



# Build a CD Pipeline

## Key Takeaways

# All rights reserved to nnSoftware GmbH

No part of this publication may be reproduced, copied, transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without the prior written permission of nnSoftware GmbH

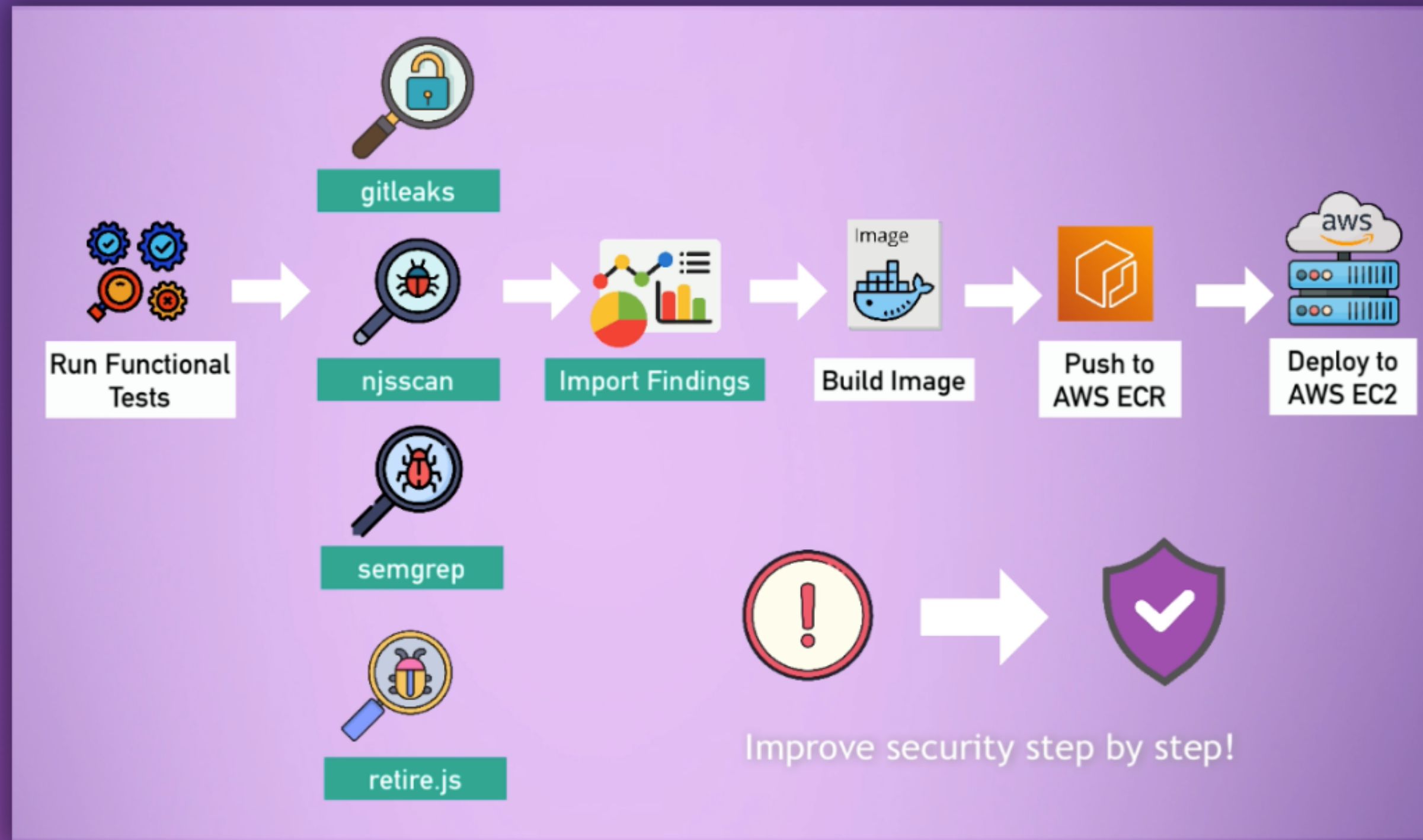
# About TechWorld with Nana

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**

# 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

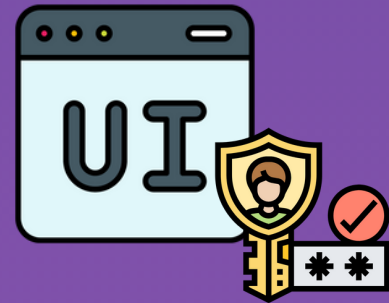


2nd: Authenticate and access service within that account



# 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

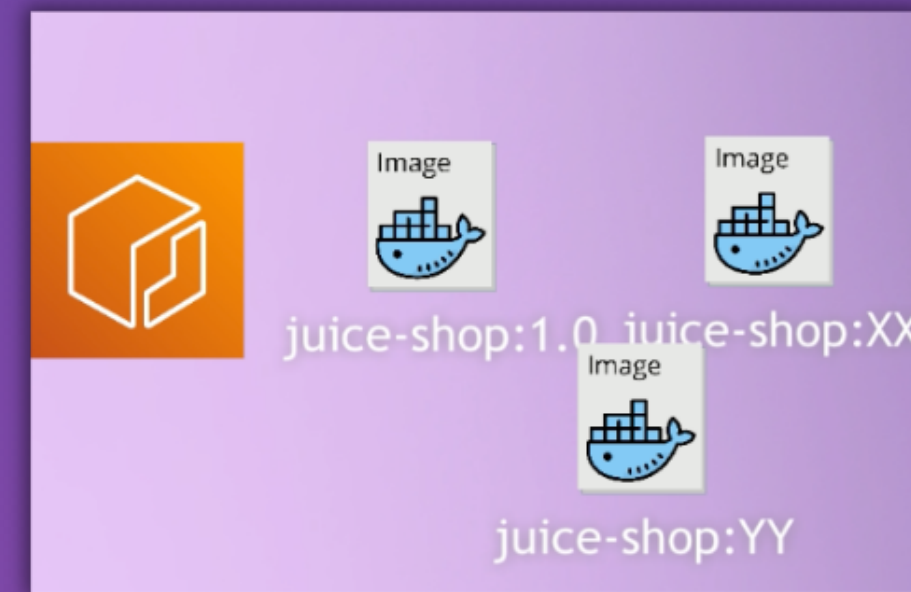
