CD Pipeline Overview and Intro to Docker

Video 4.3

What you will learn in this video

Theory:

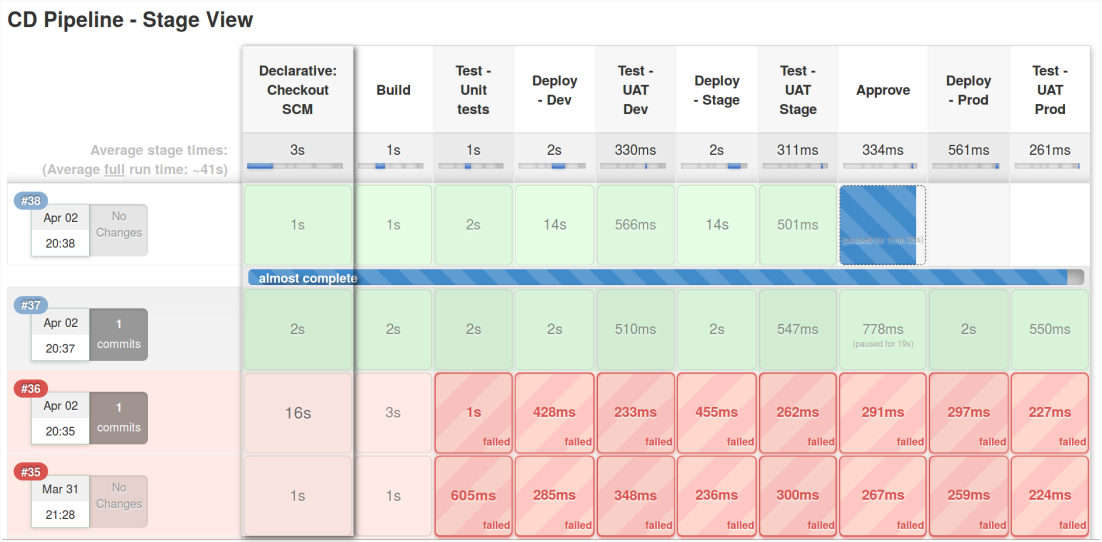
- CD Pipeline overview
- Web Application
- Docker

