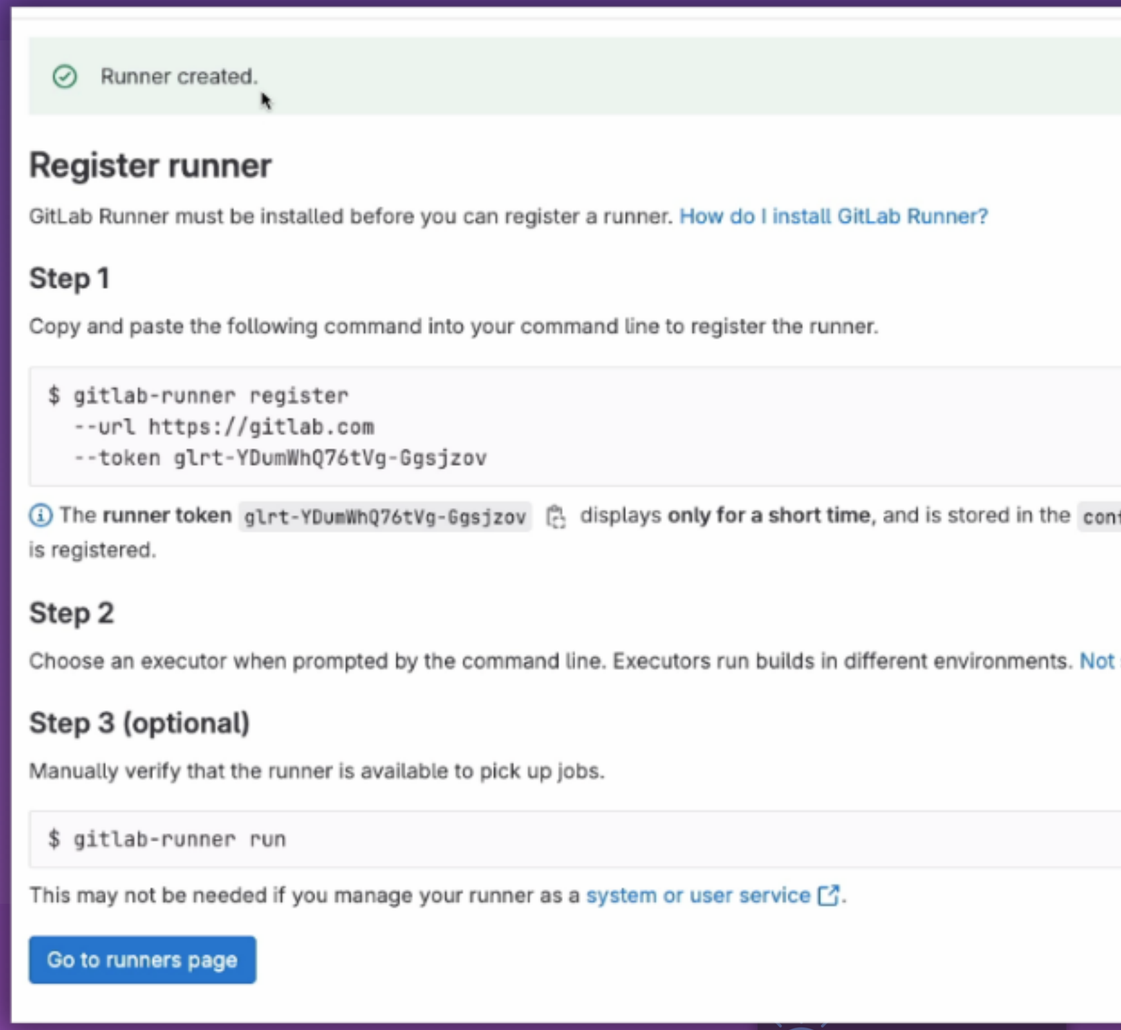
TECHWORLD WITH NANA

Register a self-managed runner

How to - Part 3

4. **Register** GitLab **Runner** on EC2 Instance





TECHWORLD WITH NANA