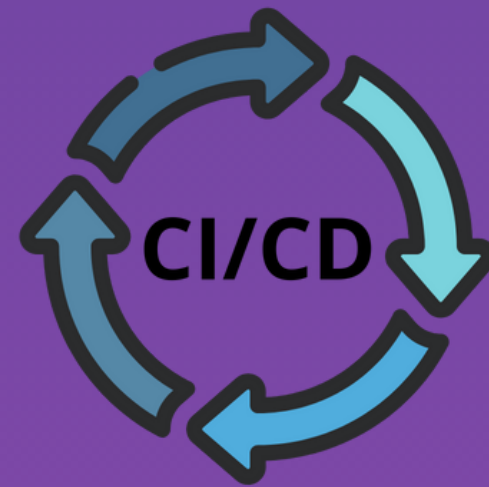


# Deploy to AWS ECR



# Push to AWS ECR

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



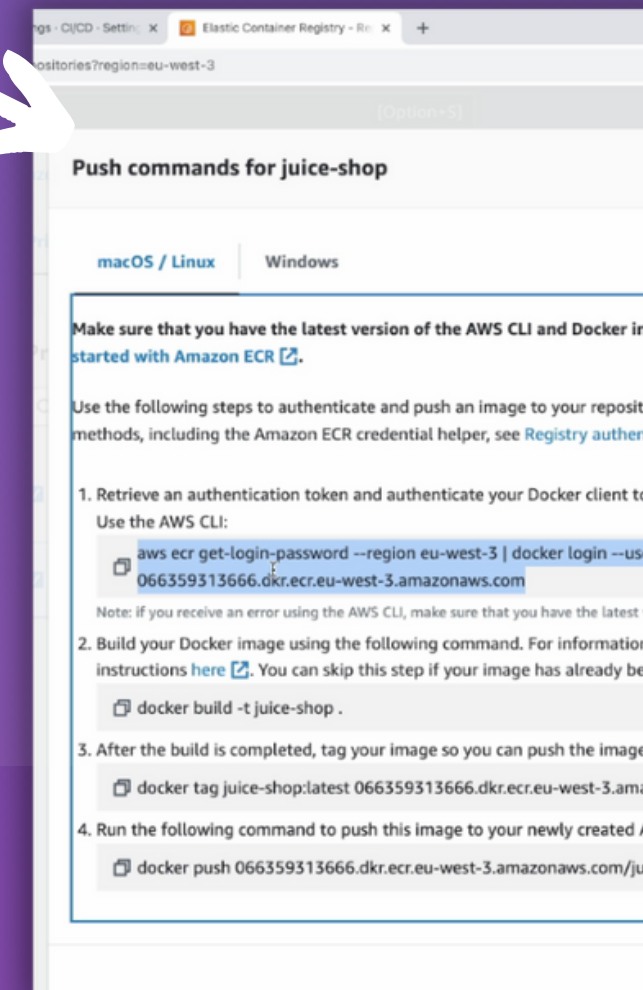
**1 - Add variables to GitLab Platform for AWS Account Login**

↑ Key		Value		Attributes
AWS_ACCESS_KEY_ID		*****		Protected Expanded
AWS_ACCOUNT_ID		*****		Protected Expanded
AWS_DEFAULT_REGION		*****		Protected Expanded
AWS_SECRET_ACCESS_KEY		*****		Protected Expanded

# 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



### Temporary Access Token



- ✓ Short-lived credentials
- ✓ Expire after certain period of time
- ✓ Reduced Exposure Time
- ✓ Reduced Impact of Token Leakage