CD Pipeline Overview

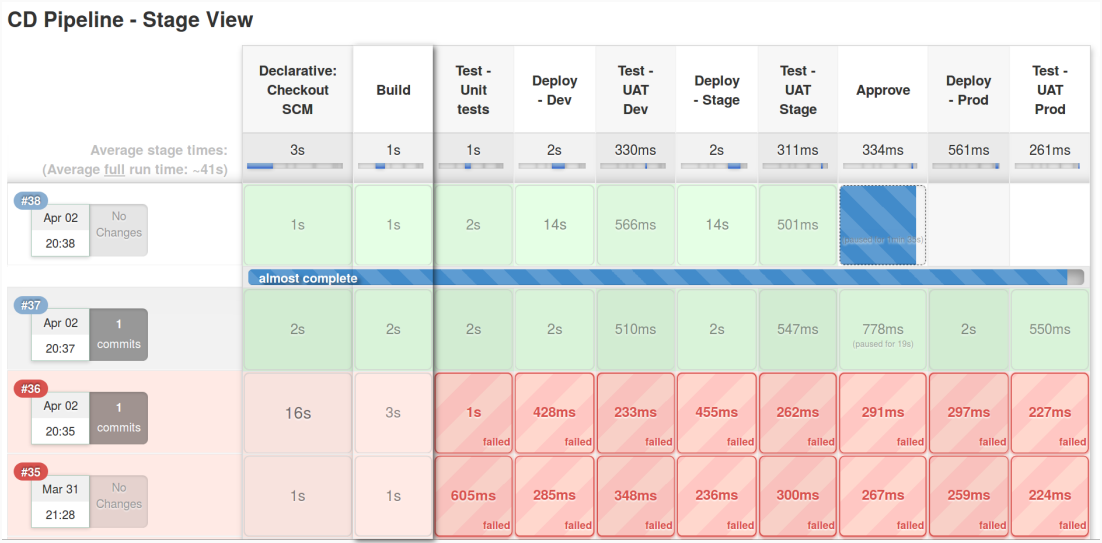
CD Pipeline - Stage View

	Declarative: Checkout SCM	Build	Test - Unit tests	Deploy - Dev	Test - UAT Dev	Deploy - Stage	Test - UAT Stage	Approve	Deploy - Prod	Test - UAT Prod
Average stage times: (Average full run time: ~41s)	3s	1s	1s	2s	330ms	2s	311ms	334ms	561ms	261ms
#38 Apr 02 20:38 — No Changes	1s	1s	2s	14s	566ms	14s	501ms	(paused for input)		
almost complete										
#37 Apr 02 20:37 — 1 commits	2s	2s	2s	2s	510ms	2s	547ms	778ms (paused for 19s)	2s	550ms
#36 Apr 02 20:35 — 1 commits	16s	3s	1s failed	428ms failed	233ms failed	455ms failed	262ms failed	291ms failed	297ms failed	227ms failed
#35 Mar 31 21:28 — No Changes	1s	1s	605ms failed	285ms failed	348ms failed	236ms failed	300ms failed	267ms failed	259ms failed	224ms failed