## 3 - Adjust Pipeline Code

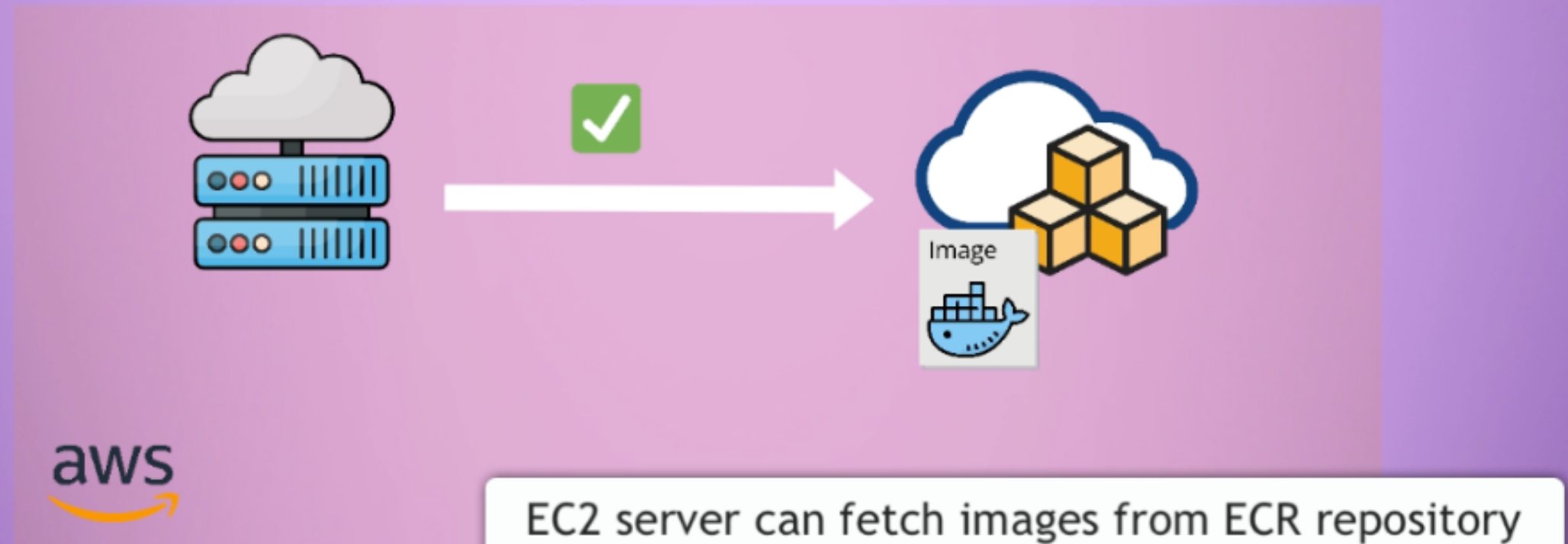
```
118 build_image:
119     stage: build
120     image: docker:24
121     services:
122     | - docker:24-dind
123     before_script:
124     | - apk --no-cache add python3 py3-pip
125     | - pip3 install --no-cache-dir awscli
126     | - aws ecr get-login-password | docker login --username AWS --password-stdin $AWS_ACCOUNT_ID.dkr.ecr.$AWS_DEFAULT_REGION.amazonaws.com
127     script:
128     | - docker build -t $IMAGE_NAME:$CI_COMMIT_SHA -t $IMAGE_NAME:latest .
129     | - docker push $IMAGE_NAME:$CI_COMMIT_SHA
130     | - docker push $IMAGE_NAME:latest
131
```

# 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



```
ubuntu@ip-172-31-16-52:~$ export AWS_DEFAULT_REGION=eu-west-3
ubuntu@ip-172-31-16-52:~$ aws ecr get-login-password | docker login --username AWS --password-stdin 0663593136cr.$AWS_DEFAULT_REGION.amazonaws.com
WARNING! Your password will be stored unencrypted in /home/ubuntu/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

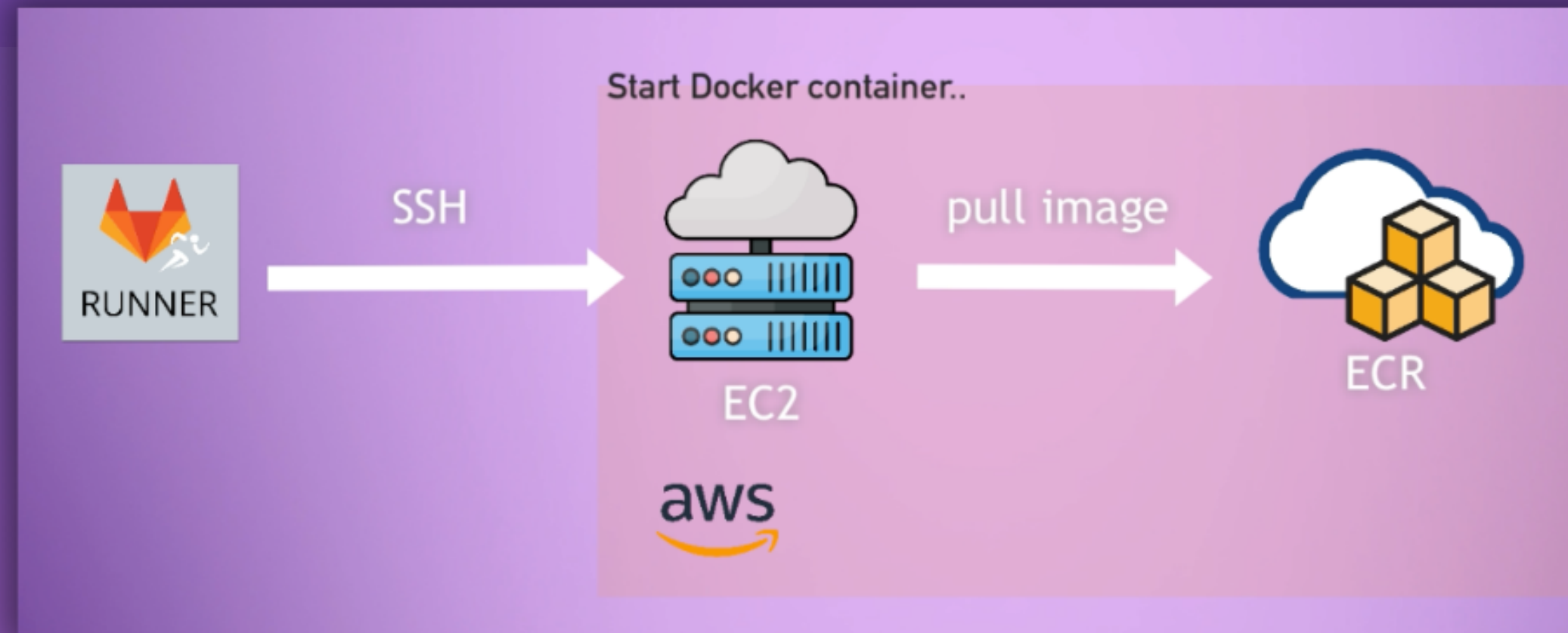
Login Succeeded
ubuntu@ip-172-31-16-52:~$
```



# 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

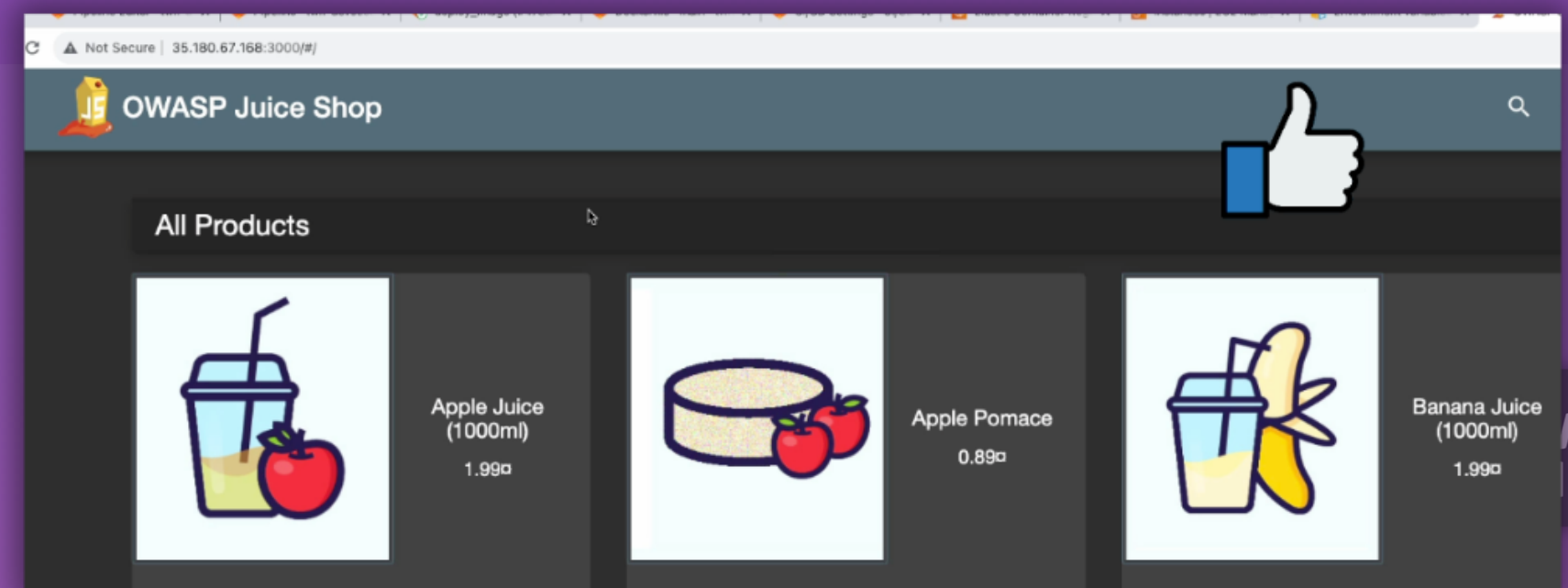
# Add Deployment Step

## 2 - Adjust Pipeline Code

```
135 deploy_image:
136   stage: deploy
137   image: debian:bullseye-slim
138   before_script:
139     - apt update -y && apt install openssh-client -y
140     - eval $(ssh-agent -s)
141     - chmod 400 "$SSH_PRIVATE_KEY"
142     - ssh-add "$SSH_PRIVATE_KEY"
143     - mkdir -p ~/.ssh
144     - chmod 700 ~/.ssh
145   script:
146     - ssh -o StrictHostKeyChecking=no $SERVER_USER@$SERVER_IP "docker pull $IMAGE_NAME:latest"
147     - ssh -o StrictHostKeyChecking=no $SERVER_USER@$SERVER_IP "docker stop juice-shop || true && docker rm juice-shop || true"
148     - ssh -o StrictHostKeyChecking=no $SERVER_USER@$SERVER_IP "docker run -d --name juice-shop -p 3000:3000 $IMAGE_NAME:latest"
149
```