CD Pipeline Overview

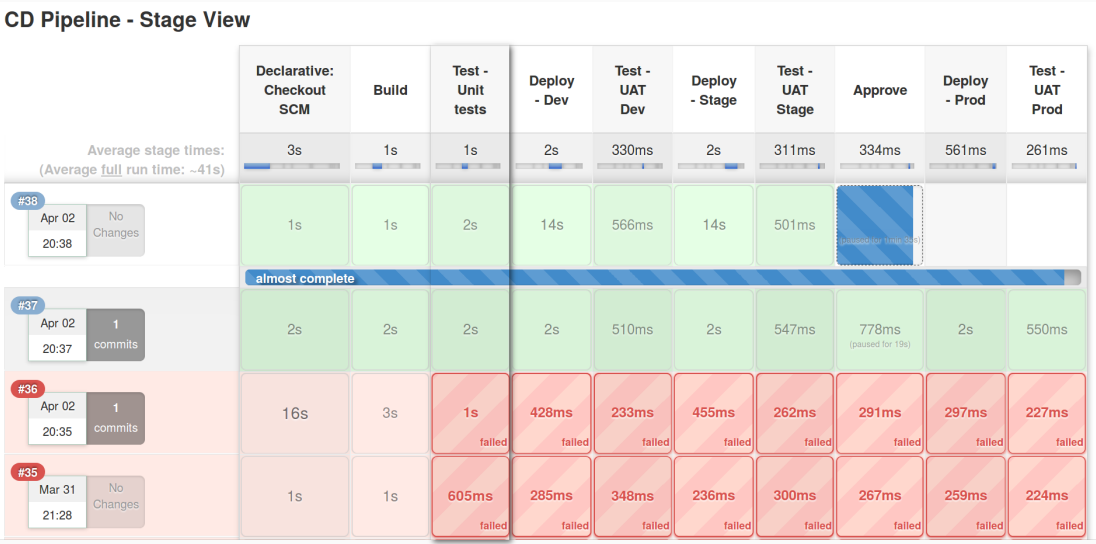


CD Pipeline Overview

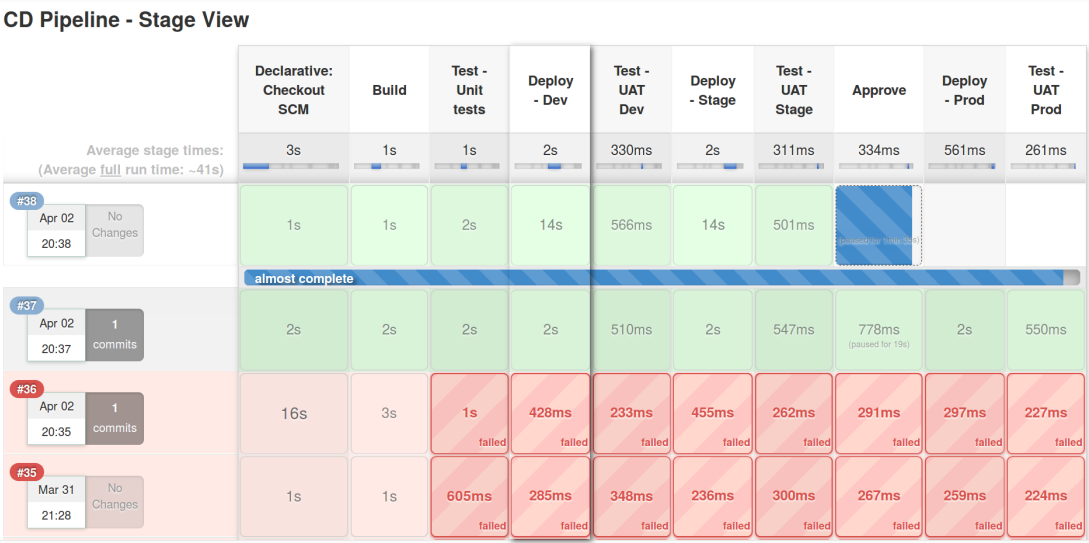


almost complete

CD Pipeline Overview



CD Pipeline Overview

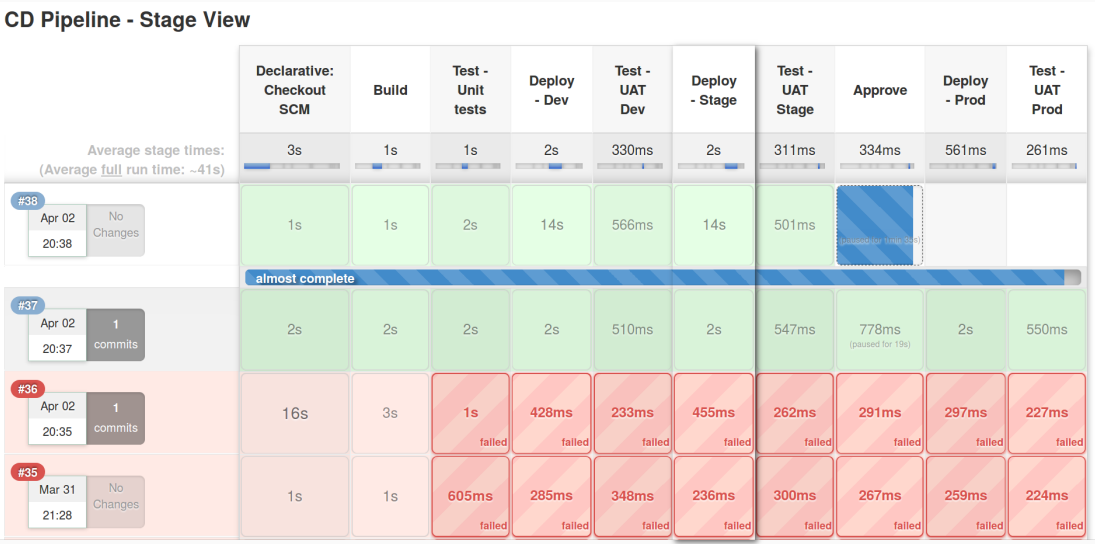


CD Pipeline Overview

CD Pipeline - Stage View

	Declarative: Checkout SCM	Build	Test - Unit tests	Deploy - Dev	Test - UAT Dev	Deploy - Stage	Test - UAT Stage	Approve	Deploy - Prod	Test - UAT Prod
Average stage times: (Average full run time: ~41s)	3s	1s	1s	2s	330ms	2s	311ms	334ms	561ms	261ms
#38 Apr 02 20:38 — No Changes	1s	1s	2s	14s	566ms	14s	501ms	Input requested (paused for 1min 58s)		
almost complete										
#37 Apr 02 20:37 — 1 commits	2s	2s	2s	2s	510ms	2s	547ms	778ms (paused for 19s)	2s	550ms
#36 Apr 02 20:35 — 1 commits	16s	3s	1s failed	428ms failed	233ms failed	455ms failed	262ms failed	291ms failed	297ms failed	227ms failed
#35 Mar 31 21:28 — No Changes	1s	1s	605ms failed	285ms failed	348ms failed	236ms failed	300ms failed	267ms failed	259ms failed	224ms failed