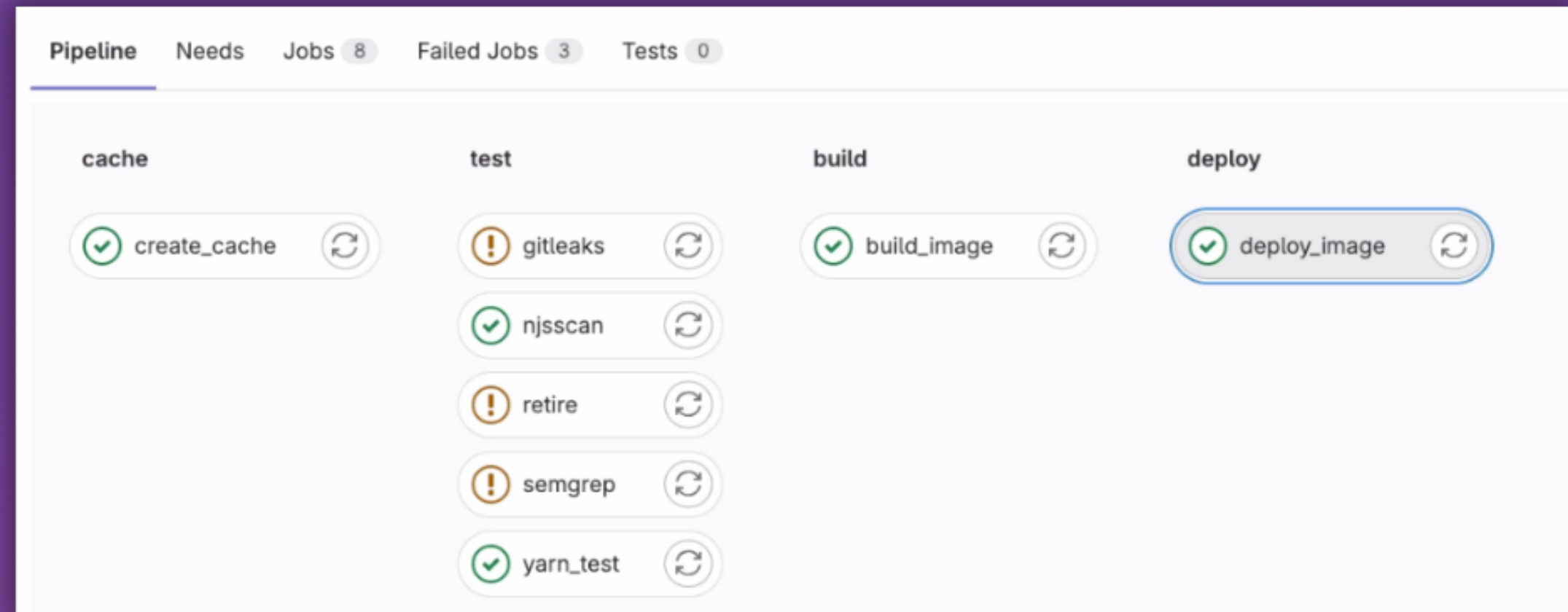
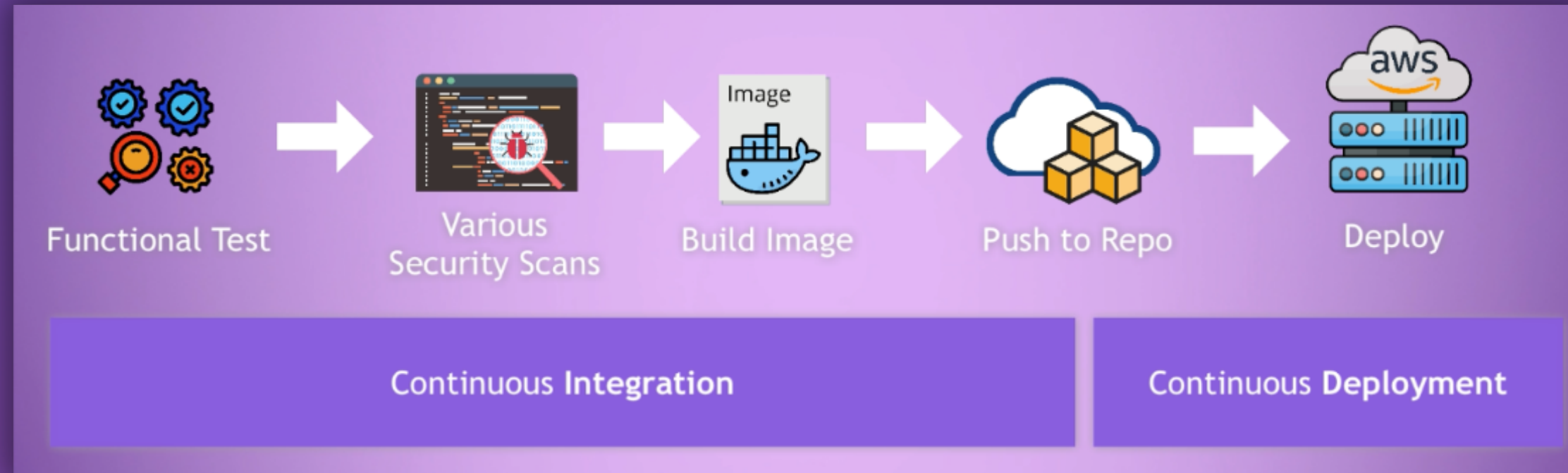
## 3 - Open Port to make application accessible

- Add inbound firewall rule to open port





# Our Basic Continuous Deployment





# Use Self-Managed GitLab Runner

# Why use a self-hosted Runner



GitLab.com's shared runners

## GitLab Shared Runner



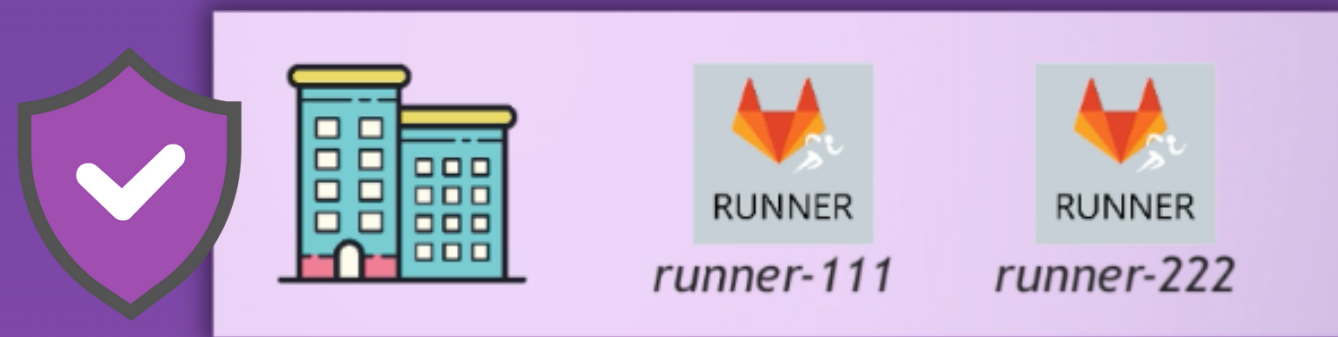
- Shared runners are available to all projects in a GitLab instance
- Shared runners on gitlab.com are available to **all users on the platform**



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

# 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

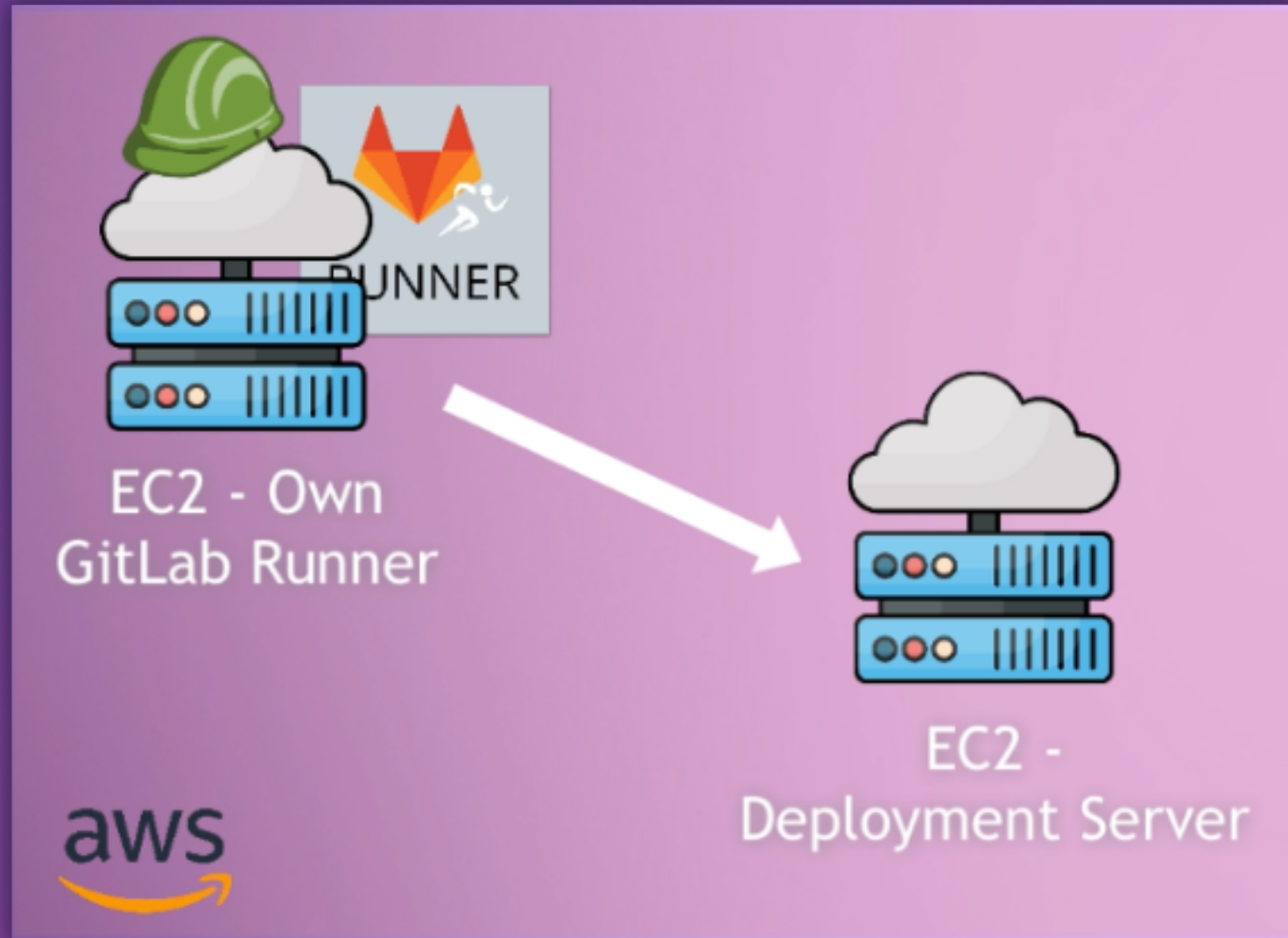


**Self-managed** GitLab instance

- Holds all your pipeline configuration data etc.
- ✓ Host your own GitLab instance to keep all your data in private network