CD Pipeline Overview

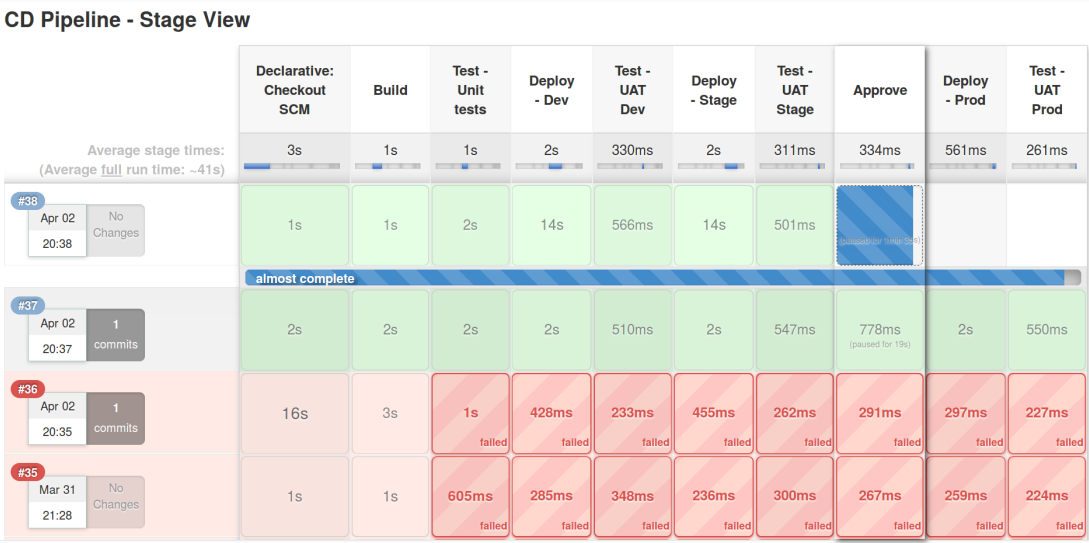


CD Pipeline Overview

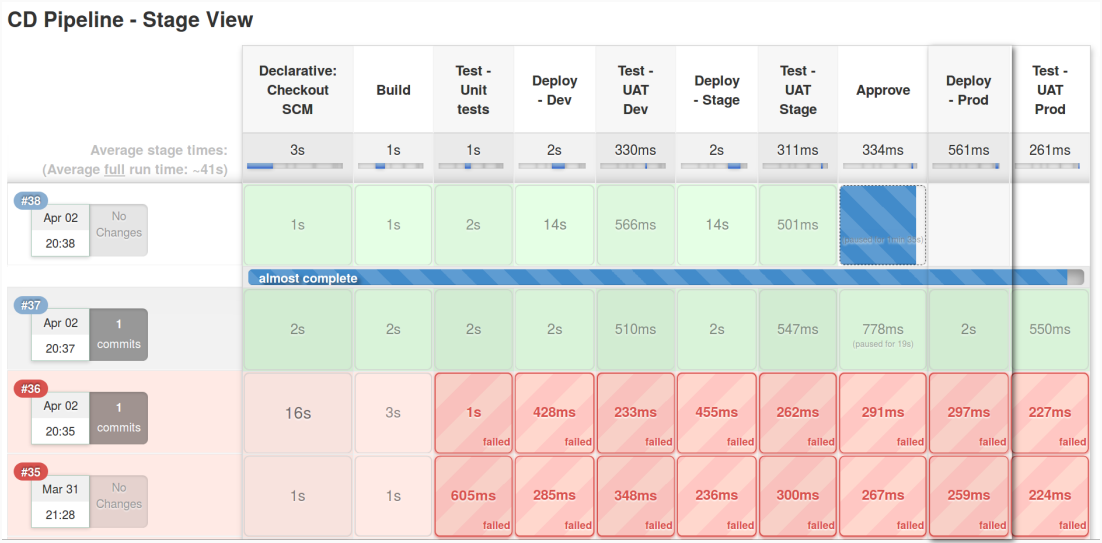
CD Pipeline - Stage View

	Declarative: Checkout SCM	Build	Test - Unit tests	Deploy - Dev	Test - UAT Dev	Deploy - Stage	Test - UAT Stage	Approve	Deploy - Prod	Test - UAT Prod
Average stage times: (Average full run time: ~41s)	3s	1s	1s	2s	330ms	2s	311ms	334ms	561ms	261ms
#38 Apr 02 20:38 — No Changes	1s	1s	2s	14s	566ms	14s	501ms	(paused for 1min 58s)		
almost complete										
#37 Apr 02 20:37 — 1 commits	2s	2s	2s	2s	510ms	2s	547ms	778ms (paused for 19s)	2s	550ms
#36 Apr 02 20:35 — 1 commits	16s	3s	1s failed	428ms failed	233ms failed	455ms failed	262ms failed	291ms failed	297ms failed	227ms failed
#35 Mar 31 21:28 — No Changes	1s	1s	605ms failed	285ms failed	348ms failed	236ms failed	300ms failed	267ms failed	259ms failed	224ms failed