# 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



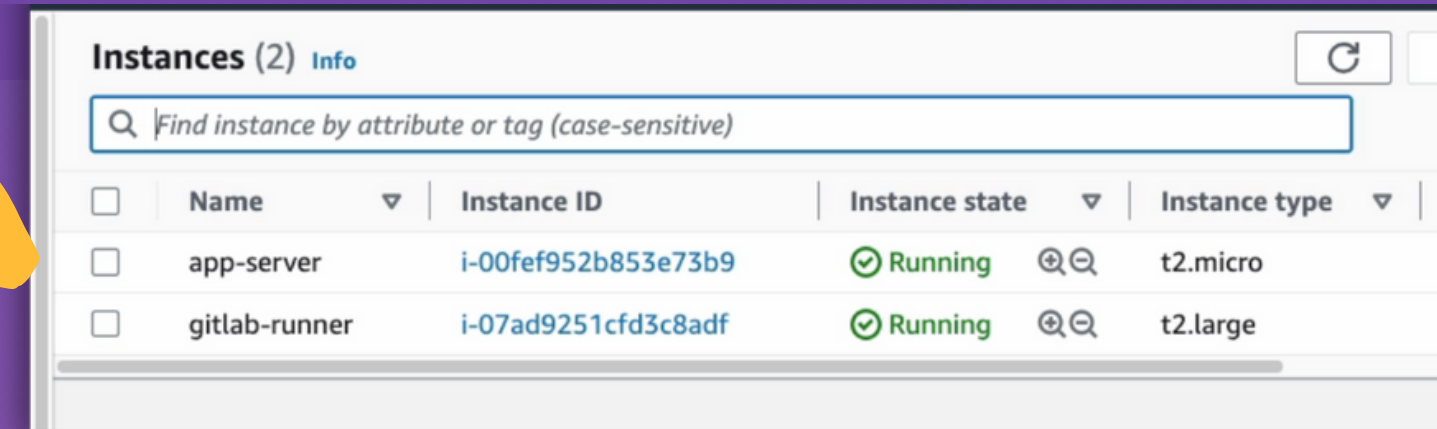
Use that role to access any services within AWS that we need



# Register a self-managed runner

## How to - Part 1

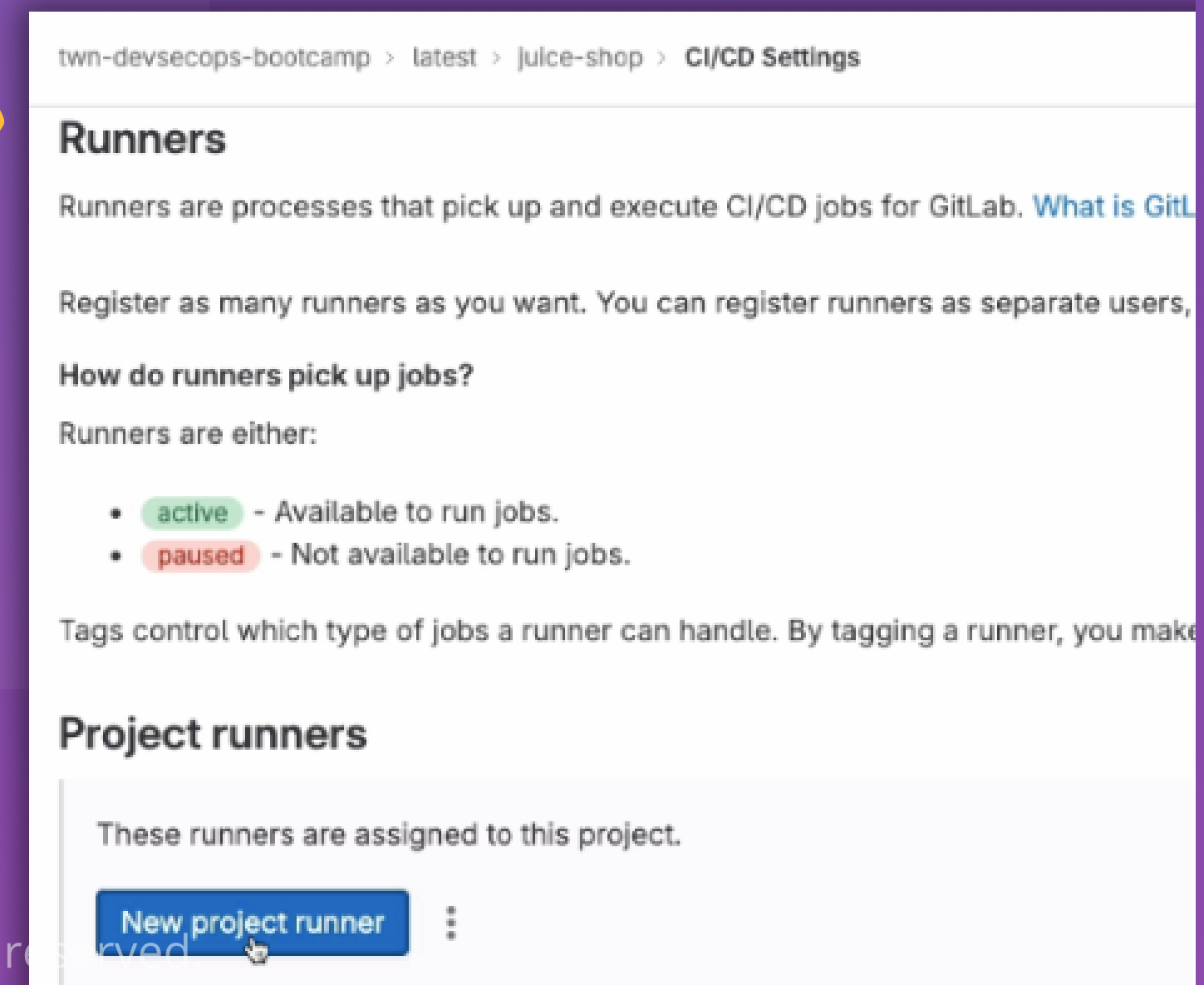
1. Create EC2 Instance for Self-Hosted Runner



	Name	Instance ID	Instance state	Instance type
<input type="checkbox"/>	app-server	i-00fef952b853e73b9	Running	t2.micro
<input type="checkbox"/>	gitlab-runner	i-07ad9251cfd3c8adf	Running	t2.large

2. Create **new project runner** on GitLab

a. This generates a command that registers the runner with all its configurations



tw-n-devsecops-bootcamp > latest > juice-shop > CI/CD Settings

### Runners

Runners are processes that pick up and execute CI/CD jobs for GitLab. [What is GitLab CI/CD?](#)

Register as many runners as you want. You can register runners as separate users, or as project runners.

#### How do runners pick up jobs?

Runners are either:

- active** - Available to run jobs.
- paused** - Not available to run jobs.

Tags control which type of jobs a runner can handle. By tagging a runner, you make it available for specific jobs.

### Project runners

These runners are assigned to this project.

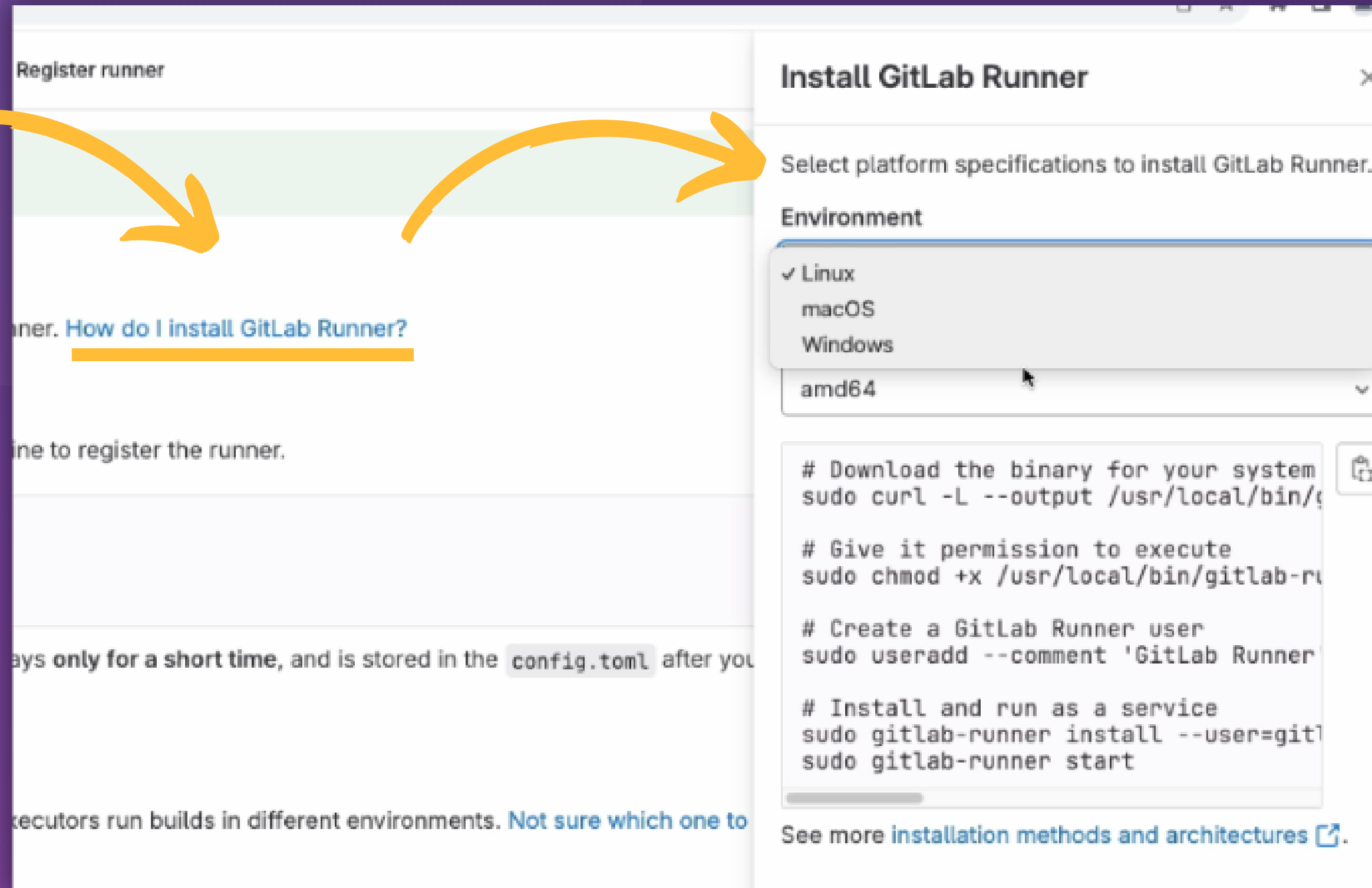
[New project runner](#)



# Register a self-managed runner

## How to - Part 2

### 3. Install GitLab Runner



The screenshot shows the 'Register runner' page on GitLab. The main content area has a link 'How do I install GitLab Runner?' which is highlighted with an orange underline. An orange arrow points from this link to the 'Install GitLab Runner' sidebar. The sidebar shows the 'Environment' section with 'Linux' selected, 'amd64' architecture, and a list of installation commands.

**Register runner**

Select platform specifications to install GitLab Runner.

**Environment**

- ✓ Linux
- macOS
- Windows

amd64

```
# Download the binary for your system
sudo curl -L --output /usr/local/bin/gitlab-runner https://gitlab-runner.gitlab.com/downloads/latest/gitlab-runner-linux-amd64

# Give it permission to execute
sudo chmod +x /usr/local/bin/gitlab-runner

# Create a GitLab Runner user
sudo useradd --comment 'GitLab Runner' --create-home --shell /bin/bash gitlab-runner

# Install and run as a service
sudo gitlab-runner install --user=gitlab-runner --service=systemd
sudo gitlab-runner start
```

See more [installation methods and architectures](#).

# Register a self-managed runner

## How to - Part 3

### 4. Register GitLab Runner on EC2 Instance