CD Pipeline Overview

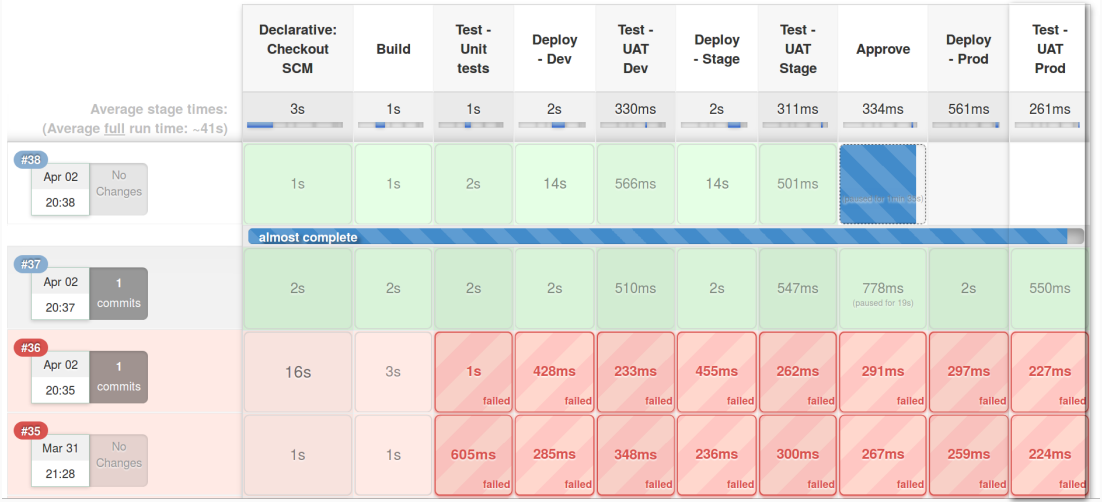


CD Pipeline Overview



CD Pipeline Overview

CD Pipeline - Stage View



What is Docker

What is Docker

- Docker is a technology for packaging and deploying **applications running inside containers**

What is Docker

- Docker is a technology for packaging and deploying **applications running inside containers**

- Docker provides development and operational teams with a **shared, consistent environment** for development, testing, and release

What is Docker

- Docker is a technology for packaging and deploying **applications running inside containers**
- Docker provides development and operational teams with a **shared, consistent environment** for development, testing, and release
- Docker avoids the classic '**but it worked on my machine**' issue

What is Docker

- Docker is a technology for packaging and deploying **applications running inside containers**
- Docker provides development and operational teams with a **shared, consistent environment** for development, testing, and release
- Docker avoids the classic '**but it worked on my machine**' issue
- Docker allows applications and their dependencies to be **moved portably across development, staging, and production environments**

Why using Docker for deployment?

Why using Docker for deployment?

- **Infrastructure as code**

Why using Docker for deployment?

- **Infrastructure as code**
- **Replicate applications and their configurations** accross environments

Why using Docker for deployment?

- **Infrastructure as code**
- **Replicate applications and their configurations** accross environments
- Avoid **dependencies conflicts**

Why using Docker for deployment?

- **Infrastructure as code**
- **Replicate applications and their configurations** accross environments
- Avoid **dependencies conflicts**
- Facilitate **portability and automation**

Why using Docker for deployment?

- **Infrastructure as code**
- **Replicate applications and their configurations** accross environments
- Avoid **dependencies conflicts**
- Facilitate **portability and automation**
- Docker **plays well with Jenkins**

Docker Images and Containers

Docker Images and Containers

Object-oriented programming analogy:

Docker Images and Containers

Object-oriented programming analogy:

- Docker **images are like classes**
- Docker **images are blueprints**
- Docker **images are like recipes**

Docker Images and Containers

Object-oriented programming analogy:

- Docker **images are like classes**
- Docker **images are blueprints**
- Docker **images are like recipes**
- Docker **containers are like objects**
- Docker **containers are concrete instances of Docker images**
- Docker **containers are like dishes**

Next Video

Video 4.3 Docker commands